L'Apport Mutuel de la Combinaison des Tâches d'Interconnexion de Données et d'Alignement d'Ontologies pour l'Alignement Expressif

C.K. Jradeh[1], J. David[2], O. Teste[1], C. Trojahn[1]

[1] Université Toulouse 2 Jean Jaurès, IRIT
[2] Université Grenoble Alpes, INRIA

khadija.jradeh@irit.fr, jerome.david@inria.fr,
olivier.teste@irit.fr, cassia.trojahn@irit.fr

Résumé

Plusieurs méthodes ont été proposées pour aborder les tâches d'interconnexion de données et d'alignement d'ontologies, qui sont généralement traitées séparément. Dans cet article, nous présentons ***DICAP****, un algorithme qui permet leur collaboration mutuelle. Les expériences réalisées montrent que l'ajout de relations* owl:sameAs *résultant de l'interconnexion de données permet de découvrir des correspondances ontologiques supplémentaires. De plus, la présence de correspondances ontologiques permet l'extraction de règles de liage supplémentaires et discriminantes.*

Mots-clés

Alignement d'ontologies, interconnection de données.

Abstract

Several methods have been proposed to address the tasks of data interlinking and ontology matching. However, these tasks are usually handled separately. In this work, we present an algorithm that allows for their mutual collaboration. Experiments show that the addition of owl:sameAs *resulting from the data interlinking task allows to discover additional ontology alignments. Moreover, it reveals that the existence of ontology alignments allows for the extraction of additional discriminating linking rules.*

Keywords

Ontology matching, data interlinking.

1 Introduction

La représentation des connaissances basée sur les graphes de connaissance a gagné en popularité au cours des dernières décennies. Récemment, les graphes de connaissances sont utilisés par des moteurs de recherche comme Google [7], et par d'autres entreprises comme Amazon [15]. Les graphes de connaissances RDF représentent des données structurées sous la forme d'entités et de relations entre elles. Ces relations sont représentées sous forme de triplets. Un triplet est formé d'un sujet, d'un prédicat et d'un objet. Les sujets et les prédicats sont des ressources, mais les objets peuvent être des ressources ou des littéraux. Les ressources sont identifiées par des identificateurs de ressources internationalisés (IRI [1]). Les graphes de connaissance (*Knowledge Graphs*,ou KGs) sont souvent créés indépendamment les uns des autres. Par conséquent, ils peuvent contenir différents IRI qui font référence à la même entité du monde réel mais qui ne sont pas explicitement liés par la propriété d'équivalence owl:sameAs.
La propriété owl:sameAs permet de déclarer que deux IRI différentes font référence à la même entité du monde réel. Le fait de lier ces IRI permet de compléter les graphes RDF. Le problème de liaison de différentes entités à travers différents graphes s'appelle l'interconnexion des données. De même, les ontologies sur lesquelles les graphes de connaissances s'appuient peuvent décrire des concepts et des propriétés qui sont reliées sémantiquement. Le problème de la recherche de relations entre les entités d'ontologies s'appelle l'alignement d'ontologies.
De nombreuses méthodes ont été proposées dans la littérature pour résoudre l'interconnexion des données et l'alignement d'ontologies. Cependant, ces tâches sont généralement effectuées séparément. Dans cet article, nous étudions le bénéfice mutuel de la coopération entre ces deux tâches. Plus précisément, nous étudions les questions sui-

1. Internationalised Resources Identifier

vantes dans un contexte, encore peu étudié dans la littérature, des alignements expressifs :

1. Que peut apporter l'alignement d'ontologique à la tâche d'interconnexion des données ?
2. Que peut apporter l'interconnexion des données pour la tâche de mise en correspondance des ontologies ?

Dans ce but, nous proposons l'algorithme **DICAP** qui, étant donné une paire de graphes de connaissances et leurs ontologies, permet d'effectuer à la fois l'interconnexion des données et l'alignement d'ontologies. Pour la tâche d'interconnexion des données, nous choisissons **Linkex** [4] et pour la tâche de mise en correspondance des ontologies, nous choisissons **CANARD** [23]. **Linkex** produit un ensemble de clés de liage, qui à leur tour sont utilisées pour établir des liens de type owl:sameAs entre les individus des graphes de connaissances. **CANARD** produit un ensemble de correspondances complexes entre les ontologies. **Linkex** utilisera ces correspondances pour générer un ensemble de clés de liage, chaque clé étant applicable à une paire de classes spécifique définie dans la correspondance. Nous supposons que la combinaison de **CANARD** et **Linkex** permet d'identifier un plus grand nombre de clés de liage discriminatoires, ainsi que la découverte de correspondances plus complexes.

Le reste de l'article est organisé comme suit : la section 2 résume et discute les travaux liés qui ont été proposés pour l'interconnexion des données et l'alignement d'ontologies. La section 3 donne les définitions et explique les techniques utilisées dans cet article. La section 4 décrit l'algorithme **DICAP**. La section 5 décrit les évaluations réalisées et discute leur résultats. Enfin, la section 6 résume l'article et présente les directions pour les travaux futurs.

2 Travaux liés

Différentes méthodes ont été proposées pour résoudre le problème de l'interconnexion des données [24, 16, 20, 3, 12]. Ces méthodes se répartissent en deux grandes catégories : les méthodes numériques et les méthodes logiques. Les approches numériques [24, 16] réduisent la tâche d'interconnexion des données à une tâche de calcul de similarité. Les approches logiques [20, 3, 12] définissent, quant à elles, un ensemble de règles ou d'axiomes permettant d'inférer des égalités entre individus.

Les *approches numériques* calculent la similarité entre deux entités qui appartiennent à des graphes de connaissances différents. La similarité entre deux entités est calculée par des fonctions de similarité, basées sur les valeurs des propriétés de la paire d'entités donnée. Les entités qui sont suffisamment similaires sont considérées comme identiques et sont liées par la propriété owl:sameAs. Certaines approches numériques comme Silk [24] permettent à l'utilisateur de spécifier tous les conditions que les entités doivent remplir pour être liées. D'autres, comme Limes [16] utilisent différentes méthodes pour découvrir automatiquement des spécifications supplémentaires.

Les *approches logiques* sont, quant à elles, divisées en approches basées sur les règles et sur les clés. Les premières sont basées sur des règles permettant de dériver des liens d'identité à partir des graphes d'entrée et de leurs ontologies [20, 3, 12]. Les approches basées sur les clés visent à extraire un ensemble de clés. Chaque clé est constituée d'un ensemble de propriétés et d'une classe, les propriétés permettant d'identifier de manière unique une instance qui appartient à la classe spécifiée. Selon cette définition, deux instances qui ont les mêmes valeurs pour les propriétés d'une clé sont considérées comme identiques. Plus précisément, une clé a la forme $(\{p_1, ..., p_k\}\ \mathsf{key}\ C)$ où $p_1, \cdots, p_k$ sont des propriétés et C une classe. Un exemple de clé est le suivant :

$$(\{\mathsf{creator}, \mathsf{title}\}\ \ \mathsf{key}\ \ \mathsf{Work})$$

indiquant que lorsque deux instances de la classe Work partagent respectivement les valeurs du rôle creator et du rôle title, elles désignent la même entité.

Pour utiliser des approches basées sur les clés afin de relier une paire d'ensembles de données, les clés candidates sont d'abord extraites des ensembles de données, puis les meilleures clés candidates sont sélectionnées en fonction de différentes mesures de qualité [21, 9, 2]. Lorsque les graphes RDF sont décrits à l'aide de la même ontologie, les clés peuvent être utilisées directement pour interconnecter ces graphes RDF. Par contre, pour interconnecter des graphes RDF qui sont décrits à l'aide de différentes ontologies, les clés doivent être combinées avec des alignements d'ontologies qui relient les propriétés et les classes. Ainsi pour interconnecter deux graphes RDF il est nécessaire d'avoir soit la même ontologie décrivant les deux graphes, soit un alignement entre les ontologies. Les clés de liage (Section 3.3) permettent de surmonter cette limitation.

D'autre part, les méthodes d'alignement d'ontologies sont utilisées pour trouver des corres-

pondances entre les entités de deux ontologies. Il existe différents types de méthodes d'alignement [19, 10, 11], notamment les méthodes basées sur les instances, lexicales, structurales, sémantiques, et hybrides.
Il existe deux types de correspondances, simples et complexes. Une correspondance simple fait référence à une relation sémantique de base entre deux concepts ou propriétés. Une correspondance complexe fait référence à un relation entre deux ontologies qui implique plusieurs classes ou propriétés. Il existe plusieurs méthodes pour extraire des correspondances simples [10, 17] et complexes [11, 19].
L'article [10] présente un outil appelé AgreementMakerLight AML, utilisé pour aligner automatiquement des ontologies. L'outil utilise diverses mesures pour comparer les concepts et les relations entre différentes ontologies, telles que des mesures de similarité lexicale, syntaxique et sémantique. Il utilise également des sources de connaissances externes et des techniques d'apprentissage supervisé pour améliorer le processus d'alignement. Le papier évalue l'outil en utilisant des bancs d'essai standard et montre qu'il surpasse plusieurs outils d'alignement d'ontologies de pointe. Cependant, cela ne permet pas de générer des alignements complexes. Correspondances complexes ont été identifiées dans divers domaines, comme les ontologies médicales [13].
AMLC [11] permet de générer des alignements complexes. AMLC est une version de l'AML développée pour la correspondance d'ontologies complexes. AMLC comprend une implémentation d'un algorithme de correspondance d'ontologies basé sur des règles d'association, qui effectue une extraction de motifs.
Il existe peu d'approches comme PARIS [22] qui aligne les instances, les relations et les classes. La méthode calcule la probabilité d'équivalence des instances et des propriétés des différentes ontologies considérées de manière itérative jusqu'à ce qu'elle atteigne la convergence. Ensuite, elle calcule la probabilité d'équivalence entre les classes.
DICAP, d'autre part, se concentre sur l'amélioration des résultats de **CANARD** et **Linkex**. Il fournit à **CANARD** les relations owl:sameAs nécessaires pour extraire de nouvelles correspondances. Il fournit également à **Linkex** des correspondances, ce qui lui permet d'extraire des clés de liage entre les classes équivalentes.
L'impact des relations owl:sameAs sur la tâche de correspondance d'ontologies a été étudié en [18]. Les expériences menées montrent que l'inclusion de liens owl:sameAs a un impact positif sur les approches de correspondance de schémas basées sur les instances en augmentant la distance de JACCARD [2] entre les classes de l'ontologie considérée.
Dans notre approche proposée, nous étudions d'une part l'impact de la relation owl:sameAs sur **CANARD**, qui relève des méthodes basées sur les instances pour la tâche de correspondance d'ontologies. D'autre part, nous étudions l'impact de l'utilisation de la correspondance sur l'extraction de clé de liage effectuée par **Linkex**.

3 Préliminaires

Dans cette section, nous introduisons les définitions nécessaires à la compréhension du reste de l'article. Nous décrivons également les systèmes **CANARD** (Complex Alignment Need and A-box based Relation Discovery) et **Linkex** qui sont utilisés par l'algorithme que nous proposons. **CANARD** traite la tâche de mise en correspondance d'ontologies en produisant des correspondances expressives entre une paire d'ontologies peuplées par des instances. **Linkex** est capable d'aborder la tâche d'interconnexion des données en produisant un ensemble de clés de liage, ces clés de liage représentent des axiomes permettant de relier des entités à travers une paire de graphes de connaissances différents.Nous commençons par définir une correspondance simple et les correspondances complexes.

3.1 Correspondances simples et complexes

Soient o_1 et o_2 deux ontologies. Une correspondance c est définie par un quadrulet $\langle e_{o_1}, e_{o_2}, r, n\rangle$ où e_{o_1} et e_{o_2} sont des membres de la correspondance. Ils peuvent être des expressions simples ou complexes entre les entités e_{o_1} et e_{o_2} :

1. une correspondance est dite simple, si e_{o_1} et e_{o_2} sont toutes deux des expressions simples (atomiques) ;
2. une correspondance est dite complexe, si au moins l'une des expressions e_{o_1} ou e_{o_2} est une expression complexe ;
3. r est une relation entre e_{o_1} et e_{o_2}, par exemple l'équivalence ($\equiv$), plus générale ($\sqsubseteq$), plus spécifique ($\sqsupseteq$) ;
4. une valeur n (généralement comprise entre [0,1]) peut être associée à la correspondance c pour indiquer le degré de confiance que la relation r existe entre e_{o_1} et e_{o_2}.

2. La distance de JACCARD mesure la similarité entre deux ensembles(classes). Elle est définie comme la différence entre la taille de l'intersection des classes et la taille de leur union, divisée par la taille de l'union.

La correspondance

$$\langle \text{o1:AcceptedPaper}, \text{o2:Paper} \sqcap \exists \text{o2:hasDecision.o2:Acceptance}, \equiv, 0.8 \rangle$$

est une correspondance complexe entre l'expression simple o1:AcceptedPaper et l'expression complexe o2:Paper ⊓ ∃o2:hasDecision.o2:Acceptance. Elle indique que le concept AcceptedPaper dans $o1$ est équivalent au concept Paper ⊓ ∃hasDecision.Acceptance dans $o2$. Le concept Paper ⊓ ∃hasDecision.Acceptance désigne les entités qui appartiennent au concept Paper et qui ont une relation hasDecision ayant la valeur Acceptance. Le degré de confiance de cette correspondance est de 0,8.

3.2 Le système CANARD

CANARD [3] est une approche d'alignement qui permet de découvrir des correspondances complexes entre des ontologies peuplées en se basant sur des questions de compétences pour l'alignement (CQAs). Les CQAs représentent les besoins en connaissances d'un utilisateur. L'approche prend en entrée une paire de KGs, source et cible, leurs ontologies et les CQAs (exprimées en SPARQL) définies par l'utilisateur. Elle renvoie en sortie un ensemble de correspondances au format EDOAL [4].
L'approche est composée par plusieurs étapes :

1. transformation des CQAs en énoncés de Logique Descriptive [6] (LD) et extraction des leurs informations lexicales ;
2. récupération des instances concernées dans le KG source ;
3. extraction des descriptions des instances liées dans les KG cibles (relation owl:sameAs) ;
4. calcul de la similarité entre les descriptions des instances sources et cibles, en conservant les triplets dont la similarité est supérieure à un seuil donné ;
5. agrégation des triplets et leur traduction en DL. Chaque correspondance est transformée en une déclaration EDOAL avec une valeur de confiance, cette valeur de confiance est calculée sur la base de la similarité entre les LD source et cible.

3. https ://gitlab.irit.fr/melodi/ontology-matching/complex/

4. EDOAL est un langage d'alignement ontologique expressif et déclaratif qui permet de représenter des alignements d'ontologies [8].

3.3 Clés de liage

Les clés de liage sont des axiomes utilisés pour générer des liens entre une paire de graphes RDF décrits à l'aide de différentes ontologies [4]. Une clé de liage entre deux graphes RDF KG_1 et KG_2 est une expression de la forme

$$(\{\langle P_1, Q_1\rangle, ..., \langle P_n, Q_n\rangle\} \text{ linkkey } \langle C, D\rangle)$$

où ⟨C, D⟩ est une paire de concepts appartenant respectivement à KG_1 et KG_2 et $\langle P_1, Q_1\rangle, ..., \langle P_n, Q_n\rangle$ est une séquence non vide de paires de propriétés où pour chaque $\langle P_i, Q_i\rangle$ dans $\{\langle P_1, Q_1\rangle, ..., \langle P_n, Q_n\rangle\}$, P_i appartient à KG_1 et Q_i appartient à KG_2. Elle stipule que si deux entités appartenant respectivement aux concepts C et D partagent au moins une valeur pour chaque paire de propriétés $\langle P_i, Q_i\rangle$ éventuellement multivaluées alors elles sont identiques.
Un exemple de clé de liage est :

$$(\{\langle \text{creator}, \text{auteur}\rangle, \langle \text{title}, \text{titre}\rangle\} \text{ linkkey } \langle \text{NonFiction}, \text{Essai}\rangle) \quad (1)$$

en déclarant que lorsqu'une instance de la classe NonFiction et une instance de la classe Essai, partagent des valeurs pour les rôles auteur et author, et pour les rôles title et titre, respectivement, elles désignent la même entité. Dans ce cas on dit que ces instances satisfont la condition de la clé de liage (1).
Les clés de liage peuvent être construites par des experts du domaine ou extraites automatiquement de deux ensembles de données [1, 4, 5]. Une fois obtenues, les clés de liage peuvent être transmises à un outil de génération de liens tel que [20] pour générer l'ensemble des liens d'identité.

3.4 Le système Linkex

Linkex [5] est un logiciel parmettant d'extraire des clés de liage à partir d'une paire de KG RDF source et cible. L'algorithme fonctionne comme suit : d'abord les triplets de chaque KG (source et cible) sont indexés sous la forme de o → sp (object→ subject properties). Ensuite, l'algorithme itère sur les entrées communes aux deux index pour génèrer un troisième index associant chaque paire de sujets à son ensemble maximal de propriétés pour lequels les paires de sujets partage au moins une valeur. Ce troisième index sert d'ensemble de descriptions, ou contexte formel, à partir duquel un treillis de concepts est calculé. Chaque intent du treillis de concepts résultat représente une clé de liage candidate et l'extent associé correspond à l'ensemble de liens générés par

5. https ://gitlab.inria.fr/moex/linkex/

cette clé de liage candidate. Finalement, les clés de liage candidates peuvent être filtrées grâce à des mesures d'estimation de leur qualité comme la couverture et la discriminabilité [4].
La couverture est définie comme la proportion d'instances des deux classes qui pourraient être liées par la clé de liage. La discriminabilité mesure la proximité d'une clé de liage candidate à un appariement 1 à 1. L'utilisation à la fois de la couverture et de la discriminabilité établit un équilibre entre la précision et la généralité des clés de liage candidates. A l'instar de la précision et du rappel, ces mesures peuvent être agrégées par la moyenne harmonique (hmean).

4 L'algorithme DICAP

Cette section présente **DICAP** [6], un algorithme qui vise à intégrer les pipelines d'interconnexion de données et d'alignement complexe des ontologies. L'algorithme prend en entrée une paire de graphes de connaissances KG_1 et KG_2, un ensemble de CQA et un seuil de confiance ρ.
Il fonctionne comme suit, tout d'abord, il appelle les systèmes **CANARD** et **Linkex** qui renvoient, respectivement, un ensemble d'alignements CC et de clés de liage Lks.
Ces correspondances et clés de liage sont ensuite utilisées pour saturer KG_1 et KG_2 en utilisant les algorithmes 2 et 3. Comme indiqué dans la ligne 1, l'algorithme utilise un booléen, appelé enter, initialisé à true, qui permet à l'algorithme d'entrer dans la boucle.
L'algorithme appelle également **Linkex** pour générer un ensemble de clés de liage entre la paire de classes présentes dans chaque correspondance (Lignes 6 à 8). Après, si il n'y a pas de nouvelles correspondances ou de clés de liage générées, enter est mis à false. Cela signifie que l'algorithme a atteint un état stationnaire (aucune clé de liage ou correspondance supplémentaire ne peut être extraite) et que les graphes de connaissances ne peuvent plus être enrichis. En conséquence, aucune nouvelles clés de liage ou alignements ne peuvent être extraits des graphes de connaissance.
L'algorithme 2 permet de saturer les graphes de connaissances en utilisant les correspondances. Il fonctionne simplement en ajoutant des assertions de concepts pour les individus appartenant à l'un des concepts présents dans une correspondance.
Cela permet d'extraire des clés de liage entre des classes équivalentes simples et complexes. Les assertions de classes complexes ne sont généralement pas explicitement indiquées dans les graphes de connaissances, ce qui ne permet pas d'extraire des clés de liage entre elles.

Algorithme 1 : DICAP

Input : Une paire de graphes de connaissances KG_1 et KG_2, un ensemble de CQA et un seuil de confiance ρ.
Output : Un ensemble d'alignements et un ensemble de clés de liage entre KG_1 et KG_2.

```
1  Lks← ∅, CC ← ∅, enter ← true;
2  while enter do
3      CC ← CANARD(KG1, KG2, CQA, ρ);
4      Lks ← Linkex(KG1, KG2);
5      Appeler l'algorithme 2 et l'algorithme 3
         pour saturer KG1 et KG2 en utilisant CC
         (KG1 devient KG1' et KG2 devient KG2') ;
6      for ⟨c1, c2, r, n⟩ ∈ CC do
7          Lks ← Linkex(KG1', KG2', c1, c2);
8      end
9      if Il n'y a pas de nouvelles clés de liage
         ou de correspondances générées à partir
         de KG1' et KG2' then
10         enter ← false;
11     end
12     KG1 ← KG1', KG2 ← KG2'
13 end
14 return CC, Lks;
```

L'algorithme 3 permet de saturer les graphes de connaissances en utilisant des clés de liage. Premièrement il ajoute la relation owl:sameAs entre les individus satisfaisant la condition de clé de liage ou par les mêmes individus par transitivité. Ensuite, il ajoute de nouvelles assertions de concept et de rôle impliquées par la présence de relations owl:sameAs. Nous utilisons $s+$ pour désigner la fermeture transitive de l'individu s par rapport à la relation $\approx$ (apparaissant dans les assertions), c'est-à-dire que $s+$ est l'ensemble tel que $s \in s+$, et si c owl:sameAs b ou b owl:sameAs c avec un certain $c \in s+$ alors $b \in s+$.
L'ajout de ces liens owl:sameAs permet à **CANARD** de trouver des correspondances entre le graphe de connaissances source et cible. En l'absence de ces liens, **CANARD** ne pourra pas trouver de correspondances.

5 Expérimentation

Dans cette section, nous décrivons d'abord l'architecture de **DICAP** et ensuite nous discutons les expériences menées.

6. https ://github.com/dace-dl-anr/DICAP

Algorithme 2 : Saturate with Correspondences

Input : Une paire de graphes de connaissances KG_1 et KG_2 et un ensemble de correspondances CC.

Output : Une paire de graphes de connaissances KG'_1 et KG'_2 telle que $\mathsf{KG}_1 \subseteq \mathsf{KG}'_1$ et $\mathsf{KG}_2 \subseteq \mathsf{KG}'_2$.

$\mathsf{KG}'_1 \leftarrow \mathsf{KG}_1$, $\mathsf{KG}'_2 \leftarrow \mathsf{KG}_2$;
for $\langle c_1, c_2, r, n\rangle \in \mathsf{CC}$ *s'il existe un individu* a *tel que* $c_1(a) \in \mathsf{KG}'_i$ *pour* $i \in \{1, 2\}$ **do**
  Ajouter $c_2(a)$ à KG'_i;
end
return $\mathsf{KG}'_1, \mathsf{KG}'_2$;

Algorithme 3 : Saturate with Link keys

Input : Une paire de graphes de connaissances KG_1 et KG_2, un ensemble de clés de liage Lks et un seuil de confiance de ρ.

Output : Une paire de graphes de connaissances KG'_1 et KG'_2 tels que $\mathsf{KG}_1 \subseteq \mathsf{KG}'_1$ et $\mathsf{KG}_2 \subseteq \mathsf{KG}'_2$.

$\mathsf{KG}'_1 \leftarrow \mathsf{KG}_1$, $\mathsf{KG}'_2 \leftarrow \mathsf{KG}_2$;
for λ *dans* Lks *et chaque paire d'instances* $a, b \in \mathsf{KG}_i, i \in \{1, 2\}$ *satisfaisant la condition de* λ **do**
  ajouter x owl : sameAs y à KG'_i, où $x \in a^+$ et $y \in b^+$;
end
for $y \approx x \in \mathsf{KG}_i, i \in 1, 2$ *tel que* $\Sigma \cap \mathsf{KG}_i = \emptyset, \Sigma \backslash \mathsf{KG}_i \neq \emptyset$, *où* $\Sigma \in C(x), C(y), R(z, x), R(z, y)$ *pour un concept* C *ou un individu* z *et un rôle* R **do**
  ajouter Σ à KG'_i;
end
return $\mathsf{KG}'_1, \mathsf{KG}'_2$;

5.1 Implémentions

DICAP [7] est un logiciel open-source écrit en Java. L'architecture de DICAP est divisée en 5 modules complémentaires comme indiqué sur la Figure 1.

Le premier module est le module principal qui implémente l'algorithme 1, il utilise l'API OWL [14] pour analyser les graphes de connaissances d'entrée. Ce module est chargé d'appeler **Linkex** et **CANARD**. Il utilise les modules de saturation et les modules d'analyse.

Les modules de saturation sont respectivement responsables de la saturation des clés de liage et de la saturation des correspondances. Le module de saturation des correspondances implémente l'algorithme 2 et le module de saturation par clés de liage implémente l'algorithme 3.

Ces modules utilisent à leur tour les modules d'analyse des clés de liage et des correspondances. Les modules d'analyse utilisent respectivement les API OWL et d'alignement [8] pour analyser les clés de liage et les alignements donnés par le module principal. L'API d'alignement utilise également l'API OWL.

5.2 Résultats et discussion

Nous avons choisi un sous-ensemble du jeu de données *Conférence* de la campagne OAEI [8]. Nous considérons la paire d'ontologies peuplées edas_100 et conference_100 qui présentent les caractéristiques indiquées dans le Tableau 1 :

Dans cette expérience, nous avons lancé l'Algorithme 1 des jeux de données edas_100 et conference_100 avec une valeur de confiance de 0.7. Les résultats sont résumés dans le Tableau 2. Dans cette expérience, nous avons atteint une

7. https ://github.com/dace-dl-anr/DICAP
8. https ://oaei.ontologymatching.org/

	edas_100	**conference_100**
Taille (MB)	21.3	33.5
Individus	10 0517	21334
Axiomes	36 1087	248831
Ass. de classe	76 993	107394
DataProp.	8988	9314
ObjectProp.	196594	94020
Prop. d'ann.	21509	16361

TABLE 1 – Caractéristiques du jeu de données.

situation stationnaire après 2 itérations. Dans l'état initial, nous avions parmi les clés de liage, la clé suivante :

$$(\{\langle \mathsf{edas{:}hasName}, \mathsf{conference{:}has\_a\_name}\rangle\} \text{ linkkey } \langle \mathsf{owl{:}Thing}, \mathsf{owl{:}Thing}\rangle)$$

Nous avions également 221 correspondances complexes et simples obtenues par canard.

La saturation de edas_100 et conference_100 avec les clés de liage initiales permet d'obtenir des relations owl:sameAs supplémentaires entre ces deux RDF KGs. En conséquence, le nombre de correspondances est passé de 221 à 341. La valeur moyenne de la confiance des correspondances obtenues a également augmenté de 0.831 à 0.878 (passant par 0,864 dans l'état intermédiaire) comme indiqué dans le Tableau 3.

Les correspondances ont été utilisées comme en-

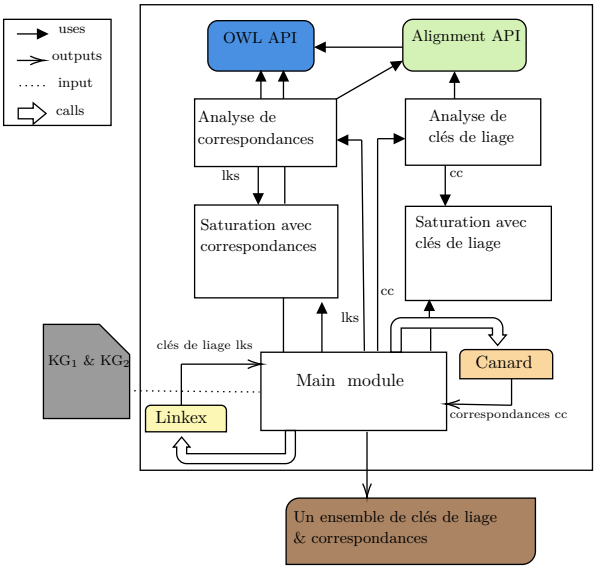

FIGURE 1 – L'architecture de **DICAP**.

	LksSEq	**LksCEq**	**Corr.**
État initial	0	0	221
État intermédiaire	23	18	291
État final	116	263	341

TABLE 2 – Le nombre de correspondances et de clés de liage entre classes simples et complexes équivalentes à différentes itérations.

trée pour **Linkex** afin d'obtenir les clés de liage suivantes :

$(\{\langle \text{rdfs:label}, \text{rdfs:label}\rangle\}$ linkkey
$\langle \text{edas:Review}, \text{conference:Review}\rangle)$

$(\{\langle \text{rdfs:label}, \text{rdfs:label}\rangle\}$ linkkey
$\langle \text{edas:Review}, \exists \text{conference:has\_authors.conference:Review}\rangle)$

Ces clés de liage possèdent une valeur hmean plus élevée que celles des clés originales, comme indiqué dans le Tableau 5.
Nous avons réalisé une autre expérience avec edas_100 et ekaw_100. Les résultats sont cohérents avec les résultats de l'expérience présentée.

5.3 Comparaison avec CANARD

CANARD suppose l'existence de relations owl:sameAs entre les entités des graphes de connaissances source et cible. Cependant, ce n'est pas toujours le cas. Pour cette raison, lorsque les liens d'identité ne sont pas disponibles, **CANARD** ajoute des relations synthétiques owl:sameAs entre les entités des graphes de connaissances source et cible s'ils partagent les mêmes valeurs pour la propriété rdfs:label. Cependant, cette condition est une condition faible pour imposer l'égalité entre les entités.
De façon à évaluer l'impact de l'exploitation des clé de liage sur **CANARD**, nous avons comparé premièrement le nombre de correspondances et la moyenne de leur valeur de confiance entre **DICAP**, **CANARD** (sans les égalités synthétiques, dans le Tableau 3 on y fait référence par **CANARD**$^-$) et **CANARD** (avec égalités synthétiques).
La saturation avec les clés de liage permet de générer les liens owl:sameAs entre les instances partageant les mêmes valeurs pour la propriété rdfs:label comme indiqué dans la première clé de liage de le Tableau 5a. Cette clé correspond en effet à ce que CANARD considère pour la génération de owl:sameAs dans le cas de leur absence. Les autres clés permettent de générer la propriété owl:sameAs entre les instances partageant des valeurs pour d'autres paires de propriétés présentes dans le reste des clés de liage, comme la paire de propriétés $\langle \text{hasFirstName}, \text{has\_the\_First\_Name}\rangle$ et $\langle \text{hasLastName}, \text{has\_the\_Last\_Name}\rangle$. Cela permet d'augmenter le nombre total d'entités liées par la propriété owl:sameAs, ce qui entraîne une augmentation du nombre de correspondances générées et donc une augmentation du rappel (ici, le *Couverture de CQA*) des correspondances (Tableau 4). Cependant, cette augmentation a un impact négatif sur la précision intrinsèque (Tableau 4). La couverture de CQA mesure à quel point un alignement permet de traduire un ensemble de requêtes SPARQL et la précision intrinsèque compare les instances des membres d'une correspondance. La précision intrinsèque équilibre la couverture de CQA de la manière dont la précision équilibre le rappel en recherche d'information.

5.4 Comparaison avec Linkex

Le Tableau 5 montre la hmean, la couverture et la discriminabilité des 3 principales clés de liage. Les clés de liage de la Tableau 5a sont obtenues par **Linkex** et les clés de liage du Tableau 5b sont obtenues par **DICAP**.
Les résultats représentés dans le Tableau 5 montrent que **DICAP** a pu produire de nouvelles clés de liage. Ces clés de liage possèdent une moyenne plus élevée que celle d'origine produite uniquement par **Linkex**. Les raisons derrière l'augmentation de la discriminabilité sont la présence de classes équivalentes, qui réduisent le domaine d'application de la clé de liage et per-

	DICAP	CANARD$^-$	CANARD
Nombre de corr.	341	206	221
Moyenne confidence	0.877	0.831	0.846

TABLE 3 – Comparaison entre **CANARD** et **DICAP**.

	DICAP	CANARD
Précision intrinsèque	0.076	0.109
Couverture de CQA	0.5	0.357

TABLE 4 – La précision et le couverture des correspondances générées par **DICAP** et **CANARD**.

mettent de générer des liens similaires à une correspondance 1 à 1. L'augmentation de la couverture est causée par la capacité de ces clés de liage à lier plus d'entités. Puisque la probabilité que les instances partagent des valeurs lorsque les entités appartiennent à des classes équivalentes est plus élevée.
Par exemple, le premier clé de liage dans le Tableau 5b

$$(\{\langle \text{rdfs:label}, \text{rdfs:label}\rangle)\} \text{ linkkey } \langle \text{edas:Review}, \text{conference:Review}\rangle) \quad (2)$$

permet de relier des entités des classes edas:Review et conference:Review si elles partagent les mêmes valeurs pour la propriété rdfs:label. Cela restreint le domaine d'application de la clé de liage en permettant de ne relier que des entités de ces classes, ce qui augmente la discriminabilité. Cela permet également de trouver plus d'instances satisfaisant la condition de cette clé de liage, car il est plus probable que les instances partagent des valeurs pour la propriété rdfs:label si elles appartiennent à des classes équivalentes. Ce n'est pas le cas pour la première clé de liage dans le Tableau 5a.

$$(\{\langle \text{rdfs:label}, \text{rdfs:label}\rangle)\} \text{ linkkey } \langle \text{owl:Thing}, \text{owl:Thing}\rangle)$$

qui relie des entités de n'importe quelle paire de classes, rendant ainsi sa correspondance loin d'être une correspondance de 1 à 1 et donc moins discriminante. La couverture de cette clé de liage est également plus faible que la clé de liage 2 car il est moins probable que les entités partagent des valeurs lorsqu'elles appartiennent à des classes non équivalentes.
Par conséquent, l'intégration de **CANARD** et **Linkex** dans **DICAP** a augmenté la qualité et le nombre de clés de liage obtenues par **Linkex** et a également augmenté le nombre de correspondances obtenues par **CANARD**.

6 Conclusion

Dans cet article, nous avons présenté **DICAP**, un algorithme combinant l'exploitation de stratégies d'interconnexion de données et d'alignement d'ontologies. Le but de cet algorithme est de tirer parti des résultats de chaque tâche pour améliorer les résultats de l'autre. Notre algorithme aborde la tâche d'interconnexion de données en utilisant des clés de liage extraites par **Linkex**. De plus, il utilise le système **CANARD** pour aborder le problème d'alignement d'ontologies en produisant des correspondances simples et complexes entre les graphes de connaissances considérés.
Nous avons mis en œuvre cet algorithme et effectué une expérience qui révèle l'importance de cet algorithme pour améliorer les résultats des systèmes considérés, c'est-à-dire **Linkex** et **CANARD**. Notamment, **DICAP** a augmenté le nombre et amélioré la discriminabilité et aussi la couverture des clés de liage générées par **Linkex**. **DICAP** a également augmenté le nombre de correspondances simples et complexes générées par **CANARD**, en augmentant en conséquence la couverture. Ainsi, notre hypothèse est valide.
Cependant, la précision de ces correspondances est inférieure à celle générée par **CANARD**. Afin d'améliorer cela, nous prévoyons d'enrichir les graphes de connaissances avec seulement des clés de liage entre des classes équivalentes. Cela permettra de créer des liens plus précis, conduisant à la génération de correspondances complexes plus précises.
Nous prévoyons également d'étendre ce travail dans plusieurs directions. Parmi elles, nous prévoyons tout d'abord de prendre en compte de nouvelles classes complexes qui n'ont pas été prises en compte dans la version actuelle, telles que celles formées à l'aide du constructeur de rôle inverse. Cela permet de saturer les graphes de connaissances avec de nouvelles assertions de classes complexes, ce qui permet d'extraire des clés de liage entre ces classes.
Nous aimerions également réaliser un ensemble d'expériences avec des jeux de données réels dé-

clé de liage	hmean	discriminabilité	couverture
{(⟨rdfs :label,rdfs :label⟩)} ⟨owl :Thing,owl :Thing⟩	0.479	0.681	0.37
{⟨hasFirstName,has_the_First_Name⟩, ⟨hasLastName,has_the_Last_Name⟩} ⟨owl :Thing,owl :Thing⟩	0.173	0.999	0.094
{⟨hasLastName,has_the_Last_Name⟩} ⟨owl :Thing,owl :Thing⟩	0.143	0.293	0.094

(a) Clés de liage obtenues par **Linkex**

clé de liage	hmean	discriminabilité	couverture
{(⟨rdfs :label,rdfs :label⟩)} ⟨edas :Review,conference :Review⟩	0.997	0.995	1
{(⟨rdfs :label,conference :has_a_name⟩)} ⟨edas :Workshop,conference :Workshop⟩	0.667	1	0.5
{(⟨rdfs :label,rdfs :label⟩)} ⟨edas :Review,∃ conference :has_authors.conference :Review⟩	0.567	0.995	0.397

(b) Clés de liage obtenues par **DICAP**

TABLE 5 – Comparaison entre les trois premiers clés de liage (selon hmean) obtenues uniquement par **Linkex** et les clés de liage obtenues par **DICAP**.

crits avec des ontologies riches telles que dbpedia et wikipedia.

Remerciements

Ce travail a été partiellement financé par le projet DACE-DL, ANR.

Références

[1] Nacira Abbas, Jérôme David, and Amedeo Napoli. Discovery of Link Keys in RDF Data Based on Pattern Structures : Preliminary Steps. In *CLA 2020 - The 15th International Conference on Concept Lattices and Their Applications*, Proceedings of the 15th International Conference on Concept Lattices and Their Applications, Tallinn / Virtual, Estonia, June 2020.

[2] Manel Achichi, Mohamed Ben Ellefi, Danai Symeonidou, and Konstantin Todorov. Automatic Key Selection for Data Linking. In *EKAW : Knowledge Engineering and Knowledge Management*, volume LNCS of *Knowledge Engineering and Knowledge Management*, pages 3–18, Bologne, Italy, November 2016. Springer International Publishing.

[3] Mustafa Al-Bakri, Manuel Atencia, Jérôme David, Steffen Lalande, and Marie-Christine Rousset. Uncertainty-sensitive reasoning for inferring sameas facts in linked data. In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen, editors, *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 698–706. IOS Press, 2016.

[4] Manuel Atencia, Jérôme David, and Jérôme Euzenat. Data interlinking through robust linkkey extraction. In *Proceedings of the Twenty-First European Conference on Artificial Intelligence*, ECAI'14, page 15–20, NLD, 2014. IOS Press.

[5] Manuel Atencia, Jérôme David, Jérôme Euzenat, Amedeo Napoli, and Jérémy Vizzini. Link key candidate extraction with relational concept analysis. *Discrete Applied Mathematics*, 273 :2–20, 2020.

[6] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, Cambridge, UK, 2 edition, 2007.

[7] Dan Brickley, Matthew Burgess, and Natasha Noy. Google dataset search : Building a search engine for datasets in an open web ecosystem. In *The World Wide Web Conference*, WWW '19, page 1365–1375, New York, NY, USA, 2019. Association for Computing Machinery.

[8] Jérôme David, Jérôme Euzenat, François Scharffe, and Cássia Trojahn dos Santos. The alignment api 4.0. *Semant. Web*, 2(1) :3–10, jan 2011.

[9] Houssameddine Farah, Danai Symeonidou, and Konstantin Todorov. Keyranker : Automatic rdf key ranking for data linking. In *Proceedings of the Knowledge Capture Conference*, K-CAP 2017, New York, NY, USA, 2017. Association for Computing Machinery.

[10] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F. Cruz, and Francisco M. Couto. The agreementmakerlight ontology matching system. In *OTM Conferences*, 2013.

[11] Daniel Faria, Catia Pesquita, Teemu Tervo, Francisco M. Couto, and Isabel F. Cruz. Aml and amlc results for oaei 2019. In *OM@ISWC*, 2019.

[12] Norbert Fuhr. Probabilistic datalog : Implementing logical information retrieval for advanced applications. *J. Am. Soc. Inf. Sci.*, 51 :95–110, 2000.

[13] Kin Wah Fung and Junchuan Xu. Synergism between the mapping projects from snomed ct to icd-10 and icd-10-cm. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2012 :218–27, 2012.

[14] Matthew Horridge and Sean Bechhofer. The owl api : A java api for working with owl 2 ontologies. In *Proceedings of the 6th International Conference on OWL : Experiences and Directions - Volume 529*, OWLED'09, page 49–58, Aachen, DEU, 2009. CEUR-WS.org.

[15] Arun Krishnan. Making search easier : How amazon's product graph is helping customers find products more easily. In *Amazon Blog*, 08 2018.

[16] Axel-Cyrille Ngonga Ngomo and Sören 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, page 2312–2317. AAAI Press, 2011.

[17] Lorena Otero-Cerdeira, Francisco J. Rodríguez-Martínez, and Alma Gómez-Rodríguez. Ontology matching : A literature review. *Expert Systems with Applications*, 42(2) :949–971, 2015.

[18] Joe Raad, Erman Acar, and Stefan Schlobach. On the impact of sameas on schema matching. In *Proceedings of the 10th International Conference on Knowledge Capture*, K-CAP '19, page 77–84, New York, NY, USA, 2019. Association for Computing Machinery.

[19] Marta Sabou, Elodie Thiéblin, Ollivier Haemmerlé, Nathalie Hernandez, and Cassia Trojahn. Survey on complex ontology matching. *Semant. Web*, 11(4) :689–727, jan 2020.

[20] Fatiha Saïs, Nathalie Pernelle, and Marie-Christine Rousset. L2R : A Logical Method for Reference Reconciliation. In *Twenty-Second AAAI Conference on Artificial Intelligence*, page 2007, Vancouver, British Columbia, Canada, July 2007.

[21] Dezhao Song and Jeff Heflin. Automatically generating data linkages using a domain-independent candidate selection approach. In *Proceedings of the 10th International Conference on The Semantic Web - Volume Part I*, ISWC'11, page 649–664, Berlin, Heidelberg, 2011. Springer-Verlag.

[22] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. Paris : Probabilistic alignment of relations, instances, and schema. 2011.

[23] Élodie Thiéblin, Ollivier Haemmerlé, and Cássia Trojahn. CANARD complex matching system : results of the 2018 OAEI evaluation campaign. In *OM@ISWC*, pages 138–143, 2018.

[24] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk - a link discovery framework for the web of data. In *LDOW*, 2009.