Étude de transférabilité des clés pour le liage de données entre graphes de connaissances

Thibaut Soulard[1], Fatiha Saïs[1], Joe Raad[1], Gianluca Quercini[1]

[1] LISN, CNRS (UMR 9015), Université Paris Saclay, France

Résumé

Le liage de données dans des graphes de connaissances est un problème crucial et de longue date ; il consiste à déterminer des liens entre les descriptions des entités de ces graphes désignant une même entité du monde réel. Les clés, qui sont des sous-ensembles de propriétés permettant d'identifier chaque instance d'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 ensuite appliquer une procédure de fusion (e.g., le produit cartésien des clés). Mais cette approche peut être très coûteuse en temps et peut parfois conduire à très peu de clés communes entre les deux graphes. Dans ce travail, afin de réduire le temps de calcul de la découverte de clés et par conséquent de la tâche de liage de données, nous étudions la question de transférabilité des clés découvertes dans un graphe vers un autre graphe. Plus précisément, nous avons conduit des expérimentations sur DBpedia et Wikidata afin d'évaluer l'impact en performance de la transférabilité des clés à la fois en termes de temps de calcul et de qualité des clés transférées.

Mots-clés

Liage de données, découverte de clés, graphes de connaissances, transférabilité.

Abstract

Data linking in knowledge graphs is a crucial and long-standing problem ; it involves finding links between descriptions of entities in these graphs that refer to the same real-world entity. Keys, which are subsets of properties that identify each entity, are important elements in finding these identity links. The classical key-based approach if data linking is based on the discovery of a set of keys in each graph, and then apply a merge (e.g. intersection) of these keys. But this approach can be very time consuming and can sometimes lead to very few common keys between the two graphs. In this work, in order to reduce the computational complexity of the data linking task using keys, we study the issue of transferability of keys discovered in one graph to another graph. More precisely, we conducted experiments on DBpedia and Wikidata in order to evaluate the performance impact of key transferability both in terms of computation time and quality of transferred keys.

Keywords

Data linking, key discovery, knowledge graphs, transferability.

1 Introduction

Aujourd'hui, nous assistons à une production sans précédent de ressources, publiées sous forme de données ouvertes liées. Cela conduit à la création de graphes de connaissances (KG) contenant des milliards de triplets RDF (Resource Description Framework), comme DBpedia, YAGO et Wikidata du côté académique, et le Google Knowledge Graph ou eBay Knowledge Graph du côté commercial. Ces KG contiennent des millions d'entités (telles que des personnes, des protéines ou des livres) et des millions de faits les concernant. Ces KG sont soit indépendant du domaine, comme Yago, DBpedia ou Wikidata, ou spécifiques à un domaine, comme la géographie avec Geonames [1] ou la biologie avec Bioportal [2]. En 2020, le web de données contenait plus de 650 milles graphes de données reliés entre eux. Ces derniers contiennent des connaissances qui sont généralement exprimées en RDF [3], comme des faits de la forme `<subject, propriété, object>` 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, hasMayor). Parfois, les différents types et relations sont représentés dans une ontologie OWL2 (Web Ontology Language) [4], qui définit leurs interrelations et des axiomes tels que la subsomption, la disjonction et la fonctionnalité des propriétés.

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 parmi les liens les plus importants pour l'amélioration de la complétude des KGs, puisqu'ils permettent de relier des descriptions qui référent la même en-

1. https://en.wikipedia.org/wiki/GeoNames
2. https://bioportal.bioontology.org/
3. https://www.w3.org/RDF/
4. https://www.w3.org/OWL/

tité 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 [9, 4] pour une revue récente de la littérature).

En effet, détecter ces liens entre les entités des deux graphes G_1 et G_2 est un problème complexe, d'une part du fait de la nécessite 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 d'instances dans G_1 et G_2 respectivement).

Dans cet article, nous nous appuyons sur les travaux de liage de données fondés sur les clés de liage [11, 1] qui sont des sous-ensembles minimaux de propriétés permettant d'identifier chaque entité (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 donnée est double : (i) il permet de distinguer des sous-ensembles de propriétés ayant un fort impact sur le liage et qui pourront donc potentiellement déduire des liens d'identité avec un bon taux de précision, et (ii) il permet également de réduire le nombre de couples propriétés valeurs considérés lors de la comparaison des descriptions d'entités. Pour ce dernier point, cela peut s'avérer très efficace dans le cas où 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 soit celles-ci sont spécifiées par un expert du domaine ou bien découvertes automatiquement par des outils dédiés comme [15, 16, 3, 13]. Mais lorsque les graphes sont volumineux ou contiennent un grand nombre de propriétés, le coût en termes de temps de calcul de la découverte de clés, qui peut atteindre quelques jours, nécessite alors d'être réduit. C'est la raison pour laquelle, nous étudions dans ce travail s'il est possible d'économiser ce temps de calcul en appliquant un transfert de clés découvertes dans un graphe vers un autre graphe.

L'article est organisé comme suit : en section 2, nous présentons les travaux de l'état de l'art. 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

Dès lors que nous souhaitons combiner ou exploiter des données provenant de différentes sources, il devient alors important de comparer les données et détecter les descriptions qui réfèrent la même entité du monde réel (par exemple, la même personne, le même livre, le même gène). C'est ce que nous nommons liage de données.

Diverses approches ont été proposées dans la littérature, certaines appliquent des techniques de similarité [8], d'autres utilisent l'apprentissage automatique [7, 6] ou encore des connaissances du domaine déclarées dans une ontologie [11, 1].

Une des difficultés du liage de données réside dans l'hétérogénéité des données qui rend le calcul de similarité entre descriptions d'entité plus difficile. L'autre difficulté à laquelle les outils du liage de données doivent faire face est le nombre de combinaisons de descriptions d'entités et le coût de leur comparaison. En effet, le coût de la comparaison des descriptions d'entités peut avoir un impact important sur les performances en terme de temps de calcul de la tâche de liage d'entités entre graphes de connaissances. En effet, pour décider si deux descriptions réfèrent la même entité du monde réel, une tâche de comparaison deux-à-deux des couples propriétés valeurs décrivant les entités dans G_1 et G_2 se rend nécessaire. Cette tâche est d'autant plus difficile si les propriétés sont multi-valuées.

Les *clés* sont des éléments fondamentaux pour la découverte de liens entre entités. La spécification manuelle des clés est généralement irréalisable à l'échelle du Web, en raison du volume des ensembles de données et de l'hétérogénéité de leurs entités. C'est pourquoi plusieurs approches de découverte de clés ont émergé au fil des ans, avec des applications communes dans les bases de données relationnelles [12] et les graphes de connaissances [15, 16, 3].

Pour utiliser les clés pour lier les entités représentées dans deux graphes G_1 et G_2, une approche idéale serait de découvrir les clés dans G_1 et dans G_2 et de calculer une intersection, ou d'appliquer une procédure de fusion (voir [10] pour un exemple). Mais cette approche peut être très coûteuse en temps et peut parfois conduire à très peu de clés communes entre les deux graphes. Une autre approche [3] exploite simultanément les deux graphes et l'alignement de leur propriétés pour découvrir des Linkkey qui sont valides dans les deux graphes. Tout comme les approches classiques de découverte de clés, cette approche a besoin d'exploiter les données des deux graphes.

Afin de réduire la quantité de données exploitées par la découverte de clés et par conséquent le temps de calcul de cette tâche, dans ce travail, nous étudions la question de *transférabilité* des clés découverte dans un graphe G_1 vers un autre graphe G_2. Plus précisément, est-ce-qu'il est possible de déterminer l'ensemble de clés découvertes, et donc valides dans dans un graphes G_1, qui peuvent aussi être valides dans un graphe G_2. Ainsi, il serait possible de s'affranchir de l'application de la découverte de clés dans G_2 et donc de l'étape de fusion de clés. À notre connaissance, le problème de transférabilité des clés n'a pas encore fait l'objet d'études.

3 Définitions et notions

Dans cette section nous introduisons les notions importantes utilisées dans notre approche de transfert de clés.

3.1 Graphes de connaissances

Définition 1. (Graphe de connaissances RDF). Nous considérons un graphe de connaissances défini par un couple $(\mathcal{O}, \mathcal{G})$, où :

– $\mathcal{O} = (\mathcal{C}, \mathcal{P})$ est une ontologie représentée en OWL 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 triplets RDF décrivant des instances de classes de $\mathcal{O}$ formant un graphe de données RDF.

Définition 2. (Graphe de données RDF). Un graphe de données RDF $\mathcal{G}$ est un ensemble de faits représenté par des triplets de la forme {(sujet, prédicat, objet) | sujet $\in \mathcal{I}$, propriété $\in \mathcal{P}$, objet $\in \mathcal{I} \cup \mathcal{L}$}, où $\mathcal{I}$ est l'ensemble des entités désignés des IRIs, $\mathcal{P}$ est l'ensemble des propriétés, et $\mathcal{L}$ est l'ensemble des littéraux (tels que les nombres et les chaînes de caractères).

Définition 3. (Description RDF d'une entité). Considérons une entité d'un graphe RDF $\mathcal{G}$ représentée par un IRI $i \in \mathcal{I}$, sa description RDF est l'ensemble $D(i)$ défini comme suit : $D(i) = \{(p, v) \mid (i, p, v) \in \mathcal{G}$ *ou* $(v, p, i) \in \mathcal{G}$ (pour les propriétés p de type objet)$\}$. On notera $P(i)$ l'ensemble de propriétés p tel qu'il existe une valeur v avec $(p, v) \in D(i)$. On notera $V(i, p)$ l'ensemble de valeurs v d'une propriété p apparaissant dans $D(i)$.

Définition 4. (Alignement de propriétés). Un alignement entre deux propriétés p_1 et p_2 de $\mathcal{G}_1$ et $\mathcal{G}_2$ (resp.) est une relation de mise en correspondance exprimant une relation d'équivalence sémantique entre p_1 et p_2 que nous notons par : $p_1 \equiv p_2$.

3.2 Clés

Une clé est un ensemble de propriétés permettant d'identifier de façon unique chaque instance (entité) d'une classe. Dans les graphes de connaissances, les clés peuvent être exploitées pour la détection de liens d'identité `owl:sameAs` entre descriptions d'entités dans les graphes de données RDF.

En général, si un ensemble de propriétés est déclaré comme étant une clé pour une classe, la non-satisfaction de la clé dans un graphe de données peut être due à des erreurs dans les valeurs des propriétés ou à des doublons inconnus. Alors que lorsque des paires d'instances provenant de différents graphes RDF ne satisfont pas la clé, ces paires d'instances peuvent être considérées comme des candidates à l'établissement de liens d'identité. En effet, chaque paire d'instances qui partagent les mêmes valeurs pour toutes les propriétés d'une clé peuvent être considérées comme candidates au liage d'entités.

Différentes sémantiques de clés ont été proposées dans le domaine du web sémantique (voir [2] 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 qui est celle du constructeur `owl:hasKey` [5] qui est formalisée dans la définition suivante.

Définition 5. (Sémantique d'une *S*-clé) [2]. 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 :

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

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.

Définition 6. (Instantiation d'une clé). Soit k une *S*-clé$(C, (p_1, \ldots, p_n))$ pour une classe C dans $\mathcal{G}$. L'instantiation de la clé k pour une instance i de la classe C dans $\mathcal{G}$ est un n-uplet de valeurs de propriétés $\pi(k, i)$ définit comme suit :

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

4 Méthodologie de transfert de clés

Pour lier les instances d'un graphe G_1 aux instances d'un graphe G_2, l'approche classique de liage de données fondée sur les clés consiste, tout à d'abord, à découvrir un premier ensemble de clés K_1 dans le graphe G_1 et un autre ensemble de clés K_2 dans le graphe G_2, et d'appliquer une procédure de fusion de clés. La procédure de fusion peu simplement revenir au calcul de l'intersection des clés découvertes dans les deux graphes à la réécriture de propriétés équivalentes prés. Elle peut également être calculée, tel que proposé dans [10], par des produits cartésiens des ensembles de propriétés apparaissant dans K_1 et celles apparaissant dans K_2 et de conserver uniquement les minimales (une clé k_1 pour qui, il n'existe pas de clé k_2 tel que $prop(k_2) \subset prop(k_1)$. Il est important de noter que lorsque les graphes de données RDF sont décrits selon deux ontologies différentes, des alignements de propriétés (cf. définition 4) sont alors nécessaires pour réécrire les clés d'un graphe selon les propriétés alignées dans l'autre graphe (voir 4.1 la procédure d'alignement de propriétés que nous avons utilisée et la définition de la réécriture d'une clé).

Les clés obtenues de l'étape de fusion de clés sont alors considérées comme étant valides dans les deux graphes et peuvent par conséquent être utilisées pour lier les instances des deux graphes. Néanmoins, lorsque les graphes sont volumineux ou contiennent un grand nombre de propriétés le coût en termes de temps de calcul de la découverte de clés, qui peut atteindre quelques jours, nécessite alors d'être réduit. C'est la raison pour laquelle, nous avons étudié de manière expérimentale s'il est possible d'économiser ce temps de calcul en appliquant un transfert de clés découvertes dans un graphe vers un autre graphe. Pour ce faire, nous

5. `https://w3.org/TR/owl2-direct-semantics/`

6. Cela pourrait être également une expression de classe définie en OWL2 `https://www.w3.org/TR/owl2-direct-semantics/#Class_Expressions`

nous appuyons sur des mesures de qualité des clés qui sont le support et la discriminabilité que nous définissons, ci-après. Ces mesures permettent d'évaluer à quel point les clés lorsqu'elles sont transférées conservent leur propriétés d'unicité et leur capacité potentielle à produire des liens d'identité.

4.1 Alignement de propriétés

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 présenté 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 [5] 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 $m - n$ de propriétés qui permet de capturer, notamment, des cas où les ontologies contiennent des propriétés alternatives implicites et lorsque 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.

Cet alignement de propriétés peut ensuite être exploité pour réécrire les clés (voir définition 7) découvertes dans un graphe $\mathcal{G}$ en exploitant les propriétés alignées d'un graphe $\mathcal{G}'$.

Définition 7. (Réécriture d'une clé). Soient deux graphes $\mathcal{G}$ et $\mathcal{G}'$ et $\mathcal{M} = \{(p_1 \equiv p'_1), \dots (p_m \equiv p'_m)\}$ l'ensemble de m alignements de propriétés de $\mathcal{G}$ et $\mathcal{G}'$.

Soit k une *S*-clé$(C, (p_1, \dots, p_n))$ pour une classe C dans $\mathcal{G}$. Une réécriture $\rho(k, \mathcal{M})$ 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$(C', (p'_1, \dots, p'_n))$ tel que il existe un alignement de propriétés $m = \{(p_1 \equiv p'_1), \dots (p_n \equiv p'_n)\} \subseteq \mathcal{M}$ et que nous avons $C \equiv C'$.

4.2 Mesures de qualité des clés

Afin d'évaluer la qualité d'une clé, nous définissons ci-dessous le support d'une clé, son nombre d'exceptions et son taux de discriminabilité.

Définition 8. (Support d'une clé). Soit k une *S*-clé$(C, (p_1, \dots, p_n))$ 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 $prop(k)$. Plus formellement :

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

Le support d'une clé relativement au nombre d'instances de la classe est formellement défini comme suit :

$$support^R(k) = \frac{support(k)}{\mid \{x \mid \forall x, C(x) \in G\} \mid}$$

Le support permet de mesurer la couverture d'une clé en termes de nombre d'instances pour qui la clé pourrait générer un lien d'identité. 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 [3, 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 9. (Partition de l'ensemble d'instances d'une clé). Étant donnée une clé k =*S*-clé$(C, (p_1, \dots, p_n))$ dans un graphe de données RDF G, la partition de l'ensemble d'instances de C pouvant être formée par k est définie par l'ensemble : $\Delta(k, G) = \{\delta_1, \delta_2, \dots, \delta_m\}$. C'est l'ensemble de partitions d'instances pouvant être formées en regroupant dans chaque partition $\delta_i \in \Delta(k, G)$ le sous-ensemble d'instances partageant les mêmes valeurs pour $prop(k)$.

Définition 10. (Taux de discriminabilité d'une clé). Le taux de discriminabilité d'une clé k =*S*-clé$(C, (p_1, \dots, p_n))$ est le nombre de partitions $\delta_i \in \Delta(k, G)$ réduites à une seule instance relativement au nombre total de partitions de $\Delta(k, G)$. Plus formellement,

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

Lorsque les graphes de connaissances contiennent des redondances (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 [15] 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 [15].

Définition 11. (Nombre d'exceptions d'une clé). Le nombre d'exceptions d'une clé k de la forme *S*-clé$(C, (p_1, \dots, p_n))$ est le nombre d'instances de C qui ne satisfont pas la clé. Plus formellement :

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

La définition 11 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 12. (Nombre d'exceptions d'une clé relatif à son support). Le nombre 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 :

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

4.3 Approche de transfert de clés

Afin d'étudier la transférabilité des clés d'un graphe G_1 vers un graphe G_2, nous avons procédé selon les étapes suivantes.

(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 [15] qui découvre des *S*-clés en considérant un nombre d'exceptions passé en paramètre. Il est à noter, comme indiqué dans la section 2, SAKey étant basée sur le calcul de non-clés d'abord il ne permet pas de fournir des mesures quantitatives des clés telles que le support et la discriminabilité.

(2) Sélection des clés en fonction de leurs scores de qualité dans le graphe d'origine. 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 $ex^R(k)$ et de sélectionner seulement les clés qui respectent le taux d'exceptions relatif maximum ex^R_{max} passé en paramètres.

(3) Réécriture des clés sélectionnées en exploitant l'alignement de propriétés entre les deux graphes. 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.

(4) Évaluation de la qualité des clés réécrites dans le graphe cible. Enfin, pour chaque réécriture de clé retenue nous évaluons sur le graphe cible G_2 sa qualité en termes de taux d'exceptions relatif $ex^R(k)$ 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 $ex^R(k) > ex^R_{max}$, et de les écarter ensuite pour les étapes de liage d'entités.

5 Évaluation expérimentale

L'évaluation expérimentale dans ce travail a pour objectif d'évaluer si le gain en performance d'une approche de liage d'entités à base de clés ayant appliqué une procédure de transfert de clés du graphe source vers un graphe cible, impacte t-il la qualité des clés dans le graphe cible. En d'autres termes, il s'agit d'étudier comment la qualité des clés évolue quand elles sont transférées à un autre graphe.
Pour répondre à cette question nous avons fait varier le taux d'exception relatif toléré pour chaque clé découverte dans le graphe source. Ceci permet de limiter l'impact de la présence d'erreurs ou de doublons dans les graphes considérés sur les résultats. Plus précisément, pour chaque taux d'exceptions maximal ex^R_{max} (avec $ex^R_{max} \in [0, 5]$), nous procédons comme suit :

1. découverte de l'ensemble de clés K_1 de C_1 dans G_1 avec chaque clé ayant un taux d'exception < ex^R_{max}
2. alignement des propriétés dans K_1 avec les propriétés dans G_2
3. utilisation T_x, un sous-ensemble des clés de K_1 dont laquelle chaque clé dans T_x a toutes ses propriétés alignées au moins à une propriété dans G_2
4. calcul du support, du taux d'exception relatif et de la discriminabilité de chaque clé dans T_x dans G_1
5. calcul du support, du taux d'exception relatif et de la discriminabilité de chaque clé dans T_x dans G_2
6. comparaison des résultats des étapes 4 et 5

5.1 Jeux de données

Les jeux de données utilisés pour l'évaluation expérimentale proviennent de DBpedia [7] et de Wikidata [8].
Pour le premier jeu de données, nous avons extrait la partie du graphe RDF représentant les instances de la classe `Person` (i.e., ayant comme `rdf:type Person`) extrait en décembre 2022. Ce premier jeu de données est ainsi composé de 1,863,013 entités et 18,960 propriétés. Cependant, il est à noter que pour un nombre non négligeable de ces propriétés correspondent à un IRI mal encodé ou erroné (IRI qui ne correspond pas à un IRI d'une des ontologies qui décrit ce graphe) comme la présence de `http://dbpedia.org/ontology/award21n,2` au lieu de `http://dbpedia.org/ontology/award`. Ces propriétés sont alors supprimées du graphe.
Le deuxième jeu de données a été extrait du graphe Wikidata du 3 mars 2021 disponible sur rdfhdt [9]. Cette extraction a été réalisée pour les IRIs typés par le triplet `<?IRI, wdt:P31, wd:Q5>` (`wd:Q5` représente la classe "Human" dans Wikidata). Ce jeu de données est composé de 3,020,916 et 3,506 propriétés.
À ces jeux de données nous avons également considéré les ontologies décrivant les instances de personnes qui contiennent respectivement 4,604 et 10,472 propriétés.

7. `https://www.dbpedia.org/`
8. `https://www.wikidata.org/`
9. `https://www.rdfhdt.org/datasets/`

Nous avons appliqué un algorithme d'alignement de propriétés que nous avons développé (décrit dans [14]) et qui applique une mesure de similarité fondée sur les plongements sur la description textuelle associée aux propriétés et aux classes de ces deux ontologies.

Suppression des propriétés non-pertinentes. Après l'extraction des données, une étape de pré-traitement 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 est dépendant du nombre de propriétés dans le graphe. Pour ce faire, nous avons seulement gardé les propriétés utilisées dans les deux graphes où nous avions au moins un alignement de propriétés disponible. Nous avons gardé seulement les propriétés du type `owl:DatatypeProperty`, c'est à dire dont les valeurs sont des littéraux. Nous avons appliqué cette stratégie car une propriété non transférable ou commune aux deux graphes n'est pas utilisable pour une réécriture de clé. De plus, une clé qui est composée de `owl:objectProperty` n'est utilisable que par les outils exploitant la propagation de scores de similarité entre paires d'instances comme [1, 11]. Après ce pré-traitement, nous obtenons respectivement 239 et 135 propriétés pour DBpedia et Wikidata.

5.2 Résultats en termes de nombre de clés découvertes

Dans le tableau 1, nous présentons le nombre de Clés-Sources (C-S) pour chaque graphe de connaissances ainsi que le nombre de Clés-Réécrites (C-R) durant la vérification dans le graphe cible. Cette vérification est divisée en deux étapes, la première filtre les réécritures n'ayant aucune instantiation (cf. définition 6) (i.e. ayant un support égal à 0). Dans un second temps, nous vérifions aussi que ces réécritures respectent le taux d'exception maximal autorisé.
Le principal facteur de ce filtrage est directement liée au manque d'instantiation des réécritures dans le graphe cible. Ce manque peut être expliqué par une différence dans les données décrivant les entités, ainsi un graphe pourrait représenter les entités personnes d'un point de vue social et un autre d'un point de vue professionnel. Il est aussi explicable par un alignement $n - m$ de certaines propriétés trop lâches donnant ainsi une réécriture inutile car ces propriétés ne sont pas utilisées pour décrire un `humain` par exemple. Enfin, lors de la vérification du taux d'exceptions nous pouvons déjà avoir une première observation sur la dégénérescence des clés réécrites vers des Non-Clés Réécrites (i.e. ayant un $ex^R(k) > ex^R(k)_{max}$). Un exemple de ces Clés Réécrites et vérifiées est :
{(`wk:P2561≡db:name`),
(`wk:P1477≡db:originalName`) }

5.3 Résultats en termes d'ensemble de clés

Dans le tableau 2, nous pouvons observer l'évolution de la discriminabilité de l'ensemble des clés en fonction du taux d'exception relatif maximal. A partir de 0.5%, nous atteignons un plateau qui ne permet pas de discriminer un nombre plus important d'entités. Ce plateau est d'autant plus intéressant que pour des taux d'exceptions maximaux plus haut nous introduisons aussi plus de bruit (i.e. des exceptions) rendant la tâche final de liage d'entités plus difficile. De plus, ces données montrent également une certaine limitation de la méthode de découverte de clés avec un pourcentage de discriminabilité très faible pour les clés découvertes sur Wikidata appliquées sur ce même graphe, i.e. évaluation des mesures de qualité des clés sur le même graphe.

	ex^R_{max}	0%	0.5 %	2%	5%
DB	C-S	835	642	642	643
	C-R brut	1 572	1 199	1 199	1 200
	C-R (sup$\neq$ 0)	69	64	64	65
	C-R vérifiées	49	52	54	56
WK	C-S	357	237	238	240
	C-R brut	475	327	328	330
	C-R (sup$\neq$ 0)	82	82	82	83
	C-R vérifiées	41	48	49	55

TABLE 1 – Résultat de la découverte de clés dans DBpedia et Wikidata sur les instances représentant des personnes

	ex^R_{max}	0%	0.5 %	2%	5%
DB	DB	0.32%	80.72%	80.72%	80.72%
	WK	0.90%	28.88%	28.89%	28.89%
WK	WK	0.31%	0.88%	0.88%	1.03%
	DB	0.10%	80.73%	80.73%	80.73%

TABLE 2 – Pourcentage de discriminabilité sur l'ensemble des Clés-Réécrites découvertes dans X puis appliquées dans X & Y

5.4 Évolution du pourcentage d'exceptions après le transfert de clés

Dans la figure 1, nous montrons l'évolution du pourcentage d'exceptions en trois échelles : une réduction, une stabilité ou une augmentation du taux exceptions dans le graphe destination. Dans le cas du transfert des clés DBpedia vers Wikidata nous pouvons observer que les Clés-Réécrites ont tendance à garder leurs taux d'exceptions dans les deux graphes et donc à être relativement stable lors du transfert. Par exemple, pour les 65 clés découvertes dans DBpedia avec un taux maximale de 5%, 46 clés (70%) gardent le même taux d'exceptions quand elles sont transférées à Wikidata, 18 clés (27%) voient leur taux d'exceptions augmenter et une seule clé voit son taux d'exception diminuer. Néanmoins, dans le sens inverse nous avons un comportement bien différent avec une majorité de clés dégénérant en des Non-Clés-Réécrites. Ce comportement, impose donc à ce que les Clées-Réécrites soient bien évaluées dans le graphe cible pour s'assurer de la qualité des liens d'identité

potentiels. Enfin pour ces deux graphes le cas où la Clé-Réécrite réduit son taux d'exceptions est relativement rare.

5.5 Évolution de la discriminabilité après le transfert de clés

Dans la figure 2, nous montrons l'évolution du taux de discriminabilité d'une Clé-Réécrite en trois échelles, comme pour le pourcentage d'exceptions cas précédent : une réduction, une stabilité ou une augmentation du taux discriminabilité dans le graphe destination.
Sur ce deuxième indicateur nous observons un comportement plus régulier pour le transfert dans les deux sens, avec une majorité de Réécritures gardant un taux de discriminabilité stable.

5.6 Temps d'exécution pour chaque étape des stratégies et scénarios

Dans le tableau 3, nous présentons les différentes valeurs du temps d'exécution en fonction de la stratégie et du scénario suivi. Toutes les expériences ont été réalisé sur un CPU "Intel® Xeon® E5-2630 v4" avec 10 coeurs et 128 GB de RAM. Le point le plus important est le temps d'exécution de la découverte des Clés-Sources de DBpedia. Cette étape est le goulot d'étranglement pour la stratégie classique appliquant la fusion de clés découvertes dans les deux graphes. Par conséquent, en partant du graphe ayant le temps d'exécution le plus rapide pour la découverte de C-S, c'est à dire celui ayant le moins de propriété, (ici le graphe source serait Wikidata) nous pouvons drastiquement réduire le temps d'exécution total. Néanmoins, en partant de DBpedia nous perdons les bénéfices de cette nouvelle stratégie tout en restant tout de même dans des temps similaires à la méthode classique appliquant la fusion de clés.

5.7 Discussion

Au travers de cette évaluation expérimentale, nous avons pu analyser le comportement des clés réécrites du côté de la source et de la cible. Malgré la nécessité de vérification des clés réécrites dans le graphe cible nous avons pu montrer qu'une majorité de ces clés restent stables et ne dégénèrent pas en non clés. Néanmoins, avec la définition des exceptions relatives appliquée sur les clés trouvée par SAKey, impacte la monotonie des clés et par conséquent la complétude de l'ensemble des clés transférées. En effet, dans le tableau 4 la clé K_1 composée de $\{P_1, P_2\}$ respecte un taux maximal d'exceptions de 25% mais la clé K_2 composée de $\{P_1, P_2, P_3\}$ ne la respecte plus avec un support plus faible tout en ayant un même nombre d'exceptions donnant ainsi un taux d'exceptions de 100% alors que la clé $K_3 = \{P_1, P_2, P_3, P_4\}$ respecte ce taux.

La complétude est d'autant plus importante que des clés plus précises (i.e. ayant un support plus faible) peuvent différencier des entités que des clés plus générales (i.e. un support plus élevé) ne le peuvent pas. Un exemple de ce phénomène est illustré dans le même tableau 4, avec la clé K_1 qui ne peut pas différencier e_2 de e_3 là où K_3 le permet. Ainsi, nous pouvons espérer augmenter le taux de discriminabilité avec l'ajout de clé réécrites plus précises différenciant des cas particuliers. Cependant, cet ajout n'est pas trivial de part la perte de la monotonie des S-Clés avec le taux d'exception relatif. Mais bien-sûr cet ajout de clés engendrera une augmentation du temps d'exécution pour la découverte de clés ainsi que pour le liage d'entités, car plus de valeurs devront être comparées. Cette problématique est néanmoins commune à l'approche classique et celle appliquant le transfert de clés, ainsi elle n'a pas d'impact direct sur cet étude mais pourrait améliorer les résultats des approches basés sur les clés.

6 Conclusion et perspectives

Dans cet article, nous avons pu décrire une stratégie de transfert de clés pour éviter le recours à la découverte de clés systématique dans tous les graphes considérés lors du liage de données. Cette découverte de clés qui peut se révéler parfois très coûteuse en temps d'exécution. Nous avons aussi pu observer l'importance de l'évaluation des Clés-Réécrites dans le graphe cible pour éviter l'utilisation de Non-Clés Réécrites. Enfin, nous avons aussi pu montrer une limitation de l'utilisation des clés pour l'alignement d'entités avec un taux de discriminabilité qui peut se révéler très faible pour certain graphes.
Dans de futurs travaux, à très court termes nous envisageons d'évaluer la qualité des clés transférées en évaluant la qualité des liens d'identité (i.e. rappel, précision et F-mesure) qu'elles permettent de générer par le biais d'un outil de liage de données. Nous comptons également comparer les résultats de cette approche avec des approches qui découvrent des clés en prenant deux graphes simultanément tel que Linkkeys[3]. Enfin, nous souhaiterons développer une approche hybride pour le liage d'entités en explorant la possibilité d'utiliser les clés pour générer automatiquement un premier ensemble réduit de lien `schema:sameAs`. Ces liens pourront ensuite être utilisés pour l'initialisation des techniques d'alignement d'entités utilisant les plongements de graphes qui requiert un ensemble de liens entre les deux graphes. Nous voudrions aussi explorer la création d'un outil de découverte de clés prenant en compte les paramètres d'exception relatifs et permettant d'avoir toutes les clés respectant le taux défini.

Références

[1] Mustafa Al-Bakri, Manuel Atencia, Steffen Lalande, and Marie-Christine Rousset. Inferring same-as facts from linked data : An iterative import-by-query approach. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 9–15. AAAI Press, 2015.

[2] Manuel Atencia, Michel Chein, Madalina Croitoru, Jérôme David, Michel Leclère, Nathalie Pernelle, Fatiha Saïs, François Scharffe, and Danai Symeonidou. Defining key semantics for the RDF datasets : Expe-

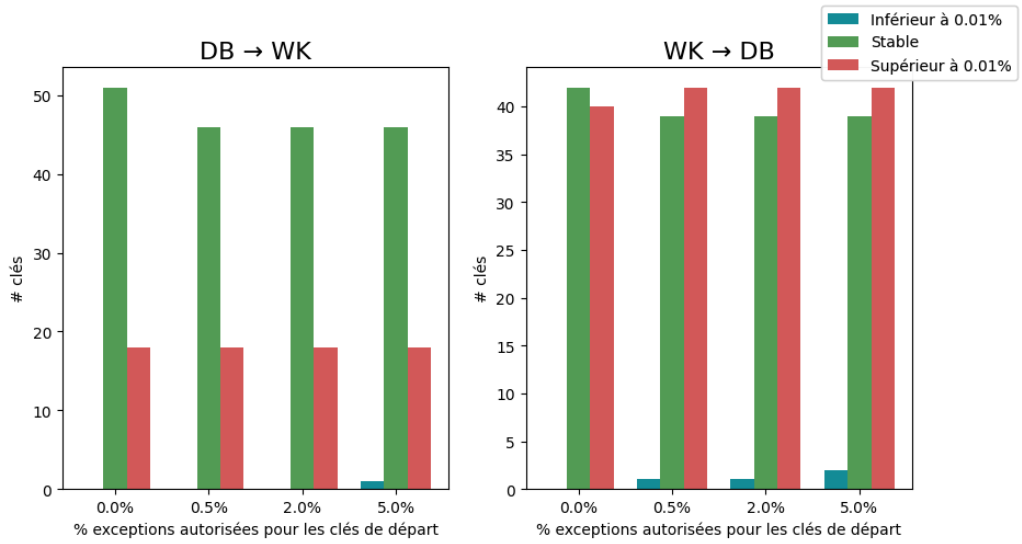

FIGURE 1 – Évolution du taux d'exceptions suite au transfert des clés découvertes

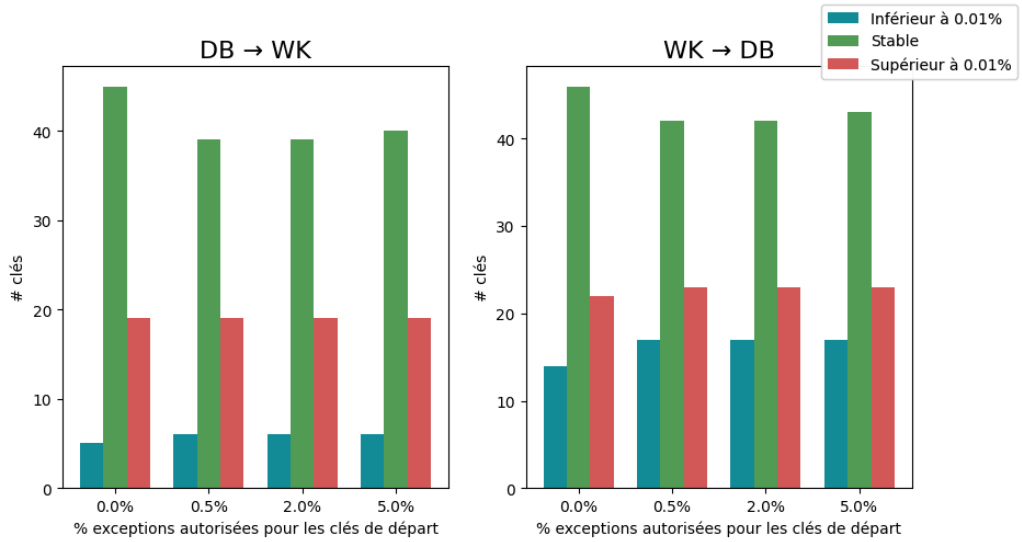

FIGURE 2 – Évolution du taux de discriminabilité suite au transfert des clés découvertes

	ex^R_{max}	0%	0.5 %	2%	5%
Classique DB ∩ WK	Découverte C-S DB	279.3	276.0	270.8	278.0
	Découverte C-S WK	5.1	5.0	5.1	5.0
	Évaluation C-S DB	33.7	25.3	25.4	25.4
	Évaluation C-S WK	3.4	1.5	1.5	1.5
	Total	321.7	308.0	302.9	310.1
Transfert Départ DB	Découverte C-S	279.3	276.0	270.8	278.0
	Évaluation C-S	33.7	25.3	25.4	25.4
	Évaluation. C-R	11.7	11.6	11.7	11.9
	Total	324.8	313.0	308.0	315.4
Transfert Départ WK	Découverte C-S	5.1	5.0	5.1	5.0
	Évaluation C-S	3.4	1.5	1.5	1.5
	Évaluation C-R	5.9	5.8	5.9	6.0
	Total	14.5	12.4	12.6	12.6

TABLE 3 – Temps d'excecution en minutes pour chaque scénario et étape.

	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	-	-

TABLE 4 – Exemple de non monotonie de clés.

riments and evaluations. In *Graph-Based Representation and Reasoning - 21st International Conference on Conceptual Structures, ICCS 2014, Iaşi, Romania, July 27-30, 2014, Proceedings*, pages 65–78, 2014.

[3] Manuel Atencia, Jérôme David, Jérôme Euzenat, Amedeo Napoli, and Jérémy Vizzini. Link key candidate extraction with relational concept analysis. *Discret. Appl. Math.*, 273 :2–20, 2020.

[4] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. An overview of end-to-end entity resolution for big data. *ACM Comput. Surv.*, 53(6), dec 2020.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding. 2018.

[6] Nikolaos Fanourakis, Vasilis Efthymiou, Dimitris Kotzinos, and Vassilis Christophides. Knowledge graph embedding methods for entity alignment : An experimental review, 2022.

[7] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, An-Hai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching : A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 19–34, New York, NY, USA, 2018. Association for Computing Machinery.

[8] Axel-Cyrille Ngonga Ngomo and Sören Auer. Limes—a time-efficient approach for large-scale link discovery on the web of data. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[9] Natasha Noy, Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. A survey of current link discovery frameworks. *Semant. Web*, 8(3) :419–436, jan 2017.

[10] Nathalie Pernelle, Fatiha Saïs, and Danai Symeonidou. An automatic key discovery approach for data linking. *Journal of Web Semantics*, 23 :16–30, 2013.

[11] Fatiha Saïs, Nathalie Pernelle, and Marie-Christine Rousset. Combining a logical and a numerical method for data reconciliation. In *Journal on Data Semantics XII*, pages 66–94. Springer, 2009.

[12] Yannis Sismanis, Paul Brown, Peter J Haas, and Berthold Reinwald. Gordian : efficient and scalable discovery of composite keys. In *Proceedings of the 32nd international conference on Very large data bases*, pages 691–702, 2006.

[13] Tommaso Soru, Edgard Marx, and Axel-Cyrille Ngonga Ngomo. Rocker : A refinement operator for key discovery. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, page 1025–1033, Republic and Canton of Geneva, CHE, 2015. International World Wide Web Conferences Steering Committee.

[14] Thibaut Soulard. Knowledge-based Entity Linking in Heterogeneous Knowledge Graphs at Web-Scale. Technical report, Université Paris Saclay, September 2022. Master thesis report.

[15] Danai Symeonidou, Vincent Armant, Nathalie Pernelle, and Fatiha Saïs. Sakey : Scalable almost key discovery in rdf data. In *International Semantic Web Conference*, pages 33–49. Springer, 2014.

[16] Danai Symeonidou, Luis Galárraga, Nathalie Pernelle, Fatiha Saïs, and Fabian Suchanek. Vickey : Mining conditional keys on knowledge bases. In *International Semantic Web Conference*, pages 661–677. Springer, 2017.