Formal Concept Analysis for Discovering Link Keys in the Web of Data

Nacira Abbas

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Formal Concept Analysis for Discovering Link Keys in the Web of Data

THÈSE

présentée et soutenue publiquement le 13 octobre 2023

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Nacira Abbas

Composition du jury

Président :	Sébastien Ferré	Professeur, Université de Rennes
Rapporteurs :	Fayçal Hamdi	Maître de conférences HDR, Conservatoire National des Arts et Métier
	Fatiha Saïs	Professeure, Université Paris-Saclay
Examinateurs :	Mathieu d'Aquin	Professeur, Université de Lorraine
	Luis Galárraga	Chercheur, INRIA IRISA
	Cassia Trojahn	Maître de Conférences HDR, Université Toulouse - Jean Jaurès
Directeurs :	Jérôme David	Maître de Conférences HDR, Université de Grenoble Alpes
	Amedeo Napoli	Directeur de recherche émérite, CNRS

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Remerciements

This dissertation marks the end of a journey where I learned a lot about science and its beauty. I met brilliant scientists from around the world, traveled to places I never thought I would be. It was also the first time that I moved to another country and lived by myself, and, more importantly, it was a journey of self-discovery.

The completion of this journey would not be possible without a number of people that I consider myself fortunate to have encountered.

I would like to express my gratitude to the thesis reporters, Fayçal Hamdi and Fatiha Saïs, as well as all the members of the thesis committee—Mathieu d'Aquin, Sébastien Ferré, Luis Galárraga, and Cassia Trojahn—who kindly agreed to review my thesis. Thank you for your meticulous work, valuable comments on my research, and engaging discussions during the thesis defense.

This thesis would not have been possible without the guidance of my thesis supervisors, Jérôme David and Amedeo Napoli. I would like to express my gratitude for the immense dedication you have shown, the high standards that encouraged me to strive for excellence. With Amedeo, I witnessed an example of extreme passion for science, and with Jérôme, I witnessed how kindness and intelligence can be embodied within a scientist. Jérôme and Amedeo, I can never thank you enough for this amazing experience and for opening many doors for me.

This thesis was conducted within the Orpailleur team. I express my gratitude to the team leader, Miguel Couceiro; your vibrant smile truly made a difference. Special appreciation goes to Alexandre Bazin, and Justine Reynaud, both of whom I consider mentors. Thank you for patiently addressing all my questions and providing support throughout and beyond the thesis. To all my LORIA colleagues, you have become family and friends: my "co-bureau" Laurine, Guilherme, Tanya, Esteban, Egor, Adrien, and all the Orpailleurs—Laura, Line, Nyoman, Georgios, Clair, Ambroise, Simon, Adrien, and Yannick. Special mentions to my LORIA friends: Athénaïs, Noémie, Omar, and all my colleagues from the Faculty of Sciences and Technologies, particularly Guilhem, for their invaluable help and support in teaching matters.

I want to express my gratitude to ELKER project members for all the discussions we had, and to the members of the mOeX team in Grenoble for taking care of me, especially during the first lockdown. Special thanks to Esther Galbrun and Pauli Miettinen for the warm welcome in Kuopio, Finland. I carry with me many memories of the breathtaking nature of Kuopio and a lot of re-descriptions.

To my professors at the University of Mouloud MAMMERI, Tizi Ouzou, thank you for planting the seeds of passion for computer science within me. I am also deeply thankful to a high school teacher, Monsieur Belhareth, whom I met only once, and whose advice completely changed my life—thank you. To all my teachers in my small Kabyle village in North Africa, they say it takes a village to raise a child, and it is undoubtedly true.

I would like to express my gratitude to my uncle, his wife, and their daughter Nelya for opening their home and their hearts to me. I always feel at home when visiting you in Paris during my thesis time.

Finally, my sincere thanks to every person I didn't mention here but who had a positive impact on this work.

To my parents, for the unconditional love. Two particular teachings from you were precious for me to accomplish this work: Vava, you taught me to ask myself at the end of each day, "What did I learn today?" Yemma, you taught me that if I can't reach my goal because there is a wall, then I have to break that wall.

To my three sisters, your unwavering support has been invaluable, and I deeply cherish it.

Contents

Résumé étendu **1**

Chapter 1
Introduction

1.1 Link Keys 9
1.2 Challenges in Link Key Discovery 10
1.3 Contributions and Outline of the Thesis 11

Chapter 2
The Web of Data

2.1 Knowledge Representation in the Web of Data 14
2.2 The Web of Data Standards 17

Chapter 3
Formal Concept Analysis

3.1 Basic Notions of Formal Concept Analysis 21
3.2 Pattern Structures 24

Chapter 4
Data interlinking

4.1 Identity 28
4.2 Data Interlinking Approaches 29
4.2.1 Embedding-based Approaches 30
4.2.2 Data Interlinking Frameworks 31
4.2.3 Key-based Approaches 32
4.3 Link Keys 34
4.3.1 Syntax 34
4.3.2 Semantics 37
4.3.3 Link Key Discovery 38

4.4 Discussion 44

Chapter 5
Pattern Structures for Link Key Candidate Discovery

5.1 Motivation 46
5.2 Extending the Definition of Link Key Candidates 48
5.3 A Pattern structure for link Key discovery 50
5.4 Conclusion and Discussion 52

Chapter 6
Selection of Relevant Link Key Candidates

6.1 Motivation 57
6.2 Link Key Selection Guided by the Classes 58
6.3 Link Key Selection Guided by the Lattice 64
6.4 The Sandwich algorithm at work 66
6.5 Conclusion and Discussion 69

Chapter 7
Investigating Redundancy of Link Key Candidates

7.1 Motivation 73
7.2 Partition-based Redundancy 73
7.2.1 Partition Pattern Structures for non Redundant Candidate Discovery . . 73
7.2.2 Partition-based Quality Measures 75
7.2.3 Experiments 79
7.3 Similarity-based Redundancy 83
7.3.1 Building an Approximate Basis of Link Keys 85
7.3.2 Experiments 88
7.4 Conclusion and Discussion 91

Chapter 8
Conclusion and perspectives

8.1 Summary of Contributions 95
8.2 Perspectives 96

Appendix A
List of Publications

Bibliography **101**

Résumé étendu

La notion de lien remonte aux premières idées du Web. En effet, dans son célèbre article intitulé "As We May Think" [23], Vannevar Bush s'est adressé à la communauté scientifique, exprimant des préoccupations concernant le volume écrasant de publications que les scientifiques devaient lire pour se tenir au courant des connaissances dans leur domaine. Il envisageait un moment où il deviendrait impossible de suivre la croissance exponentielle de ces ressources scientifiques. Pour résoudre ce problème, il a proposé un système appelé le MEMEX, qui permettrait aux individus de lire et de consulter des documents tout en établissant des liens entre eux et en notant les connexions associatives qu'ils découvraient. L'idée était que si une personne revenait ultérieurement à l'un des documents, le système lui rappellerait automatiquement l'autre document lié. Dans son article, Bush a discuté de l'importance des liens dans le fonctionnement de l'esprit humain, affirmant que "*l'esprit fonctionne par association. Avec un élément en main, il passe instantanément au suivant suggéré par l'association des pensées, conformément à une toile complexe de pistes transportées par les cellules du cerveau*". Il a souligné que le processus de liage des éléments était d'une importance capitale dans son système, "*le processus de liage de deux éléments est la chose importante*". Vingt ans plus tard, Ted Nelson [67] a inventé le terme "hypertexte" pour décrire ces documents liés, "*Laissez-moi introduire le mot "hypertexte" pour désigner un ensemble de documents écrits ou picturaux interconnectés d'une manière si complexe qu'il ne pourrait pas être présenté ou représenté commodément sur papier*". Tim Berners-Lee [19], vingt ans plus tard, a inventé le World Wide Web, où l'hypertexte n'était plus confiné à une seule machine. Il a imaginé que les documents devraient être distribués à travers le réseau, permettant à un document de référencer un autre document résidant sur une machine différente grâce à des liens entre eux [43].

Berners-Lee a attiré l'attention sur le fait que le Web existant manque de sémantique suffisante pour que les machines puissent l'utiliser efficacement. En réponse à ce défi et dans le but de rendre le Web plus accessible et compréhensible pour les machines, il a introduit le "Web des données". Ce Web peut être considéré comme une couche supplémentaire connectée au Web des documents [51]. Cet espace de données englobe un large éventail de domaines, y compris les données scientifiques et gouvernementales, les emplacements géographiques, les personnes, les entreprises et la musique. Tout comme les hyperliens dans le Web traditionnel relient les documents pour former un espace d'information global, dans le Web des données, les entités sont interconnectées via des liens pour établir un espace de données global. Le World Wide Web Consortium (W3C) [1] a adopté le *Resource Description Framework* (RDF) [26] comme modèle de représentation des données sur le Web. Son composant central est un triplet RDF (s, p, o) où s, p et o sont appelés le sujet, le prédicat (ou propriété) et l'objet (ou valeur). Dans la Figure 1.2, (a,`designation,"The Talisman"`) est un triplet RDF, où a est un sujet lié à

[1] Une communauté internationale collaborative engagée dans l'établissement de normes ouvertes favorisant la croissance du Web. https://www.w3.org

la valeur `"The Talisman"` par la propriété `designation`. Pour spécifier qu'une entité est une instance d'une classe donnée, RDF est équipé du prédicat rdf:type. Dans la Figure 1.2, le triplet (a,rdf:type,`Book`) indique que le sujet a est une instance de la classe `Book`.

En publiant leurs données sur le Web, les organisations et les individus contribuent à la construction du Web de données. Tim Berners-Lee [17] a introduit un ensemble de directives, appelées "principes des données liées", qui énoncent quatre règles pour la publication et le liage de données structurées sur le Web. La quatrième règle est la suivante : *"Inclure des liens vers d'autres URI[2], afin qu'ils puissent découvrir plus de choses"*.

Les données liées ouvertes (Linked Open Data - LOD) désignent des données liées publiées sous une licence ouverte, permettant leur réutilisation. Berners-Lee [17] propose un système de notation en étoiles pour les fournisseurs de données afin de déterminer si leurs ensembles de données répondent aux exigences du LOD. Le cinquième élément de ce système d'évaluation est le suivant : *"Lier vos données à celles d'autres personnes pour fournir un contexte"*.

Il devient clair que les liens jouent le rôle de système nerveux du Web des données. Plus particulièrement, les liens d'identité qui associent deux individus représentant la même entité. Ces liens sont représentés sous forme de triplets RDF dans la forme (a, owl:sameAs, b), indiquant que a et b font référence à la même entité, comme dans l'exemple de la Figure 1.2. Le processus d'identification de tels liens est appelé *liage de de données*. Établir ces liens entre les entités dans le Web des données est crucial pour assurer l'interopérabilité et faciliter la collaboration entre les différents systèmes. Cependant, étant donné la quantité considérable de données disponibles, le liage manuel des données est difficile à mettre en œuvre. Par conséquent, différentes approches ont été développées pour automatiser ce processus [68]. Dans cette thèse, nous sommes intéressés par une approche qui repose sur des *clés de liage* [7].

Les clés de liage peuvent être considérées comme une généralisation des clés entre ensembles de données. Elles offrent un moyen explicite et non ambigu pour lier les individus. Les clés de liage peuvent être particulièrement avantageuses car elles fonctionnent comme des règles, déclarant directement si les nouveaux individus ajoutés aux ensembles de données sont identiques. De plus, l'un des points forts des clés de liage réside dans leur utilisation en tant qu'axiomes logiques dans les ontologies, ce qui implique l'exploitation de la sémantique. Un autre aspect notable des clés de liage est leur fléxibilité. Elles sont applicables dans des scénarios impliquant différentes ontologies ou lorsque aucune alignement d'ontologies n'est disponible.

Clés de liage

Introduites pour la première fois par Atencia et al. [7], une clé de liage prend la forme de deux ensembles de paires de propriétés associées à une paire de classes. Par exemple, soit

$$k = (\{(\texttt{designation,title})\}, \{(\texttt{designation,title}), (\texttt{creator,author})\}, (\texttt{Book,Novel}))$$

Si k est une clé de liage sur deux ensembles de données, cela signifie que[3] si une instance a de la classe `Book` et b de la classe `Novel` (condition de classes) partagent au moins une valeur pour les propriétés `creator` et `author` ($\exists$-conditions) et que a et b ont les mêmes valeurs pour les propriétés `designation` et `title` ($\forall$-conditions), alors a et b désignent la même entité. Ainsi, (a, owl:sameAs, b) est un lien d'identité sur les deux ensembles de données. On dit que k génère le lien d'identité (a, b).

[2]Uniform Resource Identifier.

[3]Généralement, la lecture se fait de gauche à droite de k, mais l'ordre n'a pas d'importance.

Dans RDF, une instance peut avoir plusieurs valeurs pour une seule propriété. Par conséquent, les valeurs des propriétés sont comparées de différentes manières : (i) déterminer si les instances partagent au moins une valeur pour une paire de propriétés, on parle des ∃-conditions, et (ii) considérer si les instances ont les mêmes valeurs pour une paire de propriétés, on parle des ∀-conditions.

Dans la Figure 1.2, la clé de liage k génère le lien d'identité (a, b) car :

1. a est une instance de la classe `Book` et b est une instance de la classe `Novel` (condition de classes), et
2. a et b partagent la valeur `"Stephen King"` pour les propriétés `creator` et `author` (∃-conditions), et
3. a et b ont les mêmes valeurs pour les propriétés `designation` et `title` qui est `"The Talisman"` (∀-conditions).

Généralement, les clés de liage ne sont pas fournies avec les jeux de données. Une méthode naïve pour les découvrir consiste à explorer toutes les combinaisons possibles de propriétés et de classes, appelées *expressions de clés de liage*. Ensuite, il convient de vérifier la validité de chaque expression en tant que clé de liage, en s'assurant qu'elle génère effectivement des liens d'identité et non pas des liens arbitraires. Toutefois, en raison du grand nombre d'expressions disponibles, et afin de réduire l'espace de recherche, les algorithmes de découverte de clés de liage [7, 10] se focalisent sur l'identification d'un sous-ensemble de ces expressions, désigné sous le terme de *clés de liage candidates*. Ces candidates sont ensuite évalués à l'aide de mesures spécifiques pour déterminer les plus pertinentes qui peuvent être déclarés comme clés de liage valides. La Figure 1.3 représente les différentes étapes du processus de découverte de clés de liage.

Une relation directe a été établie dans [8] entre la découverte de clés de liage et l'Analyse Formelle de Concepts (AFC [46]), car la définition d'une clé de liage candidate correspond à la définition d'un concept formel découvert par l'AFC. L'Analyse Formelle de Concepts, un cadre mathématique d'analyse de données développé dans les années 1980, permet d'identifier des concepts formels qui regroupent des objets (constituant l'*extent* du concept formel), en fonction de leurs attributs communs (constituant l'*intent* du concept formel). Ces concepts sont organisés dans une structure hiérarchique appelée treillis de concepts formels. Atencia et al. [8, 10] ont proposé d'utiliser l'AFC pour découvrir des clés de liage candidates. A partir de deux ensembles de données RDF, cette méthode identifie des concepts formels, les clés de liage candidates étant leurs intents et l'ensemble de liens générés par ces candidats étant leurs extents.

Questions de recherche associées à la découverte des clés de liage

Cette thèse s'intéresse à la découverte des clés de liage à partir des ensembles de données. Dans ce qui suit, nous identifions les défis et les questions de recherche associés à la découverte des clés de liage.

La nécessité des paires de classes explicites dans les clés de liage

Les approches existantes [7, 10] pour la découverte de clés de liage ne spécifient pas explicitement les paires de classes associées aux clés générées. L'absence de ces paires de classes représente un défi lorsqu'il s'agit d'évaluer avec précision les clés de liage candidates. En effet, ces clés peuvent

être pertinentes pour une paire de classes (par exemple, (`Livre`,`Roman`)) mais pas pour une autre (par exemple, (`Scientifique`,`Personne`)). De plus, les algorithmes de raisonnement [54] basés sur les clés de liage nécessitent une spécification explicite des paires de classes pour être plus efficaces.

De plus, le Web des données permet des perspectives diverses sur les entités, ce qui conduit à différents niveaux d'abstraction. Par exemple, dans un ensemble de données, un individu appartient aux classes `Femme` et `Scientifique`, tandis que dans l'autre ensemble de données, la même entité appartient à une seule classe `FemmeScientifique`. Cependant, les méthodes existantes de découverte de clés de liage ne tiennent pas compte de cette différence de niveaux d'abstraction.

Question de recherche 1 : *Comment peut-on étendre la méthode basée sur l'AFC pour découvrir des clés de liage candidates tout en identifiant explicitement leurs paires de classes et en tenant compte des niveaux d'abstraction variables entre les ensembles de données ?*

L'efficacité de la méthode de sélection existante est limitée

Une fois l'ensemble des clés de liage candidates identifié, l'étape suivante consiste à sélectionner les clés de liage valides parmi cet ensemble. Pour déterminer la validité d'une clé de liage candidate, les algorithmes utilisent une stratégie de sélection des meilleurs clés de liage candidates basée sur des mesures de qualité définies, comme décrit dans [7]. Cependant, si cette stratégie se révèle efficace lorsqu'il s'agit uniquement d'une seule paire de classes, elle devient moins efficace lors de l'évaluation clé de liage candidates sur des ensembles de données avec plusieurs classes. Dans de tels cas, même si une candidate est une clé de liage pertinente, elle peut obtenir un faible classement selon ces mesures de qualité. De plus, les candidates hautement classés peuvent générer des liens uniquement pour des classes spécifiques tout en échouant à générer des liens pour d'autres classes.

Question de recherche 2 : *Comment garantir une sélection efficace des clés de liage candidates dans des ensembles de données contenant plusieurs classes ?*

L'espace de recherche des clés de liage reste important

Comme mentionné précédemment, au lieu d'explorer toutes les expressions possibles de clés de liage, les méthodes existantes réduisent l'espace de recherche à un sous-ensemble de ces expressions, appelées clés de liage candidates. En effet, un ensemble de liens peut être généré par plusieurs expressions de clés de liage, et celles qui génèrent le même ensemble de liens sont considérées comme redondantes. Pour réduire cette redondance, toutes les expressions de clés de liage qui génèrent le même ensemble de liens sont représentées par une seule expression maximale, appelée clé de liage candidate. La méthode basée sur l'Analyse Formelle de Concepts [10] garantit que chaque clé de liage candidate génère un ensemble unique de liens. Cela découle du fait que chaque intent d'un concept formel (clé de liage candidate) correspond à un extent distinct (ensemble de liens) et vice versa. Cependant, malgré cette réduction, l'ensemble de candidates résultantes reste important en raison du volume considérable de données sur le Web. Par conséquent, cela présente des défis pour la visualisation, l'évaluation et l'analyse des clés de liage candidates.

Question de recherche 3 : *Est-ce que l'ensemble de clés de liage candidates présente encore une forme de redondance qui peut être identifiée et réduite afin de faciliter la visualisation, l'analyse et l'évaluation des clés de liage candidates ?*

Contributions et plan de la thèse

Ce manuscrit est organisé de la manière suivante.

Chapitre 2

Dans ce chapitre, nous passons en revue les concepts fondamentaux associés au Web des données, fournissant ainsi une base nécessaire pour comprendre les sections suivantes de cette thèse. Nous examinons la Représentation des Connaissances, qui établit les principes fondamentaux et les formalismes pour structurer et organiser les connaissances dans le Web des données. De plus, nous présentons les normes établies par le W3C qui sont basées sur les formalismes de représentation des connaissances mentionnés précédemment.

Chapitre 3

Ce chapitre présente un aperçu de l'Analyse Formelle de Concepts et de ses extensions, en particulier des *Pattern Structures* [44] et des *Partition Pattern Structures* [15]. Ces extensions constituent la base des principales contributions de cette thèse. Les Pattern Structures sont une extension de l'Analyse Formelle de Concepts visant à adapter le formalisme de l'AFC à des objets dont la description est trop complexe pour être exprimée comme un ensemble d'attributs binaires. C'est le cas lorsque les objets ont des attributs qui peuvent être des nombres, des intervalles, des séquences, des arbres, des graphes, etc. Lorsque les descriptions dans une Pattern Structure sont des partitions, on parle de Partition Pattern Structure.

Chapitre 4

Ce chapitre introduit la tâche de liage des données, qui consiste à identifier et à relier des instances identiques à travers des ensembles de données. Pour contextualiser notre travail, nous explorons différents cadres et approches pour le liage des données, en mettant particulièrement l'accent sur les clés de liage. Tout au long du chapitre, nous fournissons des définitions et des algorithmes étroitement liés aux clés de liage.

Chapitre 5

Ce chapitre aborde la **question de recherche 1**. Notre première contribution présente une nouvelle Pattern Structure qui améliore considérablement le processus de découverte des clés de liage. Tout d'abord, elle permet la découverte de clés de liage candidates tout en spécifiant les paires de classes associées. Cela permet une évaluation plus précise des candidates et améliore l'efficacité du raisonnement. Ensuite, nous avons étendu la définition des clés de liage candidates au-delà d'une paire de classes atomiques pour inclure des paires de conjonctions et de disjonctions de classes. Cette extension permet de tenir compte des différents niveaux d'abstraction entre les ensembles de données. Ce chapitre est basé sur l'article publié dans *the 15th International Conference on Concept Lattices and Their Applications* [5].

Chapitre 6

Ce chapitre aborde la **question de recherche 2**. Ici, notre objectif est de sélectionner les clés de liage candidates pertinentes en se basant sur la Pattern Structures proposée dans le Chapitre 5. Pour cela, nous avons introduit deux méthodes. La première méthode est basée sur l'algorithme

LKSA qui est guidé par les paires de classes associées aux clés de liage candidates. Cette approche sélectionne les candidates qui génèrent des liens d'identité entre les classes pertinentes, tout en éliminant les candidates associés à des classes non pertinentes. La deuxième méthode, basée sur l'algorithme SANDWICH, est guidée par le treillis des clés de liage candidates. Cette approche utilise une stratégie d'élagage qui combine des approches ascendantes et descendantes pour sélectionner les candidates situées au "milieu" du treillis. Ce chapitre est basé sur deux articles. Le premier a été publié dans *les 21èmes Journées Francophones Extraction et Gestion des Connaissances* [6]. Le deuxième a été publié dans *the 16th International Conference on Formal Concept Analysis* [2].

Chapitre 7

Ce chapitre aborde la **question de recherche 3**. Dans ce chapitre, nous abordons le problème de la redondance parmi les clés de liage candidates, dans le but d'obtenir un ensemble plus concis de candidates qui simplifie les processus d'analyse, d'évaluation et de sélection. Pour celà, nous examinons d'abord la partition induite par la relation owl:sameAs parmi l'ensemble de liens de chaque clé de liage candidate. Les candidates qui génèrent la même partition sont considérées comme redondantes. Par conséquent, nous proposons une Partition Pattern Structure pour détecter et fusionner les candidates redondantes en fonction de l'égalité de leurs partitions. Cependant, nos expérimentations sur des ensembles de données réels révèlent que la redondance basée sur les partitions est rare ou inexistante. Nous proposons donc une approche alternative qui considère comme redondantes les candidates générant des ensembles similaires de liens. Pour identifier et réduire cette redondance, nous développons une approche utilisant le clustering hiérarchique. Dans cette approche, les candidates produisant des ensembles similaires de liens sont regroupés ensemble en clusters, et nous sélectionnons un représentant de chaque cluster. Cela permet de réduire efficacement la redondance parmi les clés de liage candidates. Ce chapitre est basé sur trois articles. Le premier a été publié dans *the 9th International Workshop "What can AFC do for Artificial Intelligence?"* [1]. Le deuxième a été publié dans *the 16th International Conference on Concept Lattices and Their Applications* [3]. Le troisième article a été publié dans l'*International Journal of Approximate Reasoning* [4].

Chapitre 8

Ce chapitre conclut ce manuscrit en résumant nos contributions et en présentant quelques perspectives de recherche. La liste de toutes nos publications est disponible dans l'annexe A.

Le travail présenté dans cette thèse a été réalisé dans le cadre du projet ELKER[4] – Enhancing Link Keys: Extraction and Reasoning ANR 2017 Project (ANR-17-CE23-0007-01), financé par l'Agence Nationale de la Recherche (ANR) en France. L'objectif de ce projet est d'étendre les bases et les algorithmes de clés de liage de deux manières complémentaires, en découvrant d'abord automatiquement des clés de liage à partir d'ensembles de données, puis en raisonnant avec ces clés.

[4]https://project.inria.fr/elker/

Chapter 1

Introduction

Contents

1.1 Link Keys **9**
1.2 Challenges in Link Key Discovery **10**
1.3 Contributions and Outline of the Thesis **11**

The significance of links can be traced back to the first ideas of the Web. Indeed, in his renowned article "As We May Think" [23], Vannevar Bush addressed the scientific community, expressing concerns about the overwhelming volume of books that scientists needed to read in order to stay updated with knowledge in their field. He foresaw a time when keeping up with the exponential growth of books would become impossible. To address this issue, he proposed a system called MEMEX, which would enable individuals to read and consult documents while establishing links between them and noting the associative connections they discovered. The idea was that if a person returned to one of the documents at a later time, the system would automatically remind them of the other linked document. In his article, Bush discussed the importance of links in how the human mind operates, stating that "*it operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain*". He emphasized that the process of linking items together was of utmost importance in his system, "*The process of tying two items together is the important thing*". Twenty years later, Ted Nelson [67] coined the term "hypertext" to describe this linked documents, "*Let me introduce the word "hypertext" to mean a body of written or pictorial material interconnected in such a complex way that it could not conveniently be presented or represented on paper*". Tim Berners-Lee [19], another twenty years later, invented the World Wide Web, where hypertext was no longer confined to a single machine. He envisioned that documents should be distributed across the network, allowing a document to reference another document residing on a different machine through links between them [43].

Berners-Lee brought attention to the fact that the existing Web lacks sufficient meaning for machines to effectively utilize. In response to this challenge and with the aim of making the Web more accessible and understandable for machines, he introduced the "Web of data". This Web can be viewed as an additional layer connected to the Web of documents [51]. Similar to how hyperlinks in the traditional Web connect documents to form a single global information space, in the Web of data, entities are interconnected via links to establish a global data space. Moreover, the Web of data enables explicit representation of the nature of links between entities, instead of simply stating that there is some kind of connection, as is the case in the Web of documents.

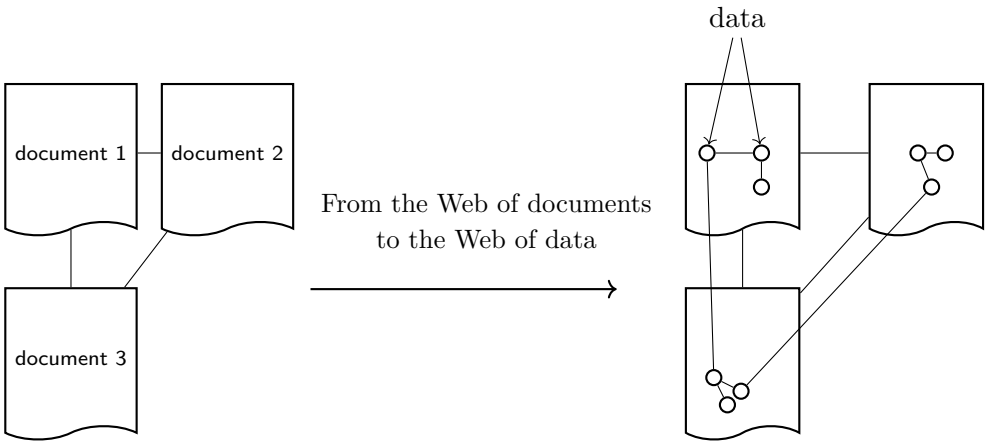


Figure 1.1: An illustration of the Web of documents, on the left, connected through hyperlinks. On the right, the Web of data with data connected via typed links.

Figure 1.1 is an illustration of how the Web of data is related to the Web of documents.

The World Wide Web Consortium (W3C)[5], adopted the *Resource Description Framework* (RDF) [26] as a framework for representing data in the Web. The core component of this framework is the RDF statement which is a triple (s, p, o) where s, p and o are called the subject, predicate (or property) and object of the statement. In Figure 1.2, (a,`designation,"The Talisman"`) is an RDF statement, or triple, where a is a subject related to the object, or the value, `"The Talisman"` by the property `designation`. To specify that an entity is an instance of a given class, RDF is equipped with the predicate rdf:type. In Figure 1.2, the triple (a,rdf:type,`Book`) states that the subject a is an instance of the class `Book`.

By publishing their data on the Web, organizations and individuals contribute to the construction of the Web of data. Tim Berners-Lee [17] introduced a set of guidelines, referred to as the "Linked Data principles", which outline four rules for publishing and interlinking structured data on the Web. The fourth rule is: *"Include links to other URIs*[6]*, so that they can discover more things"*.

Linked Open Data (LOD) refers to linked data that is released under an open license, which allows its reuse. Berners-Lee [17] provides a star rating system for data providers to determine whether their data sets meet the requirements of the LOD. The fifth element of this evaluation system is: *"Link your data to other people's data to provide context"*.

It becomes clear that links serve as the nervous system of the Web of data. More particularly, identity links play a significant role by connecting two individuals that represent the same entity. These links are represented as RDF triples in the form of $(a, \text{owl:sameAs}, b)$, indicating that a and b refer to the same entity, as in the example in Figure 1.2. The process of identifying such links is referred to as *data interlinking*. Establishing these links between entities in the Web of data is crucial to ensure interoperability and facilitate collaboration among different systems. However, given the vast amount of data available, manual data interlinking is impractical. As a result, various approaches have been developed to automate this process [68]. In this thesis, we specifically focus on an approach that relies on *link keys* [7].

Link keys, can be viewed as a generalization of keys across distinct datasets. They offer

[5]An international collaborative community committed to establishing open standards that foster the growth of the Web. https://www.w3.org

[6]Uniform Resource Identifier.

an explicit and unambiguous means of interlinking individuals. Link keys can be particularly advantageous because they operate as rules, directly declaring whether new individuals added to the datasets are the same. Furthermore, one of the strengths of link keys lies in their utilization as logical axioms in ontologies, which involves dealing with semantics. Another notable aspect of link keys is that they are applicable in data interlinking scenarios involving different ontologies or when no ontology alignment is available. This flexibility allows link keys to be employed in a wider range of situations.

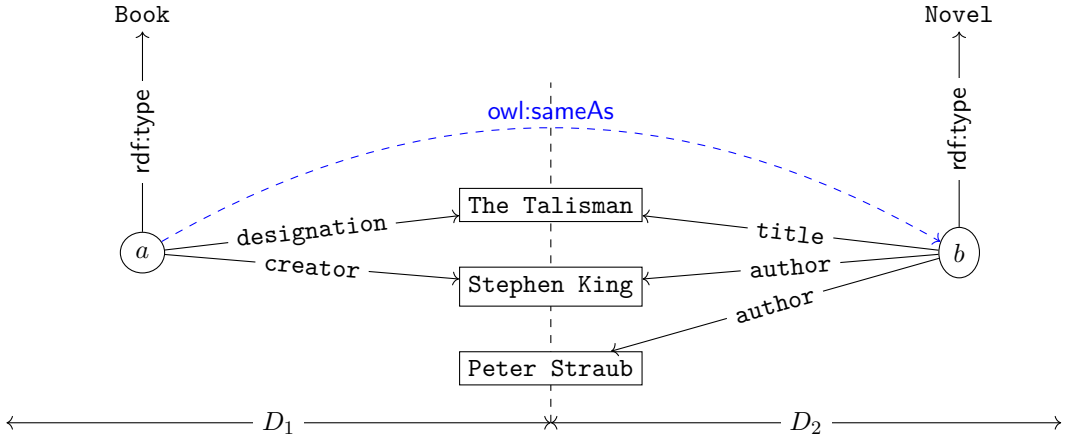


Figure 1.2: An illustration showcasing two individuals, a and b, representing the same entity. The individual a is an instance from the class `Book` in the dataset D_1, while the individual b belongs to the class `Novel` in the dataset D_2. These individuals are connected through owl:sameAs, meaning that a and b represent the same entity.

1.1 Link Keys

First introduced by Atencia et al. [7], a link key has the form of two sets of pairs of properties associated with a pair of classes. For example, let

$$k = (\{(\texttt{designation},\texttt{title})\},\{(\texttt{designation},\texttt{title}),(\texttt{creator},\texttt{author})\},(\texttt{Book},\texttt{Novel}))$$

If k is a link key over two datasets, this states that whenever[7] an instance a of the class `Book` and b of the class `Novel` (class condition) share at least one value for the properties `creator` and `author` ($\exists$-conditions) and that, a and b have the same values for the properties `designation` and `title` ($\forall$-conditions), then a and b denote the same entity. Consequently, $(a, \text{owl:sameAs}, b)$ is an identity link over the two datasets. We say that k generates the identity link (a,b).

In RDF an instance can have multiple values for a single property. As a result, property values are compared in different manners: (i) Determining whether instances share at least one value for a pair of properties, i.e., $\exists$-conditions and, (ii) considering whether instances have the same values for a pair of properties, i.e., $\forall$-conditions.

In Figure 1.2, the link key k generates the identity link (a,b) because:

1. a is instance of the class `Book` and b is instance of the class `Novel` (class condition), and

[7]Typically, we follow a left-to-right reading of k, but the order does not hold significance as there is an "and" between the conditions.

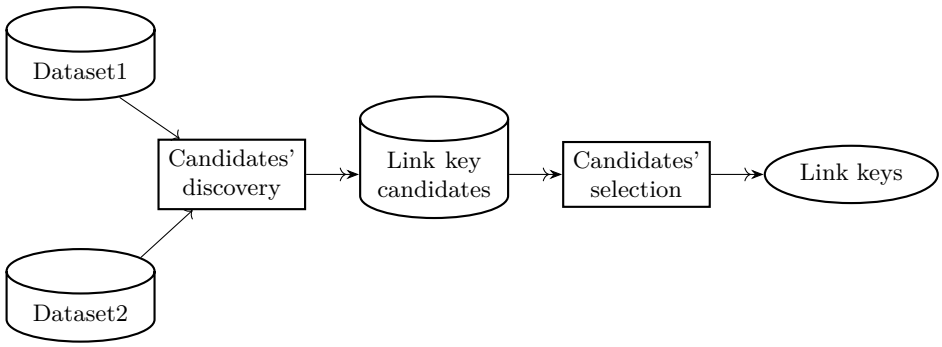


Figure 1.3: The pipeline of link key discovery.

2. a and b share the value `"Stephen King"` for the properties `creator` and `author` ($\exists$-conditions), and

3. a and b have the same values for the properties `designation` and `title` which is `"The Talisman"` ($\forall$-conditions).

Link keys are typically not provided within datasets. A straightforward automated method for discovering link keys involves exploring all possible combinations of properties and classes, referred to as *link key expressions*. However, to reduce the search space, link key discovery algorithms [7,10] focus on identifying a subset of these expressions known as *link key candidates*. These candidates are then assessed using specific measures to determine the most relevant ones that can be declared as link keys. Figure 1.3 depicts the various stages involved in the process of link key discovery.

A straightforward connections were established in [8] between link key discovery and Formal Concept Analysis (FCA [46]) as the definition of a link key candidate matched the definition of a formal concept discovered by FCA. Formal Concept Analysis, a mathematical framework for data analysis developed in the 1980s, allows for the identification of formal concepts that groups objects (forming the formal concept *extent*), based on their shared attributes (forming the formal concept *intent*). These concepts are organized in a hierarchical structure known as the formal concept lattice. Atencia et al. [8, 10] proposed utilizing FCA to discover link key candidates. Given two RDF datasets, this method identifies formal concepts, with the link key candidates as their intents and the set of links generated by these candidates as their extents.

1.2 Challenges in Link Key Discovery

The main focus of this thesis is to discover link keys from RDF datasets. Subsequently, we identify the following challenges and research questions associated with link key discovery.

The need for explicit pairs of classes in the discovered link keys The existing approaches [7, 10] for link key discovery do not explicitly specify the associated pairs of classes for the generated link keys. The absence of the pairs of classes presents a challenge when it comes to accurately evaluating the link key candidates. Indeed link keys may be relevant for one pair of classes (e.g., (`Book`,`Novel`)) but not for another (e.g., (`Scientist`,`Person`)). In addition, reasoning algorithms [54] based on link keys, require explicit specification of the associated pairs of classes to fully leverage their capabilities.

Furthermore, the Web of data allows for diverse perspectives on entities, leading to different levels of abstractions. For example, in one dataset, an individual belongs to the classes `Woman` and `Scientist`, while in the other dataset, the same entity belongs to only one class `FemaleScientist`. However, existing link key discovery methods do not support the identification of a link key associated with a pair of combined classes, i.e., class expressions, such as (`Woman` and `Scientist`, `FemaleScientist`).

Research question 1:*How can the FCA-based method be extended to discover link keys while explicitly identifying their pairs of classes and considering the varying levels of abstractions across datasets?*

The effectiveness of the existing selection method is limited Once the set of link key candidates has been identified, the subsequent step involves selecting valid link keys from this set. To determine the validity of a link key candidate algorithms employ a strategy of selecting the top-k link key candidates based on defined quality measures, as described in [7]. However, while this strategy proves effective when considering one pair of classes, it becomes less effective when evaluating candidates across datasets with multiple classes. In such cases, even if a candidate is a relevant link key, it may receive a low ranking according to quality measures. Additionally, highly ranked candidates may only generate links for specific classes while failing to generate any links for other classes.

Research question 2:*How to ensure an effective selection of link key candidates that yield valid link keys in datasets containing multiple classes?*

The search space for link keys remains large As mentioned previously, instead of exploring all possible link key expressions, the existing methods narrow down the search space to a subset of these expressions known as link key candidates. Indeed, a set of links can be generated by multiple link key expressions, and the ones that generate the same set of links are considered redundant. To address this redundancy, all link key expressions that generate the same set of links are represented by a single maximal link key expression, referred to as a link key candidate. The method based on Formal Concept Analysis [10] ensures that each link key candidate generates a unique set of links. This arises from the fact that every formal concept intent (link key candidate) corresponds to a distinct extent (set of links) and vice versa. However, despite this reduction, the resulting set of candidates remains large due to the vast volume of data in the Web. Consequently, this presents challenges for visualization, evaluation, and analysis of the link key candidates.

Research question 3:*Does the set of link key candidates still exhibit any form of redundancy that can be identified and subsequently reduced to facilitate the visualization, analysis, and evaluation of link key candidates?*

1.3 Contributions and Outline of the Thesis

This manuscript is organized as follows.

Chapter 2 In this chapter we review the fundamental concepts associated with the Web of data, providing a necessary foundation for comprehending the subsequent sections of this thesis. Here, we examine Knowledge Representation, which establishes the core principles and formalisms for structuring and organizing knowledge within the Web of data. Additionally, we

outline the standards established by the W3C that are based on the aforementioned knowledge representation formalisms.

Chapter 3 This chapter presents an overview of Formal Concept Analysis and its extensions, particularly *Pattern Structures* [44] and *Partition Pattern Structures* [15]. These extensions form the foundation for the key contributions of this thesis. Pattern structures are an extension of Formal Concept Analysis aimed at adapting the FCA formalism to objects whose descriptions are too complex to be expressed as a set of binary attributes. This is the case when objects have descriptions that can be numbers, intervals, sequences, trees, graphs. When the descriptions in a Pattern Structure are partitions it is called Partition Pattern Structures.

Chapter 4 This chapter introduces the task of data interlinking, which entails identifying and linking identical instances across multiple datasets. To contextualize our work, we explore various frameworks and approaches for data interlinking, with a particular emphasis on link keys. Throughout the chapter, we provide definitions and algorithms closely related to link keys.

Chapter 5 This chapter addresses the **Research question 1**. Our initial contribution presents a Pattern Structure that significantly improve of link key discovery process. Firstly, it enables the discovery of link key candidates while specifying their associated pairs of classes. This allows for a more precise evaluation of candidates and enhances the effectiveness of reasoning. Secondly, we extended the definition of link key candidate beyond being associated solely with a pair of atomic classes to include pairs of conjunctions and disjunctions of classes. This extension allows for the consideration of varying levels of abstraction across datasets. This chapter is based on the paper published in *the 15th International Conference on Concept Lattices and Their Applications* [5].

Chapter 6 This chapter addresses the **Research question 2**. Here, our focus is on selecting relevant link key candidates based on the proposed Pattern Structure. To achieve this, we introduced two methods. The first method, based on the LKSA algorithm, is guided by the pairs of classes associated with link key candidates. The second method, based on the SANDWICH algorithm, is guided by the lattice of candidates. This method employs a pruning strategy that incorporates both bottom-up and top-down approaches to select the candidates in the "middle " of the lattice. This chapter is based on two papers. The first one is published in *les 21èmes Journées Francophones Extraction et Gestion des Connaissances* [6]. The second one is published in *the 16th International Conference on Formal Concept Analysis* [2].

Chapter 7 This chapter addresses the **Research question 3**. In this chapter, we tackle the problem of redundancy among the link key candidates, aiming to obtain a more concise set of candidates that simplifies the analysis, evaluation, and selection processes. To do that, we firstly examine the partition induced by the owl:sameAs relation among the link set of each link key candidate. Candidates that generate the same partition are considered redundant. Consequently, we propose a Partition Pattern Structure to detect and merge redundant candidates based on the equality of their partitions. However, our experiments on real datasets reveal that redundancy based on partitions is relatively rare. Therefore, we propose an alternative approach that considers candidates generating similar set of links as redundant. To identify and minimize this redundancy, we develop an approach utilizing hierarchical clustering. Within

this approach, candidates producing similar set of links are grouped together into clusters, and we select a representative candidate from each cluster. This effectively minimizes redundancy among the link key candidates. This chapter is based on three papers. The first one is published in the *9th InternationalWorkshop "What can FCA do for Artificial Intelligence?"* [1]. The second one is published in *the 16th International Conference on Concept Lattices and Their Applications* [3]. The third paper is accepted for publication in July 2023, in *the International Journal of Approximate Reasoning* [4].

Chapter 8 This chapter concludes this manuscript by summarizing our contributions and outlining some research perspectives. The list of all our publications is available in Appendix A.

The work presented in this thesis has been conducted within the ELKER[8] – Enhancing Link Keys: Extraction and Reasoning ANR 2017 Project (ANR-17-CE23-0007-01), funded by the French National Research Agency. The aim of this project is to extend the foundations and algorithms of link keys in two complementary ways, first discovering link keys automatically from datasets and second reasoning with link keys.

[8]https://project.inria.fr/elker/

Chapter 2

The Web of Data

Contents

2.1 Knowledge Representation in the Web of Data 14
2.2 The Web of Data Standards 17

> " The first step is putting data on the Web in a form that machines can naturally understand, or converting it to that form. This creates what I call a Semantic Web – a web of data that can be processed directly or indirectly by machines."
>
> Tim Berners-Lee [18].

The Web of data is a global data space that can be thought of as an additional layer that is inextricably linked to the Web of documents [51]. This global space contains data from various sources covering a wide range of topics including scientific publications, geographic locations, music, clinical trials, etc. Additionally, anyone, with no restrictions on the vocabulary they use, can publish any type of data into this space.

This chapter explores the essential concepts related to the Web of data that are necessary to understand the remaining of this thesis. We begin by discussing Knowledge Representation, which provides the foundational principles and formalisms for structuring and organizing knowledge within the Web of data. We then describe the standards based on the aforementioned knowledge representation formalisms, as defined by the World Wide Web Consortium.

2.1 Knowledge Representation in the Web of Data

Data, information, and knowledge These terms are usually used interchangeably. Schreiber et al. [86] propose the following distinction:

- Data are the uninterpreted signals. For example, "`40`" is a sequence of symbols or signals. These signals are data as soon as no interpretation is attached to them.
- Information is data equipped with meaning. For example,"`40` degree Celsius is today's temperature". Here, new insights regarding the nature of "`40`" have been acquired.
- Knowledge is information with a purpose, it goes beyond a set of facts. It is the "intellectual machinery" that is used to accomplish a goal and to take actions. Furthermore,

knowledge is characterized by its ability to generate new information. For instance, when the temperature in Paris reaches 40 degrees Celsius, an alert is issued, and the government takes a series of protective measures. This has the potential to generate new information about the reality of climate change.

In a concise manner, Hogan et al. [52] assert that knowledge is *something that is known*.

Representation The term *Representation* is used to denote the connection between two domains, where the primary domain is aimed to "stand for" or take the place of the second domain [61].

Knowledge representation In artificial intelligence, knowledge representation is the process of formalizing knowledge in a way that can be understood and manipulated by machines.

Knowledge graph In [30], Ehrlinger and Wöß analyzed and discussed some definitions in the literature as there is no widely adopted definition of knowledge graphs. Hogan et al. [52] viewed *a knowledge graph as a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities.*

Ontology The word *Ontology* takes its root from philosophy where it refers to the study of existence. In knowledge representation, *an ontology is an explicit specification of conceptualization* [49]. A conceptualization refers to a view of the domain of interest that we represent for a particular task. An ontology is also viewed [32] as *a set of assertions that are meant to model some particular domain.* In the Web of data, ontologies are means to convey a shared understanding of a domain.

Description Logics Description logics (DLs) [13], a family of knowledge representation languages, form the foundation for knowledge representation in the Web of data. In $\mathcal{SROIQ}$, which is one of the most expressive DLs [57], a knowledge base consists of three components: the "TBox" containing terminological knowledge, i.e., the vocabulary of an application domain, the "ABox" for assertions about named individuals and the "RBox" for roles. There exist three fundamental entities in DLs [57]. Firstly, *Concepts* which denote the sets of individuals. For example, `Scientist` and `Book` can be regarded as concepts. Secondly, *Individual names* serve to identify single individuals within the domain. The individual `Marie Curie`, for example, can be considered an instance of the concept `Scientist`. Lastly, *Roles* are utilized to denote binary relations between individuals. An example of a role is `isColleague`, which establishes a connection between two individuals belonging to the concept `Scientist`.

An interpretation $\mathcal{I}$ consists of a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, called the domain of $\mathcal{I}$, and $\cdot^{\mathcal{I}}$ is a function that maps every atomic concept to a subset of $\Delta^{\mathcal{I}}$, every role name to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and each individual name to an element in $\Delta^{\mathcal{I}}$. The interpretation of combining atomic concepts and roles, adheres to the interpretation outlined in Table 2.1. Table 2.2 shows the syntax and semantics of $\mathcal{SROIQ}$ axioms.

The Closed World and the Open World Assumptions These assumptions differ in terms of how the lack of knowledge is interpreted in a given system. Within the *Closed World Assumption* (CWA), a true statement is also known to be true. As a result, a statement that

Constructor	Syntax	Semantics
Individuals		
Individual name	a	$a^{\mathcal{I}}$
Roles		
Atomic role	R	$R^{\mathcal{I}}$
Inverse role	R^{-}	$\{(x,y) \mid (y,x) \in R^{\mathcal{I}}\}$
Universal role	U	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Concepts		
Atomic concept	A	$A^{\mathcal{I}}$
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Complement	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Top concept	$\top$	$\Delta^{\mathcal{I}}$
Bottom concept	$\bot$	$\emptyset$
Existential restriction	$\exists R.C$	$\{x \mid \text{some } R^{\mathcal{I}} \text{ -successor of } x \text{ is in } C^{\mathcal{I}}\}$
Universal restriction	$\forall R.C$	$\{x \mid \text{all } R^{\mathcal{I}} \text{ -successor of } x \text{ is in } C^{\mathcal{I}}\}$
At-least restriction	$\geq nR.C$	$\{x \mid \text{at least } n \ R^{\mathcal{I}} \text{ -successors of } x \text{ are in } C^{\mathcal{I}}\}$
At-most restriction	$\leq nR.C$	$\{x \mid \text{at most } n \ R^{\mathcal{I}} \text{ -successors of } x \text{ are in } C^{\mathcal{I}}\}$
Local reflexivity	$\exists R.\text{Self}$	$\{x \mid (x,x) \in R^{\mathcal{I}}\}$
Nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$

Table 2.1: Syntax and semantics of $\mathcal{SROIQ}$ entities [57]. A is a concept name, C, D are concepts, R is a role and a is an individual. "$R^{\mathcal{I}}$-successor of x" means any individual y such that $(x,y) \in R^{\mathcal{I}}$.

Axiom Type	Syntax	Semantics
ABox		
Concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
Role assertion	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
Individual equality	$a \approx b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
Individual inequality	$a \not\approx b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$
TBox		
Concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Concept equivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
RBox		
Role inclusion	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
Role equivalence	$R \equiv S$	$R^{\mathcal{I}} = S^{\mathcal{I}}$
Complex role inclusion	$R_1 \circ R_2 \sqsubseteq S$	$(R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}}) \subseteq S^{\mathcal{I}}$
Role disjointness	$\text{Disjoint}(R,S)$	$R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$

Table 2.2: Syntax and semantics of $\mathcal{SROIQ}$ axioms [57]. C, D are concepts, R, R_1, R_2, S are roles and a, b are individuals. $R_1 \circ R_2$ is called role composition. For example, `motherOf` $\circ$ `fatherOf` $\sqsubseteq$ `grandmotherOf`.

is not presently identified to be true, is false. In contrast, the *Open World Assumption* (OWA) holds that a statement that is not known to be true does not imply that it is false. This statement is considered to be unknown.

Given the following knowledge base:

```
Marie Curie is a Scientist
Pierre Curie is a Scientist
```

The statement `Marie Curie visited Africa` is false under CWA and it is interpreted as unknown under OWA.

Typically, databases are interpreted in accordance with the CWA. The Web of data, on the other hand, has adopted the OWA due to its open and incomplete nature.

The Unique Name Assumption The Unique Name Assumption (UNA) means that different names always refer to different things. The Web of data does not make this assumption.

2.2 The Web of Data Standards

In the following, we review the Web of data terminology required to understand the remainder of this thesis, as well as the W3C standards.

Resource A *resource* refers to something in the world; an entity. It can be a person, a Web page, a building, etc. A given resource may be denoted by an identifier.

IRI URI An I*nternationalized Resource Identifier IRI* is a sequence of Unicode[9] characters following the syntax defined in [29]. IRIs are adopted by the W3C as identifiers for resources. The resource identified by an IRI is called its referent. IRIs are a generalization of URIs whose characters are restricted to a subset of US-ASCII characters. An IRI `http://example.com/ontology#Person` can be written as `ex:Person`. The prefix `ex` replaces the namespace portion of this IRI, i.e., `http://example.com/ontology#`[10].

Literal *Literals* denote data values such as strings and numbers. These values are described by a sequence of characters and a datatype may be specified. For example, we indicate that the sequence of character `"-40"` is an integer by adding `^^xsd:integer` [11]. If not specified this is interpreted as a string. A language tag maybe associated to a string. In Turtle, `"Années soixante"@fr` indicates that `Années soixante` is written in french.

Blank node A *blank node* indicates the existence of something without naming it. We can, for example, describe someone without naming him: “that scientist who discovered the radioactivity”.

[9] https://www.unicode.org/versions/latest/

[10] In the following, IRIs are written without a namespace or prefix, except for the standard ones like `rdf:type`.

[11] Using the Terse RDF Triple Language (Turtle) a textual syntax for RDF https://www.w3.org/TR/turtle/

Resource Description Framework RDF The W3C, adopted the *Resource Description Framework* [26] as a framework for representing data in the Web. The core component of this framework is the RDF statement which is a triple (s, p, o) where s, p and o are called the subject, predicate (or property) and object of the statement. In Figure 2.1, (e,`firstName`,`"Marie"`) is an RDF statement where e is a subject related to the literal `"Marie"` by the property `givenName`.

A subject can be an IRI or a blank node while a predicate has to be an IRI. An object can be an IRI, a literal or a blank node. An RDF graph is a set of RDF triples. An RDF dataset is a collection of RDF graphs.

To specify that an entity is an instance of a given class, RDF is equipped with the predicate `rdf:type`. In Figure 2.1, the triple (e,`rdf:type`,`Scientist`) states that the subject e is an instance of the class `Scientist`.

It is important to note that properties within the RDF are not necessarily functional. This can be observed in the example, where entity e is associated with two literals through the property `familyName`. Additionally, there are cases where subjects within RDF may not have any associated object for a particular property. In the example, d is an instance of a class `City`, but unlike b, there is no object associated with d through the property `designation`. This indicates that we are aware that d represents a city, however, specific information regarding its name or designation remains unknown.

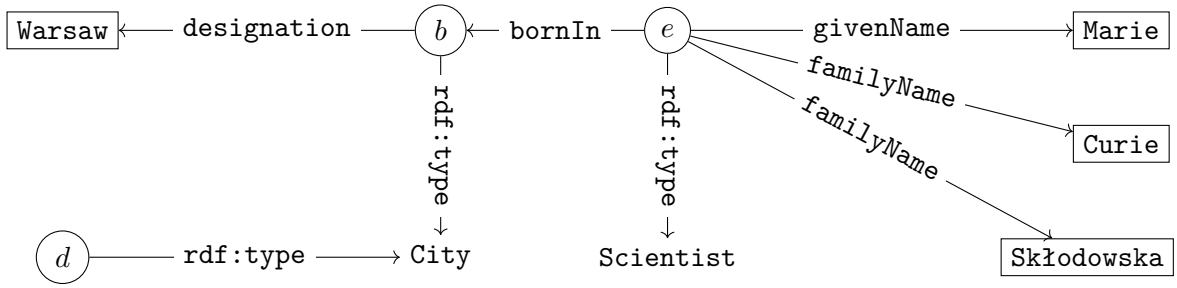


Figure 2.1: Example of an RDF dataset, where e is a subject related to the literal `"Marie"` by the property `givenName` and to the literals `"Curie"` and `"Skłodowska"` by the property `familyName`. The property `rdf:type` relates the subject e to its class `Scientist`. The subject e is also related to the subject b by the property `bornIn`.

Resource Description Framework Schema RDF(S) RDF Schema is an extension of RDF [22]. It provides for example the predicate `rdfs:subClassOf` which expresses subsumption between two classes and the predicate `rdfs:subPropertyOf` which expresses the subsumption between two properties. Like in the triples `(Scientist,rdfs:subClassOf,Person)` and `(motherOf,rdfs:subPropertyOf,parentOf)`. RDF(S) allows also to define the domain and the range of predicates thanks to `rdfs:domain` and `rdfs:range`. For the property `bornIn`, we may have the triple `(bornIn,rdfs:domain,Person)` and `(bornIn,rdfs:range,City)`.

Web Ontology Language OWL RDF and RDF(s) offer a limited expressiveness. Thus, the Web Ontology Language was designed to express more advanced knowledge about a given domain [65]. For example, to declare two equivalent classes, `owl:equivalentClass` is used, such as `(Human,owl:equivalentClass,Person)`. While the OWL property `owl:sameAs` links an individual to an individual indicating that their IRIs refer to the same thing.

The fundamental elements of OWL are similar to those of DLs, with the primary distinction being that concepts are referred to as classes and roles are referred to as properties. The W3C recommendation given in [64], provides the direct model-theoretic semantics for OWL 2 [12], which is compatible with the description logic $\mathcal{SROIQ}$.

SPARQL Protocol and RDF Query Language SPARQL *SPARQL Protocol and RDF Query Language* [48] is a query language and protocol for RDF graphs. SPARQL queries are built on basic graph patterns, which are sets of triple patterns enclosed on the *where* clause. An example of SPARQL query is given in Figure 2.2. Furthermore, SPARQL allows for the creation of advanced query patterns for filtering the results. The SPARQL protocol is used to communicate queries, and the SPARQL result format is used to represent query results in XML [13].

```
SELECT ?person
WHERE { ?person rdf:type Scientist.
?person familyName "Curie" }
```

Figure 2.2: Example of a SPARQL query for which the results are instances of the class `Scientist` whose `familyName` is `"Curie"`.

Linked data Organizations and individuals contribute to building the Web of data by publishing data on the Web. These data must be linked to ensure that people or machines can explore the Web of data and get the most out of it. Hence, linked data can be viewed as a means for achieving the Semantic Web vision. Tim Berners-Lee [17] introduced a set of rules, known as the "Linked Data principles", for publishing and interlinking structured data on the Web:

- Use URIs as names for things.
- Use HTTP[14] URIs, so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
- Include links to other URIs, so that they can discover more things.

Linked Open Data (LOD) refers to linked data that is released under an open license, which allows its reuse. Berners-Lee [17] provides a star rating system for data providers to determine whether their data sets meet the requirements of the LOD:

⋆	Available on the web with an open licence, to be Open Data
⋆⋆	Available as machine-readable structured data
⋆⋆⋆	Available in non-proprietary format
⋆⋆⋆⋆	Use open standards from W3C (RDF and SPARQL) to identify things
⋆⋆⋆⋆⋆	Link your data to other people's data to provide context

[12] OWL 2 is an extension and revision of the OWL published in 2004 referred to as OWL 1 [62].
[13] Extensible Markup Language https://www.w3.org/XML/
[14] Hypertext Transfer Protocol.

The star rating system can be adapted to assess linked data that are not open. In this category we may find governmental and company datasets that are not openly available. DBpedia [60], YAGO [33], and Wikidata [95] are among the most well-known LOD sets. DBpedia extracts structured data from Wikipedia, leveraging the comprehensive information available on the platform[15], and presents it in RDF format. YAGO, built upon Wikipedia, enriches its knowledge base by incorporating WordNet[16], a substantial English lexical database. On the other hand, Wikidata, initiated by the Wikimedia Foundation[17], as a collaborative knowledge base.

Ringler and Paulheim [82] emphasized that while DBpedia, YAGO, and Wikidata are often considered similar in terms of nature and coverage, they exhibit distinct differences. A comparative analysis between these LOD sets and others is presented in [82]. Among the findings of this study is that DBpedia contained the largest collection of artistic works, YAGO emerged as a highly suitable source for events data, and Wikidata proved to be the most appropriate for obtaining person data.

The fundamental concepts related to the Web of data are examined in this chapter, providing a basis for understanding the rest of this thesis. Additionally, the significance of links in the Web of data is highlighted. Links are included as the fourth principle in the "Linked Data principles", enabling the creation of a more connected and interoperable Web, where data can be easily shared, reused, and analyzed. The following chapter is about Formal Concept Analysis, a mathematical framework utilized for data analysis. Formal Concept Analysis has numerous applications in the Web of data, and more importantly, it serves as the foundation for the main contributions of this thesis.

[15] https://www.wikipedia.org
[16] https://www.wordnet.princeton.edu
[17] https://www.wikimediafoundation.org

Chapter 3

Formal Concept Analysis

Contents

3.1 Basic Notions of Formal Concept Analysis 21

3.2 Pattern Structures 24

This chapter provides an overview of Formal Concept Analysis (FCA) and its extensions, particularly Pattern Structures and Partition Pattern Structures that serve as the basis for the principal contributions of this thesis. FCA is a mathematical framework for data analysis that was developed in the 1980s. It enables the identification of formal concepts and their relationships within a provided dataset. In the forthcoming sections, it will become apparent that the construction and description of concepts in Description Logics and formal concepts in Formal Concept Analysis differ significantly [14], despite both involving sets of individuals (or objects in the case of FCA) that share specific properties.

3.1 Basic Notions of Formal Concept Analysis

Formal Concept Analysis [46] takes as input a *formal context* which is typically represented as a binary table where the rows represents *objects*[18], the columns represents *attributes* and a cross $\times$ in a cell (i, j) is interpreted as the object in the i^{th} row has the attribute in the j^{th} column (See Example 1). Formally, the formal context is defined as follows:

Definition 1 (Formal context)**.** *A formal context is a triple (G, M, I), where G denotes a set of objects, M a set of attributes, and $I \subseteq G \times M$ a binary relation between G and M. The statement $(g, m) \in I$ (also denoted by gIm) is interpreted as "object g has attribute m".*

Example 1. *Table 3.1 illustrates an example of a formal context, where the objects are animals, $G = \{$`dog, dolphin, donkey, duck, eagle`$\}$ and the attributes represent animals characteristics, $M = \{$`mammal, domestic, feather, live in the sea`$\}$. The symbol $\times$ in a given table cell, indicates that the object in the corresponding row has the attribute in the corresponding column. For example, the first row of Table 3.1, reads "*`dog`* has the attributes* `mammal` *and* `domestic`*".*

[18] Objects in FCA are not necessarily objects in RDF, and vice versa.

objects	attributes			
	mammal	domestic	feathers	live in the sea
dog	×	×		
dolphin	×			×
donkey	×	×		
duck		×	×	
eagle			×	

Table 3.1: An example of a formal context where the objects are animals (in rows) and the attributes are their characteristics (in columns).

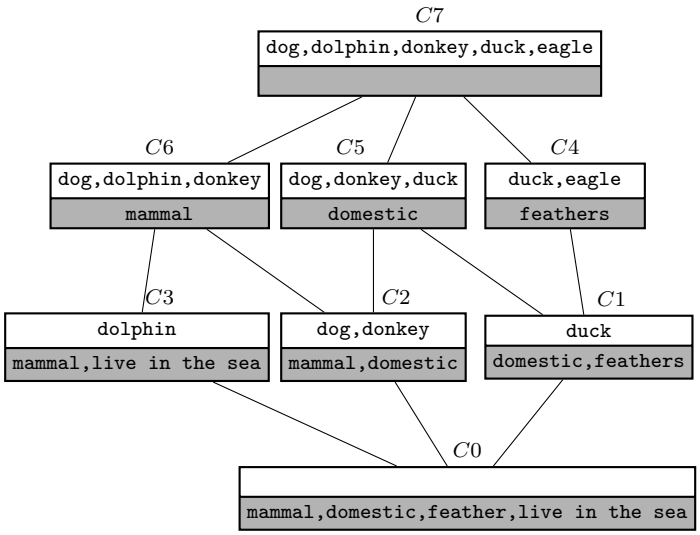


Figure 3.1: The Hasse diagram of the lattice generated from the formal context in Table 3.1.

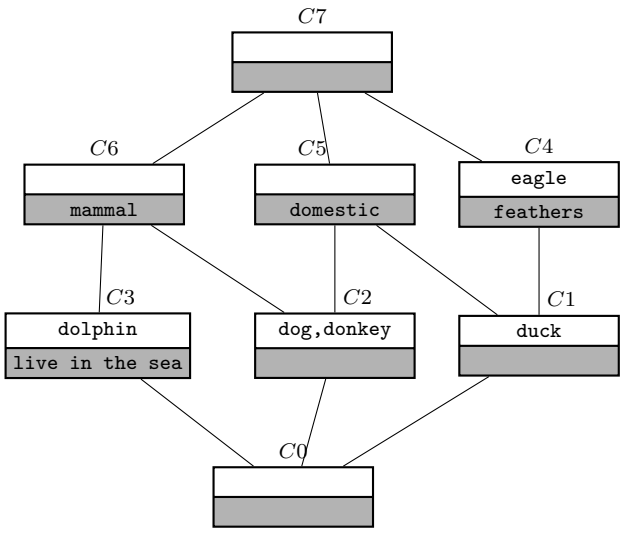


Figure 3.2: The “reduced” notation of the lattice in Figure 3.1. An object appearing in a concept is “inherited” by all concepts which are above. Dually, an attribute appearing in a concept is “inherited" by all concepts which are below.

Two derivation operators $(\cdot)'$ define a Galois connection between the powersets $(2^G, \subseteq)$ and $(2^M, \subseteq)$

$$\begin{aligned} A' &= \{m \in M \mid \forall g \in A : gIm\} & A \subseteq G \\ B' &= \{g \in G \mid \forall m \in B : gIm\} & B \subseteq M \end{aligned}$$

The closure operator $(\cdot)''$ denotes the double application of $(\cdot)'$. This operator is extensive, idempotent and monotone.

Given a formal context and the derivation operators, FCA generates *formal concepts*. A formal concept is a pair (A, B) such that $A' = B$ and $B' = A$ where $A \subseteq G$, $B \subseteq M$. The "extent" of the formal concept (A, B) is the set of objects A and its "intent" is the set of attributes B. The set of formal concepts is denoted by $\mathfrak{B}(G, M, I)$. Formal concepts are partially ordered by the subsumption relation $\leq$ as follows:

$$(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 \ (\Leftrightarrow B_2 \subseteq B_1)$$

With respect to this partial order, the set of all formal concepts forms a complete lattice called the "formal concept lattice" of the formal context (G, M, I) denoted by $\mathfrak{L}(G, M, I)$. This lattice can be represented by a Hasse diagram where the nodes depict the formal concepts, and the edges represent the subsumption relation between them. The reflexivity and transitivity edges are omitted.

The Figure 3.1 illustrates the formal concept lattice of the formal context in Table 3.1. An example of a formal concept in this lattice is $C2 = (\{\texttt{dog,donkey}\},\{\texttt{mammal,domestic}\})$ which is subsumed by the formal concept $C5 = (\{\texttt{dog,donkey,duck}\},\{\texttt{domestic}\})$. The top formal concept of this lattice is $C7$ and the bottom formal concept is $C0$.

A formal concept lattice can be represented using a simplified or "reduced" notation. In this notation, any object present in a concept is considered to be inherited by all the concepts above it, while any attribute present in a concept is considered to be inherited by all the concepts below it. Figure 3.2 shows an example of the reduced notation for the lattice presented in Figure 3.1. For example, the objects in the extents of the concepts $C2$ and $C3$ in Figure 3.2 belong to the extents of the concepts $C6$ and $C7$, while the attribute in the intent of the concept $C6$ belongs to the intents of the concepts $C2$, $C3$, and $C0$. Throughout this thesis, we will explicitly indicate the use of the reduced notation whenever it is employed.

In the reduced notation, each element (object or attribute) is represented only once in the concept that "introduces" it. This concept is referred to as the *introducer concept* which is defined as follows. Consider a formal concept C in the context $\mathfrak{B}(G, M, I)$. If an object $g \in G$ belongs to the extent of C but it does not belong to the extent of any smaller concept than C (the concepts below C) in $\mathfrak{L}(G, M, I)$, then it is said that C "introduces" g, and C is the *object-concept* of g. Dually, a concept C "introduces" an attribute $m \in M$, if m belongs to the intent of C but it does not belong to the intent of any greater concept than C (the concepts above C) in $\mathfrak{L}(G, M, I)$. In this case, C is called the *attribute-concept* of the attribute m. Both object-concepts and attribute-concepts are referred to by *introducer concepts*. Accordingly, an *AOC-poset* (Attribute/Object Concept poset) [20] is a partial order defined on introducer concepts in the concept lattice. When the lattice becomes larger in size, AOC-posets are used in certain application to simplify the visualization of lattices, while showing the relevant information.

An *OC-poset* is defined on object-concepts while *AC-poset* is defined on attribute-concepts. The OC-poset corresponding to the lattice in Figure 3.1 is depicted in Figure 3.3. It should be noted that the concepts C_0, C_5, C_6 and $C7$ are not included in the OC-poset since they do not introduce any object.

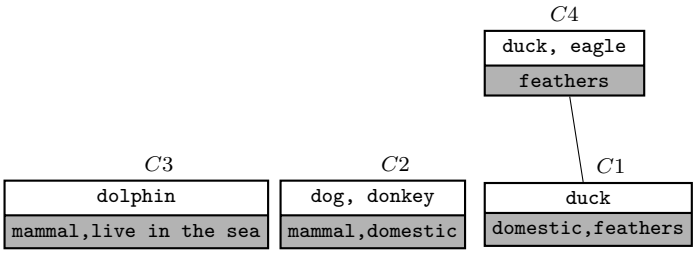


Figure 3.3: The Hasse diagram of the OC-poset corresponding to the lattice in Figure 3.1.

3.2 Pattern Structures

In practical applications, it is common to encounter non-binary data, which has led to the development of various extensions of FCA. Among these extensions, the conceptual scaling, introduced in [45]. This approach takes complex data as input and generates a traditional formal context known as the "scaled context". Furthermore, to handle three-dimensional data, particularly involving the relation "an object has an attribute under a condition," Triadic Concept Analysis was proposed in [59, 98]. Another extension, Logical Concept Analysis, was introduced in [38], where sets of attributes are replaced by logical formulas. Additionally, Relational Concept Analysis [84] serves as an extension of FCA specifically designed to handle relational datasets involving formal contexts and relations between the objects of these contexts.

The current section focuses on a particular extension of FCA, referred to as *Pattern Structures*, due to its pertinence to the contributions made by this thesis work. Pattern Structures extends FCA to deal with formal contexts in which the objects are described by complex data such as numbers, sequences, or graphs.

Let G be a set of objects, let $(D,\sqcap)$ be a meet-semi-lattice of potential object descriptions and let $\delta : G \longrightarrow D$ be a mapping taking each object to its description. Then $(G,(D,\sqcap),\delta)$ is a Pattern Structure. Elements of D are patterns and are ordered by a subsumption relation $\sqsubseteq$: $\forall c,d \in D,\ c \sqsubseteq d \Longleftrightarrow c \sqcap d = c$. An example of a Pattern Structure is given in Figure 3.4 (a), where each object is associated with a description. For example, the object g_1 is described by d_8. These descriptions are organized in a meet-semi-lattice as it is shown in Figure 3.4 (b). For instance, the meet of two descriptions d_4 and d_5 is $d_4 \sqcap d_5 = d_1$.

In classical FCA, object descriptions are sets of attributes, which are partially ordered by set inclusion, w.r.t. set intersection: let $P,Q \subseteq M$ be two attributes sets, then $P \subseteq Q \Leftrightarrow P \cap Q = P$, and $(M,\subseteq)$, also written $(M,\cap)$, is a partially ordered set of object descriptions. Set intersection $\cap$ is a meet operator, i.e., it is idempotent, commutative, and associative. A Galois connection can then be defined between the powerset of objects $(2^G,\subseteq)$ and a meet-semi-lattice of descriptions denoted by $(D,\sqcap)$ (standing for $(M,\cap)$). This idea is used to define Pattern Structures in the framework of FCA [37].

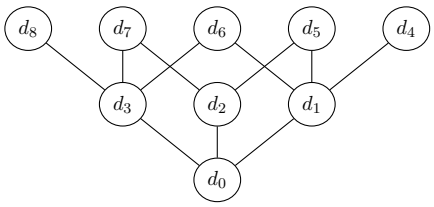
A Pattern Structure $(G,(D,\sqcap),\delta)$ gives rise to two derivation operators $(\cdot)^\square$:

$$A^\square = \bigsqcap_{g \in A} \delta(g), \quad for\ A \in 2^G \quad and \quad d^\square = \{g \in G \mid d \sqsubseteq \delta(g)\}, \quad for\ d \in D.$$

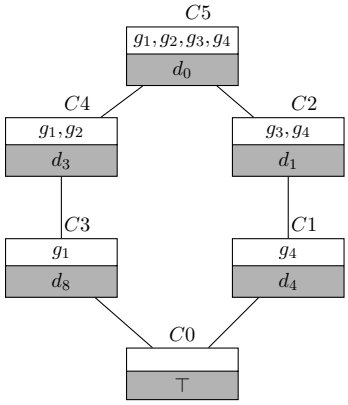
These operators form a Galois connection between $(2^G,\subseteq)$ and $(D,\sqcap)$. Pattern concepts of $(G,(D,\sqcap),\delta)$ are pairs of the form (A,d), $A \subseteq G$, $d \in (D,\sqcap)$, such that $A^\square = d$ and $A = d^\square$. For a pattern concept (A,d), d is called "pattern intent" and it is the common description of all objects

Objects	Descriptions
g_1	d_8
g_2	d_3
g_3	d_1
g_4	d_4

(a) An example of a Pattern Structure.



(b) The meet-semilattice of descriptions.



(c) The pattern concept lattice of the Pattern Structure in (a). The $\top$ symbol is an arbitrary description added to the meet-semilattice of descriptions in order to make it a lattice.

Figure 3.4: An example of a Pattern Structure (a) such that the descriptions are organized in a meet-semilattice (b). The Pattern Structure in (a) generates the pattern concept lattice in (c).

from A, called "pattern extent". When partially ordered by $(A_1, d_1) \leq (A_2, d_2) \Leftrightarrow A_1 \subseteq A_2 \ \ (\Leftrightarrow d_2 \sqsubseteq d_1)$, the set of all pattern concepts forms a complete lattice called "pattern concept lattice". Figure 3.4 (c) represents the pattern concept lattice of the Pattern Structure in Figure 3.4 (a).

The descriptions can be intervals. An example is given in Figure 3.5 where the objects in the Pattern Structure are described by intervals. The similarity of two intervals is their convex hull, i.e., $[a,b] \sqcap [c,d] = [min(a,c), max(b,d)]$.

Objects	**Descriptions**
g_1	$[1,2]$
g_2	$[2,3]$
g_3	$[3,4]$

Table 3.2: A Pattern Structure where the objects are described by intervals.

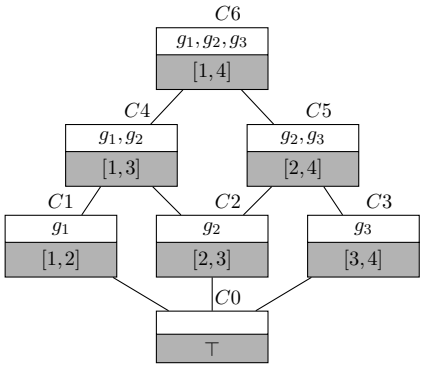


Figure 3.5: The pattern concept lattice related to the Pattern Structure given in Table 3.2.

When object descriptions of a Pattern Structure are partitions it is called *Partition Pattern Structures*. In the following we recall the definition of a partition.

Partition of a set

A *partition* of a given set E is made up of disjoint non-empty subsets of that set whose union equals E. The subsets are called *blocks*. Formally, a partition of a set E is a set $Par \subseteq 2^E$ such that:

- $\bigcup_{X \in Par} X = E$
- $X \cap Y = \emptyset$, for any $X, Y \in Par$.

For example, $Par = \{\{a,b,c\},\{d\}\}$ is a partition of the set $E = \{a,b,c,d\}$.

Equivalence relations and partitions

Partitions and equivalence relations are inextricably linked. If R is an equivalence relation on the set E, then the set of all equivalence classes defined by R forms a partition of E. This partition is called the quotient set of E by R and denoted E/R. For example, let R be an equivalence relation on the set E such that:
$E = \{a,b,c,d\}$ and $R = \{(a,a),(a,b),(a,c),(b,a),(b,b),(b,c),(c,a),(c,b),(c,c),(d,d)\}$. Then, the equivalence class of a, denoted as $[a]$, is equal to the equivalence classes of b and c, $[a] = [b] = [c] = \{a,b,c\}$ and $[d] = \{d\}$ thus $E/R = \{\{a,b,c\},\{d\}\}$.

Objects	Descriptions
g_1	$\{\{a,b\},\{c\},\{d\}\}$
g_2	$\{\{a,b,c\},\{d\}\}$
g_3	$\{\{a\},\{b,c\},\{d\}\}$

Table 3.3: Example of a partition Pattern Structure.

Partitions order A partition Par_1 is finer than a partition Par_2, and Par_2 is coarser than Par_1, if any block of Par_1 is a subset of a block in Par_2. The "finer than" relation on the set of partitions of E is a partial order denoted by $\sqsubseteq_{part}$. For example, $\{\{a,b\},\{c\},\{d\}\} \sqsubseteq_{part} \{\{a,b,c\},\{d\}\}$.

The meet of two partitions, denoted by $\sqcap_{part}$, corresponds to their coarsest common refinement. That is, the blocks of the meet of two partitions Par_1 and Par_2 is formed by all non-empty intersections of a block from Par_1 with a block from Par_2. For example,

$$\begin{aligned}\{\{a,b\},\{c\},\{d\}\} \sqcap_{part} \{\{a,b,c\},\{d\}\} =& \{\{a,b\} \cap \{a,b,c\}, \{a,b\} \cap \{d\},\\ & \{c\} \cap \{a,b,c\}, \{c\} \cap \{d\},\\ & \{d\} \cap \{a,b\}, \{d\} \cap \{d\}\}\\ =& \{\{a,b\},\{c\},\{d\}\}\end{aligned}$$

The following property holds $Par_1 \sqcap_{part} Par_2 = Par_1 \Leftrightarrow Par_1 \sqsubseteq_{part} Par_2$. Then the Pattern Structure $(G,(Partitions,\sqcap_{part}),\delta)$ is a Partition Pattern Structure with $Partitions$ represents the set of descriptions which are partitions in this case. Table 3.3 is an example of a Partition Pattern Structure, where the object g_1 is described by the partition $\{\{a,b\},\{c\},\{d\}\}$, and Figure 3.6 is the pattern concept lattice of the a Partition Pattern Structure in Table 3.3 .

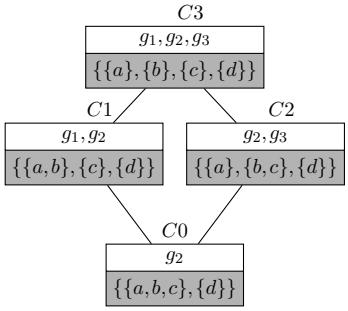


Figure 3.6: The pattern concept lattice related to the Partition Pattern Structure given in Table 3.3.

In this section, we have provided an introduction to the fundamental principles of Formal Concept Analysis and Pattern Structures. FCA has been applied in various tasks concerning the Web of data, such as ontology matching [91] and *data interlinking* [36], which is the focus of the next chapter.

Chapter 4

Data interlinking

Contents

4.1 Identity 28
4.2 Data Interlinking Approaches 29
4.2.1 Embedding-based Approaches 30
4.2.2 Data Interlinking Frameworks 31
4.2.3 Key-based Approaches 32
4.3 Link Keys 34
4.3.1 Syntax 34
4.3.2 Semantics 37
4.3.3 Link Key Discovery 38
4.4 Discussion 44

In this chapter, we present the task of data interlinking, which is also referred to as record linkage or entity resolution. This task involves the identification and linking of the same entities that exist in multiple datasets. We explore some frameworks and approaches for data interlinking, with a specific focus on link keys. A link key can be seen as a generalization of a key for connecting two datasets. Throughout the chapter, we present definitions and algorithms that are closely associated with link keys. Additionally, we examine the quality measures that are pertinent to assess the quality of link keys.

4.1 Identity

Identity has been for a long time the subject of several philosophical debates. A key principle that influences the concept of identity is "the Identity of Indiscernibles" principle. It states that if two entities a and b have all the same properties, then a is identical to b. Conversely, the "Indiscernibility of Identicals" states that if a is identical to b, then a and b have all the same properties. These principles are known as the "Leibniz's Law" [40].

In light of this law, several questions arise. For example, consider two cups of coffee having the same `color`, `dimensions`, and `texture`. Given these properties, can we conclude that the cups are identical? Furthermore, what if the property `position in the space` is taken into account, are the cups still identical? Now, consider that the cup of coffee has fallen off and cracked, is it the same cup since it no longer possesses the same properties? Gallois [42] argues that it is relevant to believe that entities can be identical one moment and distinct the next.

To investigate another aspect of identity, consider the following question : Are `Batman`[19] and `Bruce Wayne` the same person? For the people of `Gotham City`, they are not identical: `Batman` is the superhero, and `Bruce Wayne` is the wealthy philanthropist. For his butler, however, `Batman` and `Bruce Wayne` are the same person. When it comes to people watching the movie, is `Batman` the superhero himself or the actor portraying him? Philosophers have raised more complex issues like the "Ship of Theseus paradox" : Would a ship still be the same after replacing each of its component one by one? Many of these paradoxes are described in [28]. All of these questions have given rise to more complex views of identity: transworld identity i.e. identity across possible worlds, vague identity, relative identity, etc. [77].

Because the Web of data aims to represent the real world, all previous identity issues are expanded in this space. The Web of data is indeed capable of representing disagreement and contradictory information about a given entity. However, to ensure interoperability, entities are interconnected through links, particularly identity links. Following the principle of the indiscernibility of identicals, an identity link in the Web of data, represented as an RDF triple $(a, \mathsf{owl{:}sameAs}, b)$, signifies that the IRIs a and b are referring to the same entity. The task of identifying such links is known as *data interlinking* [34].

4.2 Data Interlinking Approaches

The process of finding links is related to various similar problems [34], leading to a lack of clarity and confusion surrounding the associated terminology. In their study, Ferrara et al. [34] explored the related terminology in the field of data linking and attempted to provide understanding of the terminology landscape within this field. They categorized the terminology based on three dimensions: (i) the objective of the comparison, which includes data cleaning and data integration; (ii) the object of comparison, encompassing unstructured data, structured data (ontological instance); and (iii) the semantic nature of the resulting links, distinguishing between generic relations of similarity between entities or identity relations. Consequently, data interlinking can be viewed as a subtask of data linking, where the primary objective is data integration targeting structured RDF datasets, and the resulting links represent identity relations.

In the field of databases, finding links among data has been extensively studied and commonly referred to as record linkage, data cleaning, entity resolution, or data deduplication [31]. A fundamental distinction between record linkage and data interlinking lies in the fact that the former considers generally a single database where all entities are described using a unified vocabulary, typically with functional properties and operating under the Closed World Assumption. In contrast, data interlinking focuses on multiple datasets described by diverse vocabularies, operating under the Open World Assumption, where properties may not necessarily be functional.

In formal Concept Analysis, a method proposed by Ferré [36] allows discovering equivalence links between properties, classes, and individuals across knowledge graphs, i.e., data interlinking. Two different scenarios are considered: (i) when the two knowledge graphs share common values, and (ii) when pre-aligned pairs are known. This method is based on an extension of FCA called *Graph-FCA*, which has been introduced in [35, 39] to address multi-relational data, particularly within knowledge graphs in the Web of data. While FCA defines a formal context as a relation between objects and attributes, Graph-FCA defines a "graph context" as a relation between tuples of objects and attributes.

[19] https://dbpedia.org/page/Batman

In the Web of data, data interlinking and ontology matching [32] can be viewed as complementary tasks. On one hand, when dealing with two datasets that employ different ontologies, data interlinking can benefit from ontology matching to find identity links. On the other hand, ontology matching can employ extensional or instance-based methods that can leverage data interlinking. They can also be utilized simultaneously, starting from one task, to contribute to the outcome of the other, until eventually yielding positive results [85]. The term "ontology matching" is sometimes used to refer to both schema matching and data interlinking.

In the subsequent sections, we outline several data interlinking approaches to contextualize our work.

4.2.1 Embedding-based Approaches

Knowledge graph embedding approaches have gained significant attention and research activity in recent years. The ability to represent knowledge graphs in RDF establishes a relevant connection between our approach and knowledge graph embedding approaches, which can indeed be employed for identifying identity links.

A model for knowledge graph embedding generates for each entity (and sometimes each property) a vector that represents it in a low dimension space. These vectors are referred to as embeddings. The primary objective of knowledge graph embedding is to encode the structural information of the knowledge graph such that the similarity of two entities can be captured by the similarity of their correspond embeddings.

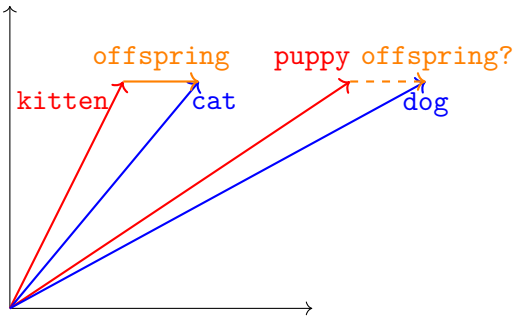


Figure 4.1: An example of the projection of the triple `(kitten,offspring,cat)` on a two dimensions embedding space. The subject `puppy` and the object `dog` are also projected on this space. A translational distance model [21] could predict the missing link `(puppy,offspring,dog)`.

Knowledge graph embedding models are used to perform *link prediction*, a task that identifies missing links or estimates the probability of new links. Hogan et al. [52] defined three types of link prediction tasks in knowledge graphs: (i) General link prediction where the missing links are arbitrary. For example, the missing link `(puppy,offspring,dog)` in Figure 4.1. (ii) Type-link prediction where the missing links are the ones relating an entity to its class. (iii) Finally, the identity-link prediction where the missing links are the identity links. The task of data interlinking, which is the focus of this thesis work, can be seen as a subtask of link prediction across knowledge graphs.

Numerous works on link prediction focus on finding links within a single knowledge graph using different embedding models [81,83,96]. Translational distance models, such as TransE [21], consider link prediction as a geometric task, where the graph is projected onto a vector space and relations translate subjects to objects. Tensor decomposition models, on the other hand, treat

a graph as a multi-dimensional tensor that is decomposed into tensors capturing the underlying graph structure. DistMult [99] serves as an example of a tensor decomposition model utilized for link prediction in graphs. Other approaches, such as ConvE [27], are based on deep neural networks.

Unlike link prediction within a single knowledge graph, the task of link prediction across knowledge graphs poses a significant challenge, as the comparison of entity embeddings across different spaces is not straightforward. Hao et al. [50] proposed a model that jointly learns the embeddings of the entities and the relations given two knowledge graphs and some initial alignments referred to by *seed entity alignments*. Here entities having similar embeddings are considered as the same entities. The approach proposed by Zhu et al. [100] involves the utilization of an iterative alignment procedure where the highly confident aligned entities are incorporated to update the joint knowledge embeddings. Chen et al. [24] were interested in cross-lingual knowledge graph alignment. They introduce MTransE, a translation-based model for multilingual knowledge graph embeddings, where entities and relations of each language are projected in a separated embedding space. This approach is based on TransE and requires a set of aligned triples of two knowledge graphs. Wang et al. [97] also aimed to align cross-lingual knowledge graphs. Their approach is based on graph convolutional networks given a set of pre-aligned entities. Here, entities of each language are embedded into a unified vector space. Additional embedding-based approaches for link prediction across knowledge graphs are presented and compared in [16].

Baumgartner et al. [16] emphasize the importance of bringing interoperability to knowledge graphs embeddings. Consequently, data providers possess the freedom to utilize any embedding model of their choice and have the flexibility to select the hyperparameters that best suit their data. Building upon this idea, Baumgartner et al. introduce the concept of the "Web of Embeddings" to mirror the concept of the Semantic Web that is distributed and interoperable. Within the Web of Embeddings, embedding spaces are integrated into a unified space. This allows applications to leverage and combine different knowledge graph embedding spaces, regardless of the embedding methods and hyperparameters employed in their construction. In the same work [16], Baumgartner et al. introduce FedCoder, an approach that integrates multiple heterogeneous knowledge graph embedding spaces via a latent space and autoencoders. In this latent space, entities are comparable across embedding spaces.

4.2.2 Data Interlinking Frameworks

Numerous frameworks have emerged to establish links between different datasets, taking into account the conditions that two instances must satisfy to be linked. These conditions are commonly referred to as link specifications or linkage rules. They generally involve determining the properties to be compared, the transformations applied to these properties, the similarity measures utilized for comparing pairs of property values, the aggregation functions employed for combining multiple similarity values, and the thresholds that determine when two values are considered similar. LIMES [76] is one of these frameworks where specifications can be manually declared by a user who selects the properties to compare, the metrics to use, the classes to link, and many other parameters. Alternatively, LIMES offers an automatic discovery of these specifications. It includes a collection of machine learning algorithms that can be customized for different scenarios, such as unsupervised, active, and batch learning [71, 72, 74, 75, 87, 90]. In order to ensure efficiency in real-world applications with large datasets, LIMES incorporated time-efficient techniques that leverage the characteristics of metric spaces [69, 70], orthodromic spaces [73], and filter-based paradigms [89].

```
<LinkageRule linkType="<http://www.w3.org/2002/07/owl#sameAs>">
<Aggregate id="average1" weight="1" type="average">
<Compare id="jaccard1" weight="1" metric="jaccard" threshold="0.5" indexing="true">
<Input id="sourcePath1" path="<http://exemple.fr/o1#familyName>"/>
<Input id="targetPath1" path="<http://exemple.fr/o2#name>"/>
</Compare>
<Compare id="jaccard2" weight="1" metric="jaccard" threshold="0.5" indexing="true">
<Input id="sourcePath2" path="<http://exemple.fr/o1#given>"/>
<Input id="targetPath2" path="<http://exemple.fr/o2#firstName>"/>
</Compare>
</Aggregate>
<Filter/>
</LinkageRule>
```

Figure 4.2: An excerpt of a link specification file of SILK.

SILK [94] is another framework that provides the capability for manual configuration of link specifications as well as supervised learning of these specifications. The framework utilizes the declarative SILK - Link Specification Language (SILK-LSL) to describe linkage rules and specify which types of RDF links should be discovered. Figure 4.2 depicts a segment of a link specification that has been provided to SILK, aimed at discovering identity links. This specification encompasses the properties that necessitate value comparisons, for example `familyName` and `name`, along with the utilization of similarity measures, here Jaccard with a threshold value of 0.5, as well as the declaration of an aggregation function which is the average in this example. Other parameters may be considered within these specification.

Additional frameworks are presented and compared in the work of Nentwig et al. [68].

4.2.3 Key-based Approaches

A *key* is a set of properties whose values uniquely identify the instances of a given class. In OWL 2 [78], a key axiom can be declared in an ontology through the construct HasKey:

$$HasKey(CE(OPE_1,...,OPE_{m>0})(DPE_1,...,DPE_{n>0}))$$

which states that each (named) instance of the class expression CE is uniquely identified by the object property expressions OPE_i and the data property expressions DPE_j — that is, no two distinct (named) instances of CE can coincide on the values of all object property expressions OPE_i and all data property expressions DPE_j [78]. An example of a key is $HasKey$(`Person`()(`firstName,name`)) stating that each person is uniquely identified by his first name and his last name.

Although keys were initially not intended for data interlinking, many applications rely on them for this task. For example, given two datasets, such that `firstName` and `name` are properties and `Person` a class in these datasets. If (`Person`()(`firstName,name`)) is declared as a key in both datasets, then: If an instance of `Person` from the first dataset and an instance of the same class in the second dataset have the same `firstName` and the same `name`, then these two instances represent the same entity and an identity link is created between them.

Keys are not necessarily included in datasets, and identifying them by a human expert is not always an easy task. A naive approach to discover them automatically consists in exploring all property combinations and determining whether or not they are keys for a given class. To reduce the search space of keys, many approaches have been proposed. The KD2R algorithm introduced by Pernelle et al. [80], discovers keys from datasets fulfilling the Unique Name Assumption. It

first finds keys in each dataset separately then it merges them to obtain keys that are valid in both datasets. To reduce the search space, KD2R operates on the following principle: to guarantee that a set of properties is a key, all the instances of a given class need to be examined. Conversely, confirming that a set of properties is not a key requires finding two instances with the same values for these properties. Starting from this idea inspired by the Gordian algorithm [88], KD2R computes first a set of maximal non-keys which consist of combinations of properties sharing the same values for at least two instances. Then it computes the set of keys from the set of non-keys.

Because the datasets are usually not complete, KD2R introduces *undetermined keys* which are sets of properties that are not keys and also not non-keys. Indeed a given instance may not have a value for a given property. For example we know the title of a given movie but the director of this movie is not mentioned in the dataset. KD2R interprets the absence of information in two different ways: (i) using a pessimistic heuristic where the undetermined keys will not be considered as keys. (ii) Using an optimistic heuristic where the undetermined keys will be considered as keys. The experiments indicate that the optimistic approach is more efficient and yields keys of superior quality. Conversely, the pessimistic approach employed by KD2R is not suitable for sparse data scenarios. Additionally, when confronted with datasets containing duplicate or erroneous data, no keys can be discovered. In the case of datasets containing a substantial number of properties, the algorithm lacks scalability.

To handle datasets containing duplicated or erroneous data, a new type of keys was proposed. Atencia et al. [11] have introduced *pseudo-keys*, which allows for exceptions, meaning that two or more instances may have the same values for the properties in the key. This tolerance to error is controlled by specific measures and thresholds. Furthermore, in their study, Atencia et al. proposed an algorithm that follows the strategy of the functional dependencies discovery algorithm TANE [53]. The proposed algorithm involves exploring the lattice of the power set of properties, starting from singleton property sets. For each set of properties, it constructs a partition of instances in which each equivalence class comprises instances with the same property value sets. Additionally, the pruning strategy employed by this algorithm entails discarding the super sets of properties if a given property set is identified as a pseudo key, or if its support (i.e., the ratio of instances covered by the partition) is too low, or if it contains a functional dependency. Addressing the same issue, Danai et al. [92] have introduced a novel type of keys with exceptions termed *n-almost keys*. A set of properties is considered as an n-almost key in a given class, when there are, at most, n instances that possess identical values for that particular set of properties. The degree of tolerance to error associated with this type of keys is regulated by the integer value of n. In the same work [92], SAKey has been introduced to discover such keys. It follows the same strategy as KD2R: it first discovers the maximal sets of properties that are not n-almost keys, called *n-non keys*. Then it computes the set of n-almost keys from the set of n-non keys. In Addition, several filtering and pruning strategies are used to improve the efficiency of the algorithm.

In several datasets, it is observed that either no keys or only a limited number of keys are valid across the entire dataset. Based on this observation, Danai et al. [93] have introduced a novel type of keys called *conditional keys*. For instance, within the atomic class `Researcher`, {`lastName`} is not a key. However, in a specific subset of the instances of `Researcher` where the research team is assigned the value "Orpailleur", {`lastName`} becomes a valid key because no two researchers among the members of this particular research team have the same last name. The same work [93] has introduced an algorithm named VICKEY for the purpose of discovering these conditional keys. It first discovers the non-keys using SAKey [92], then it applies a breadth-first strategy to find conditional keys.

A closer look on the way keys are used in data interlinking shows that the two datasets considered for interlinking should rely on the same ontology or an ontology alignment has to be provided. This can be seen as a limitation that can be overcome by using *link keys*.

4.3 Link Keys

First introduced by Atencia et al. [7], *link keys* can be viewed as a generalization of keys for two distinct datasets. A link key has the form of two sets of pairs of properties associated with a pair of classes. For example, let

$$k = (\{(\texttt{designation,title})\}, \{(\texttt{designation,title}), (\texttt{creator,author})\}, (\texttt{Book,Novel}))$$

If k is a link key over two datasets, this states that whenever[20] an instance a of the class `Book` and b of the class `Novel` (class condition) share at least one value for the properties `creator` and `author` ($\exists$-conditions) and that, a and b have the same values for the properties `designation` and `title` ($\forall$-conditions), then a and b denote the same entity. Consequently, $(a, \mathsf{owl{:}sameAs}, b)$ is an identity link over the two datasets. We say that k generates the identity link (a, b).

4.3.1 Syntax

In the following, the syntactic definitions associated with the link keys are provided.

For the sake of simplicity we do not make the distinction between RDF graph and RDF dataset'.

Definition 2 (RDF Dataset)**.** *Let U denote a set of IRIs, B a set of blank nodes, and L a set of literals, i.e., "values".*

An RDF dataset is a set of triples $(s,p,o) \in (U \cup B) \times U \times (U \cup B \cup L)$.

Let D be an RDF dataset. Then:

- $S(D) = \{s \mid \exists p, o \text{ such that } (s,p,o) \in D\}$ denotes the set of individual identifiers (subjects),
- $P(D) = \{p \mid \exists s, o \text{ such that } (s,p,o) \in D\}$ the set of property identifiers,
- $C(D) = \{c \mid \exists s \text{ such that } (s, \texttt{rdf:type}, c) \in D\}$ the set of class identifiers,
- $I(c) = \{s \mid \exists c \text{ such that } (s, \mathsf{rdf{:}type}, c) \in D\}$ denotes the set of instances identifiers of c,
- $p(s) = \{o \mid \exists s, p \text{ such that } (s,p,o) \in D\}$ denotes the set of objects –or values– associated with s through property p.

Example 2. *Two datasets D_1 and D_2 are depicted in Figure 4.3, such that:*

- $P(D_1) = \{\texttt{given}, \texttt{familyName}\}$ *and* $P(D_2) = \{\texttt{firstName}, \texttt{name}\}$ *are the sets of property identifiers.*
- $C(D_1) = \{\texttt{Woman}\}$ *and* $C(D_2) = \{\texttt{FemaleScientist}\}$ *are the sets of class identifiers.*
- $I(\texttt{Woman}) = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$ *and* $I(\texttt{FemaleScientist}) = \{b_3, b_4, b_5, b_6, b_7, b_8\}$ *are the sets of instances identifiers.*

[20] Typically, we follow a left-to-right reading of k, but the order does not hold significance as there is an "and" between the conditions.

Figure 4.3: Example of two RDF datasets. In one hand the dataset D_1 populated with instances of the class `Woman`. In the other hand the dataset D_2 populated with instances of the class `FemaleScientist`.

– *Finally, the "values" of* b_8 *for property* `name` *is* $\texttt{name}(b_8) = \{\texttt{Curie,Skłodowska}\}$.

We start by providing the definition of a *link key expression* which is the syntactic form of a link key.

Definition 3 (Link key expression)**.** *Let* D_1 *and* D_2 *be two RDF datasets,* $k = (Eq, In, (c_1, c_2))$ *is a* link key expression *over* D_1 *and* D_2 *iff* $In \subseteq P(D_1) \times P(D_2)$*,* $Eq \subseteq In$*,* $c_1 \in C(D_1)$ *and* $c_2 \in C(D_2)$*.*

The set of link key expressions is denoted as LKE.

A link key expression is made up of two sets of pairs of properties and a pair of classes. In RDF an instance can have multiple values for a single property. As a result, property values can be compared in different manners: (i) considering whether instances have the same values for a pair of properties. These specific pairs are indicated in the Eq set, referred to as $\forall$-conditions. (ii) Determining whether instances share at least one value for a pair of properties. These specific pairs are given in the In set, known as $\exists$-conditions. For example, in Example 2 $k = (\{(\texttt{familyName,name})\}, \{(\texttt{familyName,name}), (\texttt{given,firstName})\}, (\texttt{Woman,FemaleScientist}))$ is a link key expression over the datasets D_1 and D_2.

In the following, for the sake of clarity, each pair of properties is represented once in a link key expression. This means that if a pair of properties belongs to the set Eq, it will not be included in the set In because Eq is a subset of In. For instance, the given expression k will be written as $k = (\{(\texttt{familyName,name})\}, \{(\texttt{given,firstName})\}, (\texttt{Woman,FemaleScientist}))$.

A link key expression can generate *links* over two datasets. A link is defined as a pair of instances issued from these datasets. It is important to note that a link (a, b) is not always an identity link; that is, a and b may not refer to the same entity. A link has the potential to be an identity link. If a link is an identity link, it is called a "correct link"; otherwise, it is called an "erroneous link". A collection of links, often referred to as the gold standard, can include correct links denoted as L^{ref}. Additionally, erroneous links, denoted as L^-, can be provided, which can be expressed by the triples $(a, \texttt{owl:differentFrom}, b)$ determining that a and b are different. This collection might be unavailable, or only partially available. In Example 2, we define the set of correct links L^{ref} for datasets D_1 and D_2 as pairs of links (a_i, b_j) where $i = j$. Conversely, the set of erroneous links L^- consists of pairs of links (a_i, b_j) where $i \neq j$. Here, the indices i and j are integers, where $1 \leq i \leq 8$ and $3 \leq j \leq 8$.

A *link set* generated by link key expression is defined as follows:

Definition 4 (Link set generated by link key expression)**.** *Let* $k = (Eq, In, (c_1, c_2))$ *be a link key expression over* D_1 *and* D_2*. The link set generated by link key expression* k*, denoted as* $L(k)$*, is the set of links* $(a, b) \in I(c_1) \times I(c_2)$ *satisfying:*

1. $\forall\ (p, q) \in Eq,\ p(a) = q(b)$ *and* $p(a) \neq \emptyset$*,*
2. $\forall\ (p, q) \in In \setminus Eq,\ p(a) \cap q(b) \neq \emptyset$*.*

Link key expressions do not convey semantics. For example, assuming that the classes `Woman` and `Building` are disjoint or incompatible, i.e., they cannot describe the same individuals, $k = (\{(\texttt{name,phoneNumber})\}, \{(\texttt{year,ISBN})\}, (\texttt{Woman,Building}))$ is a link key expression but it does not qualify as a link key because it generates erroneous links.

Link key expressions are partially ordered by $\sqsubseteq$ and they may be combined using meet $\sqcap$ and join $\sqcup$ as follows:

Definition 5 (Subsumption, meet and join of link key expressions)**.** *Given two datasets D_1 and D_2. Let $k = (Eq_k, In_k, (c_1, c_2))$ and $h = (Eq_h, In_h, (c_1, c_2))$ be link key expressions over D_1 and D_2, $k, h \in$ LKE. We say that k* is subsumed by *h, written $k \sqsubseteq h$, if $Eq_k \subseteq Eq_h$ and $In_k \subseteq In_h$. The* meet *and* join *of k and h, denoted as $k \sqcap h$ and $k \sqcup h$, respectively, are defined as follows:*

$$k \sqcap h = (Eq_k \cap Eq_h, In_k \cap In_h, (c_1, c_2))$$
$$k \sqcup h = (Eq_k \cup Eq_h, In_k \cup In_h, (c_1, c_2))$$

Let k_1, k_2 and k_3 link key expressions from Example 2, such that:

$$k_1 = (\{\}, \{\texttt{familyName,name}\}, (\texttt{Woman,FemaleScientist}))$$
$$k_2 = (\{\texttt{given,firstName}\}, \{\}, (\texttt{Woman,FemaleScientist}))$$
$$k_3 = (\{\texttt{given,firstName}\}, \{\texttt{familyName,name}\}, (\texttt{Woman,FemaleScientist}))$$

The link key expressions k_1 and k_2 are subsumed by k_3 and we write, $k_1 \sqsubseteq k_3$ and $k_2 \sqsubseteq k_3$. In addition, $k_1 \sqcap k_3 = k_1$ and $k_1 \sqcup k_2 = k_3$. Notice here that the Eq set of k_1 is empty. The In set of k_2 is not empty and it is equal to its Eq set.

4.3.2 Semantics

Link keys are logical constructors allowing to deduce equality (owl:sameAs) links. The semantics of link keys are defined using the Description Logics interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ [13] as follows:

Definition 6 (Link key)**.** *A* link key *has the form*

$$(Eq, In, (c_1, c_2))$$

where Eq and In are sets of pairs of properties with $Eq \subseteq In$, c_1 and c_2 are classes. A DL interpretation $\mathcal{I}$ satisfies $(Eq, In, (c_1, c_2))$ if, for any $\alpha \in c_1^{\mathcal{I}}$ and $\beta \in c_2^{\mathcal{I}}$,

$$\begin{aligned} &(1)\ p_i^{\mathcal{I}}(\alpha) = q_i^{\mathcal{I}}(\beta) \text{ with } p_i^{\mathcal{I}}(\alpha) \neq \emptyset \text{ for all } p_i, q_i \in Eq, \text{and} \\ &(2)\ p_j^{\mathcal{I}}(\alpha) \cap q_j^{\mathcal{I}}(\beta) \neq \emptyset \text{ for all } p_j, q_j \in In \\ &\qquad\qquad \text{imply } \alpha = \beta. \end{aligned}$$

The semantics of nine types of link keys, namely *weak, plain, and strong link keys*, along with their variants including Eq, In, and $EqIn$ (or hybrid) link keys, have been defined within Description Logics by Atencia et al. [9].

The nine kinds of link keys differ based on two dimensions: (i) whether their underlying properties are keys; and (ii) how they handle multiple values. Let's explore the first dimension through an example. Consider a link key represented as $k = (\{(p_1, q_1), (p_2, q_2)\}, \{\}, (c_1, c_2))$. If k is a *strong* link key, then its underlying properties, i.e., $(\{p_1, p_2\}, c_1)$ and $(\{q_1, q_2\}, c_2)$ keys. On the other hand, if k is categorized as a *weak* link key then its underlying properties are not necessarily keys. In addition, k is a *plain* link key if $\{p_1, p_2\}$ and $\{q_1, q_2\}$ are keys only on the linked instances of c_1 and c_2 respectively.

When it comes to handling multiple values, various variants of link keys have been introduced, namely Eq, In, and $EqIn$ link keys. An Eq link key does not possess the In property set, whereas an In link key does not possess the Eq property set. An $EqIn$, or hybrid link key possesses both Eq and In properties simultaneously.

Combining these two dimensions gives rise to the nine types of link keys described in Table 4.1. For example, $k = (\{(p_1, q_1), (p_2, q_2)\}, (c_1, c_2))$ is a *weak-In-link key* because $(\{p_1, p_2\}, c_1)$ and $(\{q_1, q_2\}, c_2)$ are not necessarily keys, and only In properties are present in k, i.e., when k generates a link (a, b), it implies $p_1(a) \cap q_1(b)) \neq \emptyset$ and $p_2(a) \cap q_2(b) \neq \emptyset$.

	multiple values	**properties are keys**
Weak In-link key	$\exists$	not necessarily
Weak Eq-link key	$\forall$	not necessarily
Weak EqIn-link key	$\forall\exists$	not necessarily
Strong In-link key	$\exists$	yes
Strong Eq-link key	$\forall$	yes
Strong EqIn-link key	$\forall\exists$	yes
Plain In-link key	$\exists$	only on the linked instances
Plain Eq-link key	$\forall$	only on the linked instances
Plain EqIn-link key	$\forall\exists$	only on the linked instances

Table 4.1: Comparison of the different types of link keys in terms of handling multiple values and whether their underlying properties are keys.

The detailed and formal definitions of the semantics for the different types of link keys, along with their properties and relationships with keys, can be found in [9]. The link key defined in Definition 6 corresponds to a weak $EqIn$-link key. In the context of this thesis work, we specifically utilize this type of link key and refer to it as "link key". Additionally, we also make use of its variant, the weak In-link key, which we refer to as "In-link key".

4.3.3 Link Key Discovery

Link keys are not usually provided with the datasets. A naive automatic approach to discover link keys is to examine all possible combinations of properties and classes, i.e., link key expressions. However, to narrow down the search space, link key discovery algorithms [7, 10] focus on discovering a subset of these expressions known as *link key candidates.* These candidates are then evaluated using specific measures to identify the most relevant ones to be designated as link keys. Figure 4.4 illustrates the different stages involved in the process of link key discovery.

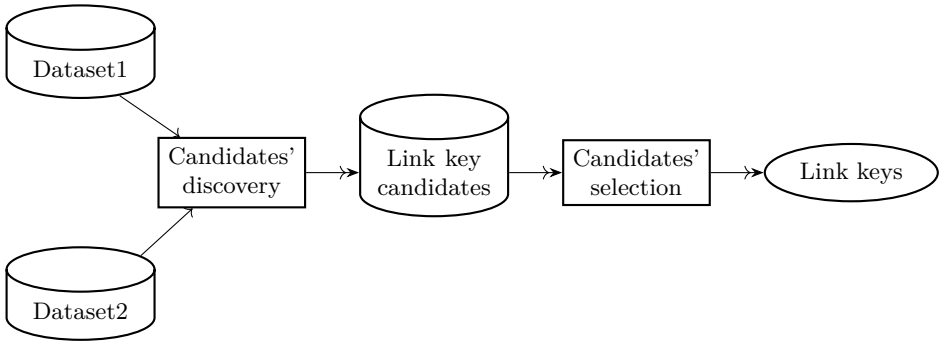


Figure 4.4: The pipeline of link key discovery.

Two definitions of link key candidates can be found in the literature. In the following, we make precise each definition and we describe the related algorithm.

The link key candidate, defined in [7], is referred to as the "base link key candidate" in [10]. This candidate is characterized by two conditions: (i) it generates at least one link, and (ii) if multiple link key expressions generate the same link, only the expression that is maximal among all others is considered as a candidate. Formally:

Definition 7 (Base link key candidate)**.** *Let D_1 and D_2 be two datasets and k a link key expression over these datasets, k is a base link key candidate over D_1 and D_2 if*

- *There exists $l \in L(k)$, and*
- *If h is another link key expression over the same pair of classes as k such that $l \in L(h)$ and $k \sqsubseteq h$ then $k = h$.*

Atencia et al. [7], developed an algorithm which takes as input two datasets and returns a set of base link key candidates. The algorithm operates as follows: first, it builds an index for each dataset. Starting from the set of triples (a, p, v) in the first the RDF dataset, this index associates to each value v, the set of pairs instance-property (a, p), such that, the instance a has the value v for the property p. Another index is created for the other RDF datasets with the triples (b, q, v). After that, using the previous indexes, it creates another index associating each pair of instances (a, b) to the maximal set of pairs of properties (p, q) for which they share a value. Finally, it extracts the set of base candidates from this last index.

While [7] considered as a candidate the expression that is maximal on one link, [10] considered as a candidate the expression that is maximal on a link set. These candidates are defined as follows:

Definition 8 (Link key candidate)**.** *Let D_1 and D_2 be two datasets and k a link key expression over these datasets, k is a* link key candidate *over D_1 and D_2 if*

- *$L(k) \neq \emptyset$, and*
- *If h is another link key expression over the same pair of classes as k such that $L(h) = L(k)$ and $k \sqsubseteq h$ then $k = h$.*

The set of link key candidates over two datasets D_1 and D_2 is denoted as LKC. Let k_1, k_2 and k_3 link key expressions from Example 2 generating the same link set $L(k_1) = L(k_2) = L(k_3) = \{(a_3, b_3), (a_4, b_4), (a_5, b_5), (a_6, b_6), (a_7, b_7), (a_8, b_8)\}$, such that:

$$k_1 = (\{\}, \{\texttt{familyName,name}\}, (\texttt{Woman,FemaleScientist}))$$
$$k_2 = (\{\texttt{given,firstName}\}, \{\}, (\texttt{Woman,FemaleScientist}))$$
$$k_3 = (\{\texttt{given,firstName}\}, \{\texttt{familyName,name}\}, (\texttt{Woman,FemaleScientist}))$$

The link key expression k_3 is maximal on the link set it generates because $k_1 \sqsubseteq k_3$ and $k_2 \sqsubseteq k_3$ then k_3 is a link key candidate while k_1 and $k2$ are not.

Here, the set of link key expressions is quotiented by the link sets they generate. This forms a partition of the link key expressions. Link key candidates are the maximal elements of the equivalence classes of this partition which matches the definition of a closed set. Consequently, [8, 10] proposed the utilization of Formal Concept Analysis for the discovery of link key candidates.

This approach takes two RDF datasets as input and computes the corresponding formal context. In this formal context, the objects represent the set of pairs of instances from both datasets, while the attributes represent the set of pairs of properties from the two RDF datasets.

Formally, given two datasets D_1 and D_2 and a pair of classes $(c_1, c_2) \in C(D_1) \times C(D_2)$. The *LK-formal context* or the *formal context for link key candidates* over the pair of classes (c_1, c_2) is the triple $(I(c_1) \times I(c_2), \{\exists, \forall\} \times P(D_1) \times P(D_2), \mathcal{I})$ such that:

- The set of objects of the LK-formal context is the set of pairs of instances, i.e., the link $(a, b) \in I(c_1) \times I(c_2)$.
- The set of attributes of the LK-formal context is the set of pairs of properties $(p, q) \in P(D_1) \times P(D_2)$ preceded by a quantifier in $\{\exists, \forall\}$, i.e., $\forall(p, q)$ and $\exists(p, q)$.
- The relation $\mathcal{I}$ between an object and an attribute is defined as follows:

$$(a, b)\ \mathcal{I}\ \forall(p, q) \quad \text{iff } p(a) = q(b) \neq \emptyset$$
$$(a, b)\ \mathcal{I}\ \exists(p, q) \quad \text{iff } p(a) \cap q(b) \neq \emptyset$$

	$\forall(familyName, name)$	$\forall(given, firstName)$	$\exists(familyName, name)$	$\exists(given, firstName)$	$\exists(given, name)$
(a_1, b_3)					×
(a_2, b_3)				×	
(a_3, b_3)		×	×	×	
(a_4, b_4)	×	×	×	×	
(a_5, b_5)	×	×	×	×	
(a_6, b_6)	×	×	×	×	
(a_7, b_7)	×	×	×	×	
(a_8, b_8)		×	×	×	

Table 4.2: The LK-formal context for link key discovery over the pair classes (`Woman`,`FemaleScientist`) depicted in Figure 4.3

.

Table 4.2 represents the LK-formal context for link key discovery over the pair of classes (`Woman`,`FemaleScientist`) depicted in Figure 4.3. The objects of this formal context are the set of pairs of instances, i.e., links $(a, b) \in (I(\texttt{Woman}) \times I(\texttt{FemaleScientist}))$, the attributes are the pairs of properties $(p, q) \in P(D_1) \times P(D_2)$ preceded by the quantifiers $\exists$ or $\forall$. The link (a_1, b_3) is related to the pair of properties $\exists$`(given,name)` because $\texttt{given}(a_1) \cap \texttt{name}(b_3) = \{\texttt{Murray}\}$. While the link (a_3, b_3) is related to the pair of properties $\forall$`(given,firstName)` because $\texttt{given}(a_3) = \texttt{firstName}(b_3) = \{\texttt{Grace}\}$.

The formal concepts of the lattice or the LK-lattice, generated from the LK-formal context, correspond to the pairs $Ck = (L(k), k)$, where k, the intent of this formal concept, is a link key candidate and $L(k)$, the extent of this formal concept, is the link set generated by the candidate k. Figure 4.5 illustrates the LK-lattice derived from the LK-formal context presented in Table

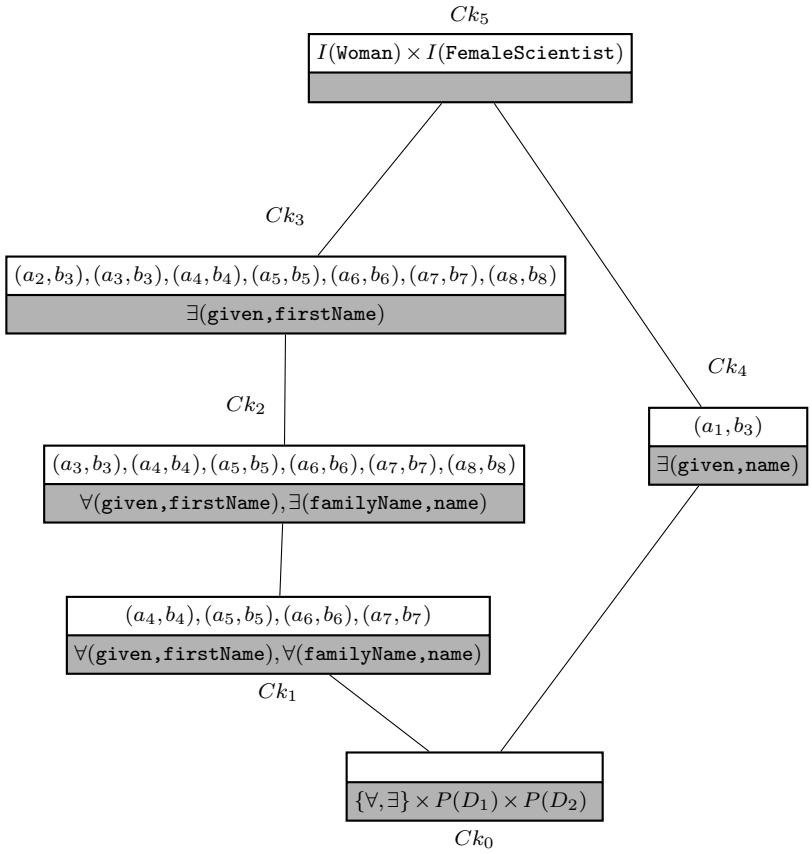


Figure 4.5: The lattice of the LK-formal context in Table 4.2. For example, the formal concept Ck_2 has as intent the link key candidate $k_2 = (\{(\texttt{given,firstName})\},\{(\texttt{familyName,name})\},(\texttt{Woman,FemaleScientist}))$. The candidate k_2 generates the link set $L(k_2)$ which is the extent of Ck_2, $L(k_2) = \{(a_3,b_3),(a_4,b_4),(a_5,b_5),(a_6,b_6),(a_7,b_7),(a_8,b_8)\}$.

4.2. Each formal concept, denoted as $Ck_i = (k_i, L(k_i))$, consists of a link key candidate k_i, which represents the intent of Ck_i, and its corresponding link set $L(k_i)$, which represents the extent of Ck_i.

To interpret the intents of this lattice as link key candidates $k = (Eq, In, (c_1, c_2))$, the pairs of properties preceded by $\forall$ belong to the sets Eq and In. The pairs of properties preceded by $\exists$ belong to the In set. The classes are provided as input.

For instance, consider the formal concept Ck_2, which has the link key candidate $k_2 = (\{(\texttt{given,firstName})\},\{(\texttt{familyName,name})\},(\texttt{Woman,FemaleScientist}))$. The candidate k_2 generates the link set $L(k_2) = \{(a_3,b_3),(a_4,b_4),(a_5,b_5),(a_6,b_6),(a_7,b_7),(a_8,b_8)\}$, which is the extent of Ck_2.

Quality of link key candidates

As for link key expressions, link key candidates do not convey semantics. Indeed, not all link key candidates are eligible to be considered as "valid" link keys. Some candidates may include properties that are too general. A candidate such as $(\{\},\{(\texttt{country,country})\},(\texttt{Person,Human}))$ would state that two individuals sharing the same country represent the same person. Further-

more, some link key candidates may incorporate pairs of properties that are coincidental, leading to the generation of erroneous owl:sameAs links. For instance, $(\{\}, (\texttt{name},\texttt{title}), (\texttt{Person},\texttt{Book}))$ is an example of a candidate that erroneously identifies persons with books based on the presence of shared values in the `name` and `title` properties.

To evaluate the quality of link key candidates, Atencia et al. [7] proposed two criteria: the *correctness* and *completeness* of the generated links. Ideally, a link key candidate should exhibit both correctness and completeness. Correctness refers to the ability of a link key candidate to produce only correct links, while completeness pertains to its capacity to identify all the correct links. However, it is uncommon for link key candidates to be entirely correct and complete. Consequently, specific measures are required to assess these qualities under supervised and unsupervised scenarios [7].

In the supervised setting, i.e., when a set of reference links L^{ref} is available, the correctness and completeness of a link key candidate k are evaluated using precision and recall, respectively. The precision of k is given by the proportion of correctly identified links by k to the total number of links identified by k:

$$pre(k) = \frac{|L(k) \cap L^{ref}|}{|L(k)|}$$

The recall of candidate k is calculated as the ratio of the number of correctly identified links by k to the total number of reference links in L^{ref}:

$$rec(k) = \frac{|L(k) \cap L^{ref}|}{|L^{ref}|}$$

The overall quality of candidate k is assessed using the F-measure, which takes into account both precision and recall. Specifically, the F-measure of k is defined as:

$$F\text{-}measure(k) = 2 \,.\frac{rec(k).pre(k)}{rec(k) + pre(k)}.$$

In the unsupervised scenario, where the reference links are not available, it becomes essential to develop measures that capture a link key candidate's precision and recall rankings. To assess the completeness and correctness of a candidate, two quality measures, namely "coverage" and "discriminability," were introduced in [7]. These measures are based on the link set generated by a link key candidate.

Let $L \subseteq I(c_1) \times I(c_2)$ be a set of links. Let us consider $\pi_1(L) = \{a|(a,b) \in L\}$ and $\pi_2(L) = \{b|(a,b) \in L\}$. Then the coverage of $k = (Eq, In, (c_1, c_2))$ is given by:

$$co(k) = \frac{|\pi_1(L(k)) \cup \pi_2(L(k))|}{|I(c_1) \cup I(c_2)|}$$

The maximum coverage is achieved when all instances within the considered classes are linked to at least one other instance.

The discriminability of k is given by:

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

Maximum discriminability occurs when instances are linked to only one other instance. This measure indeed assume the Unique Name Assumption UNA for the two datasets independently.

Similarly to the F-measure, discriminability and coverage can be combined using a harmonic mean:

$$hm(co(k), di(k)) = 2 \,.\frac{co(k).di(L)}{co(k) + di(k)}$$

Combination of link keys

Using a combination of link keys is better than using one link key to find identity links among datasets. Thus, [12] studied the combination of link keys via conjunction and disjunction. They first generalized the definition of a link set generated by a link key expression to link sets generated by the conjunction of expressions as well as disjunction expressions as follows. Given the link key expressions k and h over the datasets D_1 and D_2, the *link sets generated by* $k \sqcap h$ and $k \sqcup h$ are subsets of $S(D_1) \times S(D_2)$ defined by

$$L(k \sqcap h) = L(k) \cap L(h)$$
$$L(k \sqcup h) = L(k) \cup L(h)$$

Given a lattice of link key candidates, Atencia et al. [12] proposed to find relevant combinations of link key candidates w.r.t. a given quality measures. First of all, they demonstrated that there is a link key candidate in the lattice that generates the link set generated by any conjunction of candidates in this lattice. Thus, there is no need to find conjunctions of link key candidates. Whereas, the link set generated by a given disjunction of candidates may not be generated by any link key candidate in the lattice. Therefore, it is crucial to identify these disjunctions. To do so, they first find antichains (non-comparable candidates) in the LK-lattice.

The next step is to select the relevant disjunctions (antichains) of link key candidates. Evaluating each antichain can be computationally expensive because there can be, in the worst case, 2^n antichains in a given lattice of link key candidates with n is the number of its candidates [47]. Thus, given a lattice of link key candidates, two strategies have been proposed to select the best antichains of link key candidates: (i) The top-k strategy, relies on the assumption that the best antichains are those that contain only the best link key candidates. Therefore this strategy selects the top-k candidates w.r.t. the link key quality measure. Then, it enumerates all the antichains made up of the selected candidates. (ii) The expand-best strategy assumes that the better an antichain is, the more likely it is to generate antichains that are even better by adding candidates to it. The process starts with a set of antichains A of size one, i.e. single link key candidates, then selects the best antichain w.r.t. the quality measures. After that, it expands this antichain by adding single candidates to it. Finally, the strategy adds the antichains obtained by expansion to the first set of antichains A while deleting the explored one from A. The process iterates by expanding again the best antichain in A and ends after a given number of iterations without achieving any improvement in antichains quality.

Experimental evaluation showed that generally, disjunctions of link key candidates are better than single link key candidates. More precisely, the disjunctions selected by the top-k strategy improve the F-measure of single candidates. The expand-best strategy has better recall than top-k because it selects longer antichains. While top-k (with k=10) scales better than expand-best.

Dependent link key candidate

Up to this point, the existing methods for link key discovery have exclusively handled datatype properties, which are properties whose values are literals. Nonetheless, situations may arise where the ranges of properties themselves are instances of other classes. These properties are known as object properties or relations.

Comparing the values of datatype properties involves comparing two strings. However, to determine if the values of the object properties in two datasets are equal, it becomes necessary to determine if they represent the same entity. This is precisely the aim of link keys. For

example, let the candidate $k = (\{(\texttt{designation,title})\},\{(\texttt{creator,author})\},(\texttt{Book,Novel}))$, and the value of the property `creator` is not a literal but an IRI e which is an instance of a class `Scientist`. Similarly, the value of the property `author` is the IRI t which is an instance of a class `FemaleScientist`. We need to establish whether the instances e and t refer to the same entity. To accomplish this, we must discover a link key over the classes `FemaleScientist` and `Scientist`. Consequently, the link key k is considered as a *dependent link key candidate*.

Atencia et al. [10] introduced a method for identifying dependent link key candidates using Relational Concept Analysis (RCA) [84]. RCA is an extension of FCA tailored to manage relational datasets containing formal contexts and relationships between the objects in these contexts.

Reasoning with link keys

Link keys, as logical axioms, can be integrated with ontologies and ontology alignments to enable logical reasoning. The detection of inconsistent link keys plays a crucial role in the discovery of link keys by enabling the elimination of link key candidates that generate erroneous links. The process of verifying the consistency of link keys involves examining whether the satisfaction of a link key contradicts the logical axioms defined in the ontologies or ontology alignments. Jradeh's work [54] focused on studying the problem of reasoning with link keys. Since directly modeling link keys within Description Logics is not feasible, link keys and individual equalities were incorporated into the $\mathcal{ALC}$ DLs [13], resulting in a new variant called $\mathcal{ALC}+\mathcal{LK}$. Additionally, this work introduced an algorithm for determining the consistency of the $\mathcal{ALC}+\mathcal{LK}$ ontology.

4.4 Discussion

To provide context for our work, this chapter introduced various approaches that are employed to perform the data interlinking task.

Embedding approaches capture complex relationships between individuals that may not be easily expressed through link keys. However, they can also be computationally intensive and require input entity alignments, which is not the case for link keys. Furthermore, many approaches in this category enable the interlinking of datasets from different languages, which is not possible using link keys as they rely on value comparison unless an external translator is employed. On the other hand, link keys offer a more explicit and unambiguous means of interlinking individuals compared to embedding approaches, which generate links without explicitly specifying the reasons behind them. Link keys can be particularly advantageous because they operate as rules, directly declaring whether new individuals added to the datasets are the same or not. In contrast, embedding approaches are not efficient in this scenario unless the entire embedding algorithm is executed for each new individual. Furthermore, one of the strengths of link keys lies in their utilization as logical axioms in ontologies, which involves dealing with semantics. This advantage is not applicable to embedding approaches, which do not incorporate the semantic reasoning.

Data interlinking frameworks such as SILK [94] and LIMES [76] can consider link keys as a link specifications. Consequently, link keys offer valuable assets and supplements by providing the properties and classes that need to be compared. These properties and classes support link specification by incorporating similarity measures and thresholds. On the other hand, link keys can also benefit from these frameworks. They not only offer equality comparisons between values but also allow for relaxation to similarity, which can result in more generated links compared to using a link key alone. Additionally, the scalability strategies employed in these frameworks

can be advantageous for generating links using link keys, especially when dealing with large datasets.

To use keys for the data interlinking tasks, the two datasets being considered must be based on the same ontology, or an ontology alignment needs to be provided. This requirement can be viewed as a limitation, which can be addressed by employing link keys. Indeed, link keys offer the advantage of being applicable in data interlinking scenarios involving different ontologies or when no ontology alignment is available. This flexibility allows link keys to be employed in a wider range of situations, overcoming the limitations associated with keys.

Overall, there is no intrinsic superiority of one type of data interlinking approach to another. They rather have to be used in a complementary way since the effectiveness of different methods may depend on the particular datasets.

Building upon the link key discovery algorithm based on FCA discussed in this chapter, we now propose an extension to address several limitations encountered in various stages of the discovery process. Our proposed contributions focus on discovering more expressive link key candidates, implementing a more effective selection strategy, and detecting redundant link key candidates.

Chapter 5

Pattern Structures for Link Key Candidate Discovery

Contents

5.1 Motivation **46**
5.2 Extending the Definition of Link Key Candidates **48**
5.3 A Pattern structure for link Key discovery **50**
5.4 Conclusion and Discussion **52**

This chapter is based on the paper published in:

Nacira Abbas, Jérôme David, and Amedeo Napoli. Discovery of Link Keys in RDF Data Based on Pattern Structures: Preliminary Steps. In Proceedings of International Conference on Concept Lattices and Their Applications, CLA, CEUR Workshop Proceedings 2668, pages 235–246, 2020 [5].

5.1 Motivation

The question of using Formal Concept Analysis to discover link keys has arisen naturally, since the definition of a link key candidate matches the definition of a formal concept in FCA [8, 10].

One of the strengths of the existing methods [7,10] for link key discovery is that they attempt to minimize a priori knowledge. For example, they rely solely on the Abox without considering the domain and range of a given property expressed by the `rdfs:domain` and `rdfs:range` axioms.

These methods employ one of the following strategies to discover link key candidates. The first strategy involves taking all subjects from two datasets as input, ignoring the classes to which the subjects belong. The result is a set of candidates of the form $k = (Eq, In)$. This strategy, however, has several drawbacks:

– The evaluation of these candidates considers the entire datasets as a whole, without taking into account the specific pair of classes. This evaluation lacks accuracy since a link key candidate may be relevant for one pair of classes, e.g., (`Book`,`Dictionary`), but not relevant for another pair, e.g., (`Scientist`,`FemaleScientist`).

– In order to fully leverage their capabilities, reasoning algorithms necessitate the explicit specification of class pairs to which the link key is applicable.

The second strategy that the existing algorithms employ, consists in finding link key candidates for one particular pair of classes, then evaluating these candidates w.r.t. this particular pair. This strategy iterates the process for all pairs of classes derived from the two datasets, enabling a more precise evaluation of the candidates. However, determining which classes to consider as input is not known in advance. Consequently, a naive approach would involve considering all pairs of classes from the Cartesian product of the sets of classes in the provided datasets, or requiring class alignment as a prerequisite. The first solution is computationally expensive, and the second one is not always feasible because the class alignment may not be provided.

In addition, the Web of data allows for the representation of diverse perspectives on entities. This leads to potentially different classifications of the same entity across datasets. For instance, in one dataset, `Marie Curie` is an instance of two classes, `Woman` and `Scientist`, while in another dataset, `Marie Curie` is an instance of the class `FemaleScientist`. For example, a potential link key candidate could be associated with the pair (`Woman and Scientist`,`FemaleScientist`), where `Woman and Scientist` represents a class expression. The existing link key discovery methods [7, 10] do not enable the discovery of such link keys.

In the following, we present an approach [5] based on Pattern Structures that addresses the aforementioned limitations. This method enables the discovery of link key candidates in a single pass, thereby eliminating the need for iterative processing of every pair of classes. Moreover, it allows for explicit specification of the pairs of classes associated with each link key candidate without requiring prior alignment. Furthermore, our proposed approach takes into account the disparity in abstraction between datasets by extending the concept of a link key candidates, traditionally associated to a pair of atomic classes, to a link key candidate associated with a pair of classes.

To illustrate the one pass strategy, we examine Figure 5.1, which depicts two RDF datasets. Each dataset comprises individuals from three distinct classes. Dataset D_1 consists of the classes $C(D_1) = \{$`Woman, Scientist, Book`$\}$, while dataset D_2 contains the classes $C(D_2) = \{$`FemaleScientist, Dictionary, Novel`$\}$.

The *LK-formal context* over the datasets D_1 and D_2, is the triple $(S(D_1) \times S(D_2), \{\exists, \forall\} \times P(D_1) \times P(D_2), \mathcal{I})$ represented in Table 5.1. The objects here can be considered as pairs of instances belonging to the owl:Thing class containing all individuals.

Figure 5.2 depicts the lattice of the LK-formal context represented in Table 5.1.

The existing methods for link key discovery select the link key candidates having the highest score for the harmonic mean, of coverage and discriminability, to be "valid" link keys. Table 5.2 provides the scores of link key candidates from the lattice in Figure 5.2 for these measures. In this example we may choose k_7, k_8 or k_9 to be link keys because they have the highest hm. However, even if these candidates are relevant for the pair (`Woman,FemaleScientist`), they do not generate any links for the pair (`Book,Dictionary`). In fact for this latter pair, the link key candidate k_2 is more relevant, because it generates only and all correct links, even if it has a low hm. For the pair (`Book,Novel`) , the candidate k_4 is more relevant. Despite their relevance, these link key candidates exhibit a low hm because their evaluation considers the entire datasets instead of specific classes. To enable a more precise evaluation, we introduce a Pattern Structure in the following, which explicitly specifies the pairs of classes associated with link keys. To accomplish this, we need to generalize the definition of link key expressions.

	∀(given,firstName)	∀(familyName,name)	∀(creator,author)	∀(date,year)	∀(designation,title)	∃(given,firstName)	∃(familyName,name)	∃(creator,author)	∃(date,year)	∃(designation,title)
(a_2,b_3)	×					×				
(a_3,b_3)	×	×				×	×			
(a_4,b_4)	×	×				×	×			
(a_5,b_5)	×	×				×	×			
(a_6,b_6)	×	×				×	×			
(a_7,b_7)	×	×				×	×			
(a_8,b_8)	×	×				×	×			
(a_9,b_8)		×					×			
(a_{11},b_{11})				×	×				×	×
(a_{11},b_{12})					×					×
(a_{12},b_{11})					×					×
(a_{12},b_{12})				×	×				×	×
(a_{13},b_{13})			×		×			×		×
(a_{13},b_{14})			×					×		
(a_{14},b_{13})			×					×		
(a_{14},b_{14})			×		×			×		×
(a_{15},b_{15})					×			×		×

Table 5.1: The *LK-formal context* over the datasets represented in Figure5.1.

5.2 Extending the Definition of Link Key Candidates

We generalize the notion of a link key candidate, as defined by Atencia et al. [10], from being associated with a pair of atomic classes to being associated with a pair of classes. We limit ourselves to a subset of classes derived from Description Logics [13]. This subset comprises atomic classes connected by conjunction and disjunction operators. Given an RDF dataset D, an atomic class c in D, i.e., $c \in C(D)$, and $\sqcap_{dl}$, $\sqcup_{dl}$ being respectively the conjunction, disjunction operators as they are defined in Description Logics. The set of classes of D denoted as $CE(D)$ is built as follows:

- If c is an atomic class in D, then c is a class in D, i.e., $c \in CE(D)$.
- If c_1 and c_2 are a classes in D, then $c_1 \sqcap_{dl} c_2$ is a class in D, i.e., $c_1 \sqcap_{dl} c_2 \in CE(D)$.
- If c_1 and c_2 are a classes in D, then $c_1 \sqcup_{dl} c_2$ is a class in D, i.e, $c_1 \sqcup_{dl} c_2 \in CE(D)$.

Following the semantics of the operators in Description Logics, the set of instances belonging to a class is built as follows:

- If c_1 and c_2 are atomic classes in D, then the set of instances of $c_1 \sqcap_{dl} c_2 \in CE(D)$ is $I(c_1 \sqcap_{dl} c_2) = I(c_1) \cap I(c_2)$, i.e., the intersection of the set of instances of c_1 and the set of instances of c_2.
- If c_1 and c_2 are atomic classes in D, then the set of instances of $c_1 \sqcup_{dl} c_2 \in CE(D)$ is $I(c_1 \sqcup_{dl} c_2) = I(c_1) \cup I(c_2)$, i.e., the union of the set of instances of c_1 and the set of instances of c_2.

	Coverage *co*	**Discriminability** *di*	**Harmonic mean** *hm*
k_1	0.384	0.714	0.499
k_2	0.153	1	0.265
k_3	0.153	0.600	0.243
k_4	0.230	1	0.373
k_5	0.153	0.5	0.234
k_6	0.153	1	0.265
k_7	0.500	0.857	0.631
k_8	0.500	0.857	0.631
k_9	0.461	1	0.631

Table 5.2: Evaluation of link key candidates from Figure 5.2.

– If c_1 and c_2 are classes in D, then the set of instances of $c_1 \sqcap_{dl} c_2 \in CE(D)$ is $I(c_1 \sqcap_{dl} c_2) = I(c_1) \cap I(c_2)$.

– If c_1 and c_2 are classes in D, then the set of instances of $c_1 \sqcup_{dl} c_2 \in CE(D)$ is $I(c_1 \sqcup_{dl} c_2) = I(c_1) \cup I(c_2)$.

After establishing the specific classes under consideration and identifying the subjects that belong to these classes, we substitute the pair of atomic classes in a link key expression, as defined in [10], with a pair of classes. This modification allows for a more flexible and expressive representation of link keys. An example of such a link key expression over the datasets depicted in Figure 5.1 is $k = (\{(\texttt{given,year})\}, \{\}, (\texttt{Woman} \sqcap_{dl} \texttt{Book,FemaleScientist}))$. In light of the aforementioned observations, link key expressions do not covey semantics. This is precisely the reason why the class expression $\texttt{Woman} \sqcap_{dl} \texttt{Book}$ appears in k, even though there is no instance that belongs to both classes.

A link key expression associated with a pair of class expressions may generate links among these classes. For example, let k be a link key expression over the datasets depicted in Figure 5.1, such that $k = (\{\}, \{(\texttt{given,firstName})\}, (\texttt{Woman} \sqcap_{dl} \texttt{Scientist,FemaleScientist}))$. The link $(a_3, b_3) \in L(k)$ because $a_3 \in I(\texttt{Woman})$ and also $a_3 \in I(\texttt{Scientist})$. Moreover, the instance $b_3 \in I(\texttt{FemaleScientist})$, and $\texttt{given}(a_3) \cap \texttt{firstName}(b_3) = \{\texttt{Grace}\}$.

Considering our extended definition of link key expressions, it is imperative to further expand the definition of the meet and join operations, as originally introduced by Atencia et al. [10]. In this context, we no longer impose the restriction that link key expressions must be defined on the same pairs of classes. Furthermore, the operators we define here go beyond simple intersections and unions between sets of pairs of properties, as they encompass conjunction and disjunction between the associated classes as well.

Definition 9 (Meet, join of link key expressions associated with a pair of classes)**.** *Given two datasets D_1 and D_2. Let $k_1 = (Eq_1, In_1, (CE_1^1, CE_2^1))$ and $k_2 = (Eq_2, In_2, (CE_1^2, CE_2^2))$ be link key expressions over D_1 and D_2. The meet $\sqcap$ and the join $\sqcup$ of k_1 and k_2 are defined as follows:*

$$k_1 \sqcap k_2 = (Eq_1 \cap Eq_2, In_1 \cap In_2, ((CE_1^1 \sqcup_{dl} CE_1^2), (CE_2^1 \sqcup_{dl} CE_2^2)))$$
$$k_1 \sqcup k_2 = (Eq_1 \cup Eq_2, In_1 \cup In_2, ((CE_1^1 \sqcap_{dl} CE_1^2), (CE_2^1 \sqcap_{dl} CE_2^2)))$$

As the number of property pairs to compare in a link key expression decreases, the number of subject pairs satisfying these property pairs increases. Consequently, the classes in a link key expression become larger,and dually for $k_1 \sqcup k_2$.

As an example from the datasets represented in Figure 5.1, the meet and the join of two link key expressions k_1 and k_2:

$k_1 = (\{(\texttt{given,firstName})\}, \{\}, (\texttt{Woman,FemaleScientist}))$
$k_2 = (\{(\texttt{given,firstName})\}, \{(\texttt{name,familyName})\}, ((\texttt{Woman} \sqcap_{dl} \texttt{Scientist}), \texttt{FemaleScientist}))$
$k_1 \sqcap k_2 = (\{(\texttt{given,firstName})\}, \{\}, (\texttt{Woman,FemaleScientist}))$
$k_1 \sqcup k_2 = (\{(\texttt{given,firstName})\}, \{(\texttt{name,familyName})\}, ((\texttt{Woman} \sqcap_{dl} \texttt{Scientist}), \texttt{FemaleScientist}))$

The definition of link key candidate associated with a pair of classes, follows now the maximality as given in Definition 9.

5.3 A Pattern structure for link Key discovery

In the following, we propose a method based on Pattern Structures that, given two datasets, discovers link key candidates in one pass (without iterating on every pair of classes) while explicitly specifying the classes associated with each link key candidate. We define the pattern structure for link key candidates discovery where the set of objects is the set of pairs of subjects issued from two datasets. In the Pattern Structure, the description of a potential link is given by the maximal link key expression that generates this link. Then the meet of two descriptions corresponds to the meet of link key expressions as introduced in Definition 9.

Definition 10 (Pattern structure for link key candidate discovery)**.** *Given two datasets D_1 and D_2. The pattern structure for link key candidate discovery between D_1 and D_2, called hereafter the LK-*pattern structure*, is the triple $(S(D_1) \times S(D_2), (E, \sqcap), \delta)$ where:*

- *The set of objects $S(D_1) \times S(D_2)$ is the set of pairs of subjects over D_1 and D_2.*
- *E is the set of potential object descriptions. A description is a link key expression $k = (Eq, In, (CE_1, CE_2))$ over D_1 and D_2.*
- *$(E, \sqcap)$ is a meet semilattice where the meet $\sqcap$ of two link key expressions given in Definition 9. The descriptions are partially ordered by $\sqsubseteq$ defined w.r.t. the similarity operator $\sqcap$. If $k_1 \sqcap k_2 = k_1 \Leftrightarrow k_1 \sqsubseteq k_2$.*
- *The mapping $\delta : S(D_1) \times S(D_2) \rightarrow E$ associates each pair of subjects $(s_1, s_2) \in S(D_1) \times S(D_2)$ to its description $\delta((s_1, s_2)) = (Eq, In, (CE_1, CE_2))$ where, $Eq = \{(p_1, p_2) | p_1(s_1) = p_2(s_2) \neq \emptyset\}$, $In = \{(p_1, p_2) | p_1(s_1) \cap p_2(s_2) \neq \emptyset\}$, CE_1 (resp. CE_2) is the conjunction of the classes of s_1 (resp. s_2) over $C(D_1)$ (resp. $C(D_2)$).*

The $LK-$pattern structure for the datasets in Figure 5.1 is given in Table 5.3. For example, the description of the pair $(a_8, b_8) \in S(D_1) \times S(D_2)$ as follows:

$\delta((a_8, b_8)) = (\{(\texttt{given,firstName}), (\texttt{familyName,name})\}, \{\}, ((\texttt{Woman} \sqcap_{dl} \texttt{Scientist}), \texttt{FemaleScientist}))$

The meet of the two descriptions k_1 and $k_2$is given as follows:

$k_1 = (\{(\texttt{given,firstName}), (\texttt{familyName,name})\}, \{\}, ((\texttt{Woman} \sqcap_{dl} \texttt{Scientist}), \texttt{FemaleScientist}))$
$k_2 = (\{(\texttt{given,firstName})\}, \{\}, (\texttt{Woman,FemaleScientist}))$
$k_1 \sqcap k_2 = (\{(\texttt{given,firstName})\}, \{\}, (\texttt{Woman,FemaleScientist}))$
$k_1 \sqcap k_2 = k_2$, hence, $k_2 \sqsubseteq k_1$

The derivation operators $.^{\square}$ form a Galois connection between $2^{S(D_1) \times S(D_2)}$ and E and defined as follows:

Objects	Descriptions		
$S(D_1) \times S(D_2)$	Eq	In	(CE_1, CE_2)
(a_2, b_3)	{(given,firstName)}	{(given,firstName)}	(Woman, FemaleScientist)
(a_3, b_3)	{(given,firstName), (familyName,name)}	{(given,firstName), (familyName,name)}	((Woman⊓$_{dl}$Scientist), FemaleScientist)
(a_4, b_4)	{(given,firstName), (familyName,name)}	{(given,firstName), (familyName,name)}	((Woman⊓$_{dl}$Scientist), FemaleScientist)
(a_5, b_5)	{(given,firstName), (familyName,name)}	{(given,firstName), (familyName,name)}	((Woman⊓$_{dl}$Scientist), FemaleScientist)
(a_6, b_6)	{(given,firstName), (familyName,name)}	{(given,firstName), (familyName,name)}	((Woman⊓$_{dl}$Scientist), FemaleScientist)
(a_7, b_7)	{(given,firstName), (familyName,name)}	{(given,firstName), (familyName,name)}	((Woman⊓$_{dl}$Scientist), FemaleScientist)
(a_8, b_8)	{(given,firstName), (familyName,name)}	{(given,firstName), (familyName,name)}	((Woman⊓$_{dl}$Scientist), FemaleScientist)
(a_9, b_8)	{(familyName,name)}	{(familyName,name)}	(Scientist, FemaleScientist)
(a_{11}, b_{11})	{(date,year), (designation,title)}	{(date,year), (designation,title)}	(Book,Dictionary)
(a_{11}, b_{12})	{(designation,title)}	{(designation,title)}	(Book,Dictionary)
(a_{12}, b_{11})	{(designation,title)}	{(designation,title)}	(Book,Dictionary)
(a_{12}, b_{12})	{(date,year), (designation,title)}	{(date,year), (designation,title)}	(Book,Dictionary)
(a_{13}, b_{13})	{(creator,author), (designation,title)}	{(creator,author), (designation,title)}	(Book,Novel)
(a_{13}, b_{14})	{(creator,author)}	{(creator,author)}	(Book,Novel)
(a_{14}, b_{13})	{(creator,author)}	{(creator,author)}	(Book,Novel)
(a_{14}, b_{14})	{(creator,author), (designation,title)}	{(creator,author), (designation,title)}	(Book,Novel)
(a_{15}, b_{15})	{(designation,title)}	{(creator,author), (designation,title)}	(Book,Novel)

Table 5.3: $LK-$pattern structure for the datasets given in Figure 5.1

$$L^{\square} = \prod_{(s_1,s_2)\in L} \delta((s_1,s_2)) \qquad L \subseteq S(D_1) \times S(D_2)$$
$$k^{\square} = \{(s_1,s_2) \in S(D_1) \times S(D_2) | k \sqsubseteq \delta((s_1,s_2))\} \qquad k \in E$$

L is a closed set if $L^{\square\square} = L$ and k is a closed set if $k^{\square\square} = k$. Then a pattern concept verifies $L^{\square} = k$ and $k^{\square} = L$. The link key expression k is a link key candidate for the two datasets D_1 and D_2 if and only if (L, k) is a pattern concept of the $LK-$pattern structure for D_1 and D_2.

It should be noticed that the link key candidate k_9 in Figure 5.3 corresponds to the link key candidate k_9 in Figure 5.2 (calculated with plain FCA). In the pattern concept lattice we can see that the link key k_9 is associated with the pair ((Woman⊓$_{dl}$Scientist),FemaleScientist), whereas this was not possible using plain FCA (see [10]). Moreover, different link key candidates may have the same sets of pairs of properties (Eq and In) but are associated with different pairs of classes. For example, k_{1a} and k_{1b}, which correspond to the link key candidate k_1 in Figure 5.2, have the same sets of pairs of properties {(designation,title)},{(designation,title)}, but they are associated with different classes: k_{1a} with (Book,(Dictionary⊔$_{dl}$Novel)) and k_{1b} is associated with (Book,Novel). The properties appearing in such link key candidates are used to describe subjects belonging to different classes. For example, the property title appearing in k_{1a} and k_{1b} is used to describe the instances of the classes Dictionary and Novel. Furthermore, we may notice that one pair of classes can be associated with more than one link key candidate. For example, we can see the pair of classes (Book,Novel) (in orange) which is associated with

four link key candidates k_3, k_4, k_5 and k_6. This means that there are four possible choices to select a link key candidate for the pair (`Book,Novel`) among the four candidates.

Specifying the pairs of classes associated with a link key candidate is a critical task to properly evaluate this candidate. For example, the link key candidate k_4, in Figure 5.2, shows a low harmonic mean `hm=0.37`, because it is evaluated on the whole datasets. Consequently, k_4 is poorly ranked by a system based on FCA. This means that k_4 will not be returned as a relevant candidate despite the fact that it generates all the correct links between the classes `Book` and `Novel` while no other candidate is able to generate those links. By contrast, in Figure 5.3, k_4, shows a good harmonic mean `hm=0.85` because it is evaluated on the "right pair" of classes (`Book,Novel`). The candidate k_4 will be returned by a $LK-$pattern structure as a relevant candidate for the pair of classes (`Book,Novel`). Hence, we can appreciate the importance of introducing the notion of $LK-$pattern structure and the discovery of link key candidates associated with pairs of classes.

Linkex

This method is implemented in the Linkex tool[21]. This tool takes two RDF datasets as input, provided as files in Turtle, RDF/XML, or NTriple format, and outputs the extracted link key candidates.

The link key extraction procedure involves the following steps:

1. Indexation and preprocessing of RDF datasets:During this phase, values are normalized to reduce their heterogeneity. The transformation includes removing diacritics, tokenizing (using sequences of non-digit or letter characters as separators), and sorting the resulting bag of tokens. Linkex also allows for considering property compositions and inverse properties. Additionally, properties can be filtered using a minimal support and/or a discriminability threshold.

2. Construction of a set of descriptions that associates, for each pair of instances, the sets of pairs of properties for which the instances share at least one value or all values.

3. Computation of the lattice of link key candidates using the addIntent algorithm [63].

4. Quality evaluation of candidates.

5. Rendering of candidates:The resulting link key candidates can be rendered in EDOAL[22], or in a text format where each link key is associated with several quality measure values, or as a GraphViz dot file representing the lattice.

5.4 Conclusion and Discussion

In this chapter, we introduce a Pattern Structure that significantly improves the link key discovery process. Firstly, it enables the discovery of link key candidates while specifying their associated pairs of classes. This allows for a more precise evaluation of candidates. Secondly, we extend the definition of link key candidates beyond being associated solely with a pair of atomic classes to include pairs of conjunctions and disjunctions of classes. This extension allows for the consideration of varying levels of abstraction across datasets.

[21] https://gitlab.inria.fr/moex/linkex
[22] https://moex.gitlabpages.inria.fr/alignapi/edoal.html

Future work can be pursued in several directions. Firstly, in this work, we restrict ourselves to link keys with datatype properties, we aim to extend this work to handle object properties and discover dependent link key candidates. These candidates are discovered using Relational Concept Analysis in [10]. However this method is restricted to the strategies described Section 5.1. To discover dependent link key candidates, we aim to investigate the use of the method proposed in [25], which combines Pattern Structures and Relational Concept Analysis.

Another direction could involve the discovery of more expressive link keys, which include more complex class definitions, similar to what is done for conditional keys [93]. Furthermore, even though we aim to minimize the need for a priori knowledge of datasets, we may exploit the semantics by using the ontology, i.e., the terminological knowledge within the pattern structure, using, for example, the properties `rdfs:subClassOf` and `rdfs:subPropertyOf`.

Finally, given the link key candidates discovered through the Pattern Structure presented in this chapter, we need an effective selection of relevant link key candidates, which will be the subject of the next chapter.

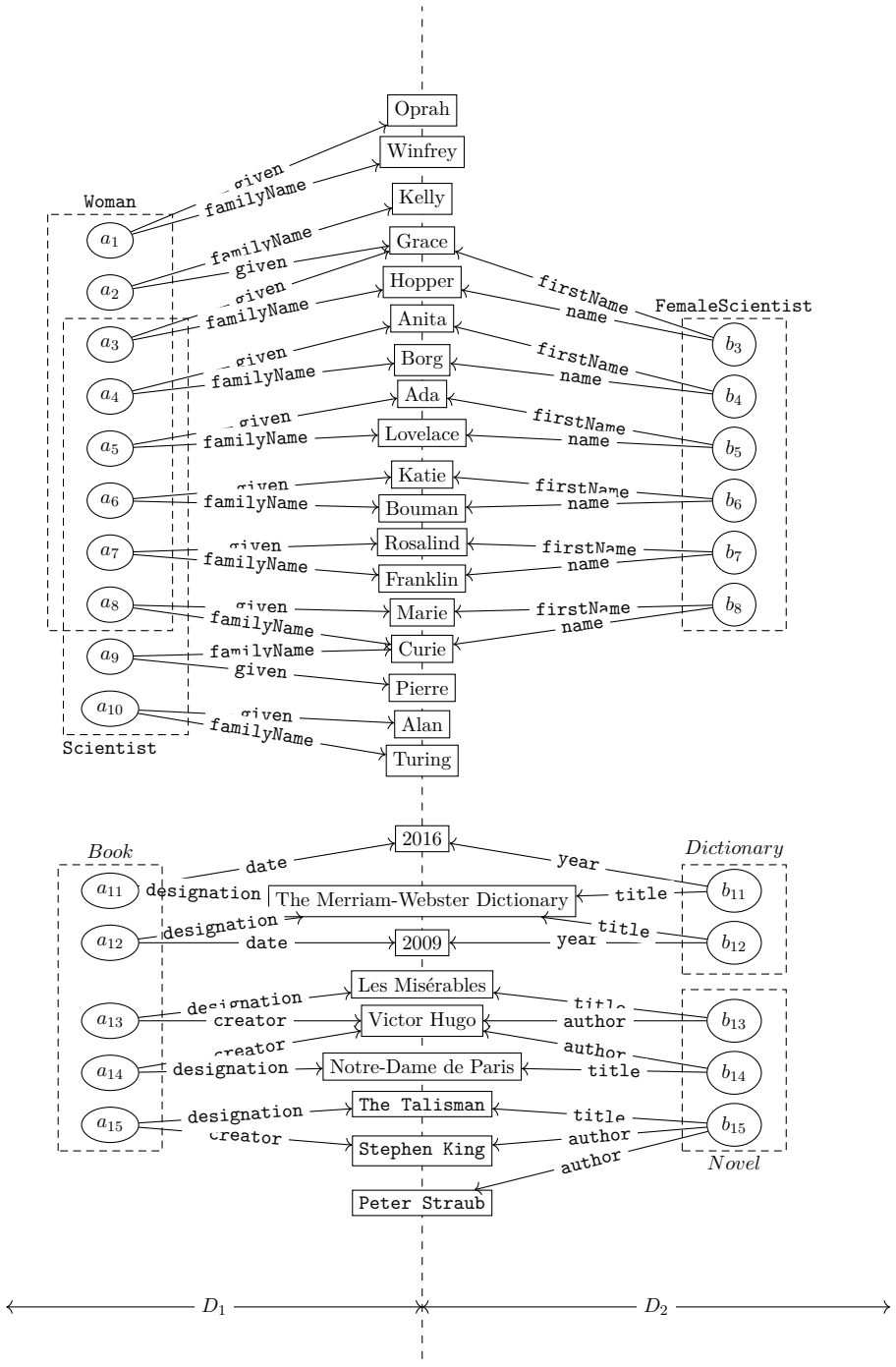


Figure 5.1: Example of two RDF datasets. In one hand the dataset D_1 populated with instances of the classes: `Woman`, `Scientist` and `Book`. In the other hand the dataset D_2 populated with instances of the classes: `FemaleScientist`, `Dictionary` and `Novel`.

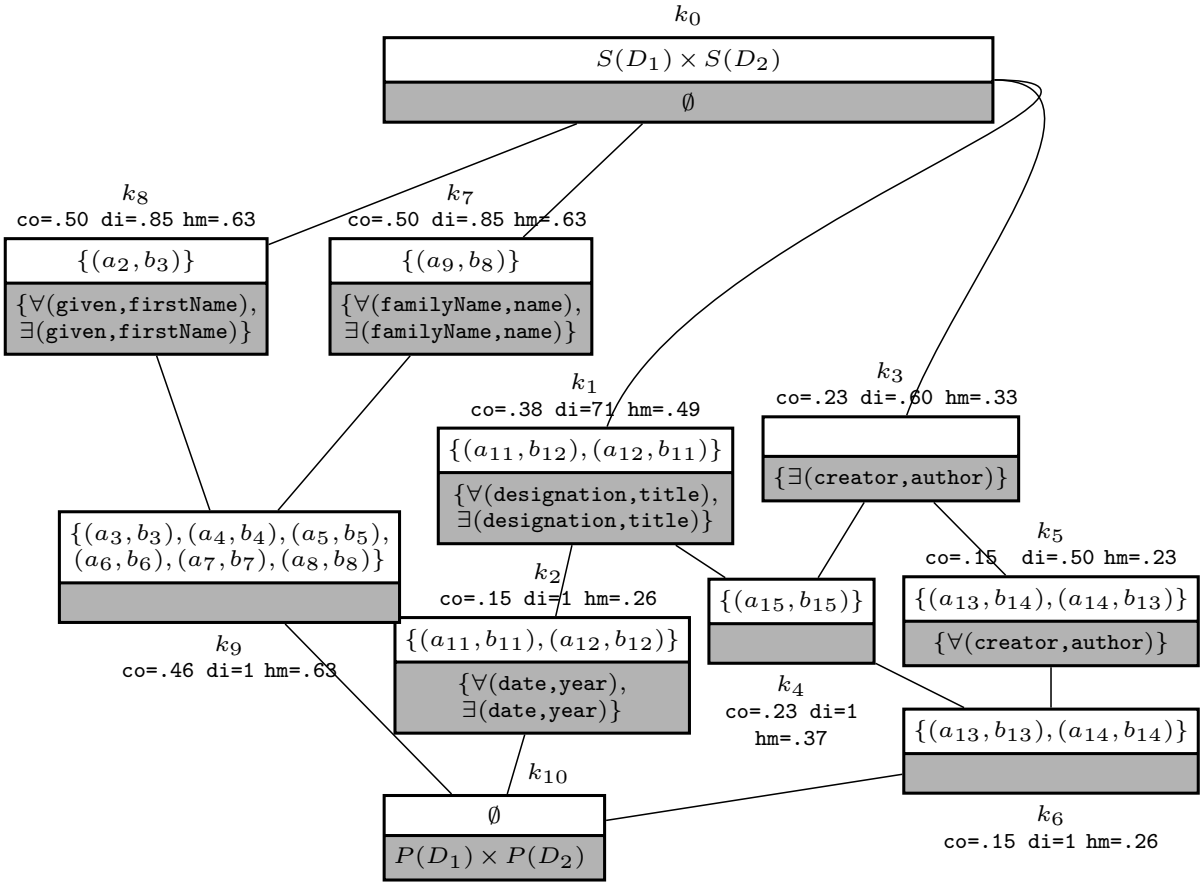


Figure 5.2: The reduced representation of the lattice of the $LK-$formal context represented in Table 5.1.

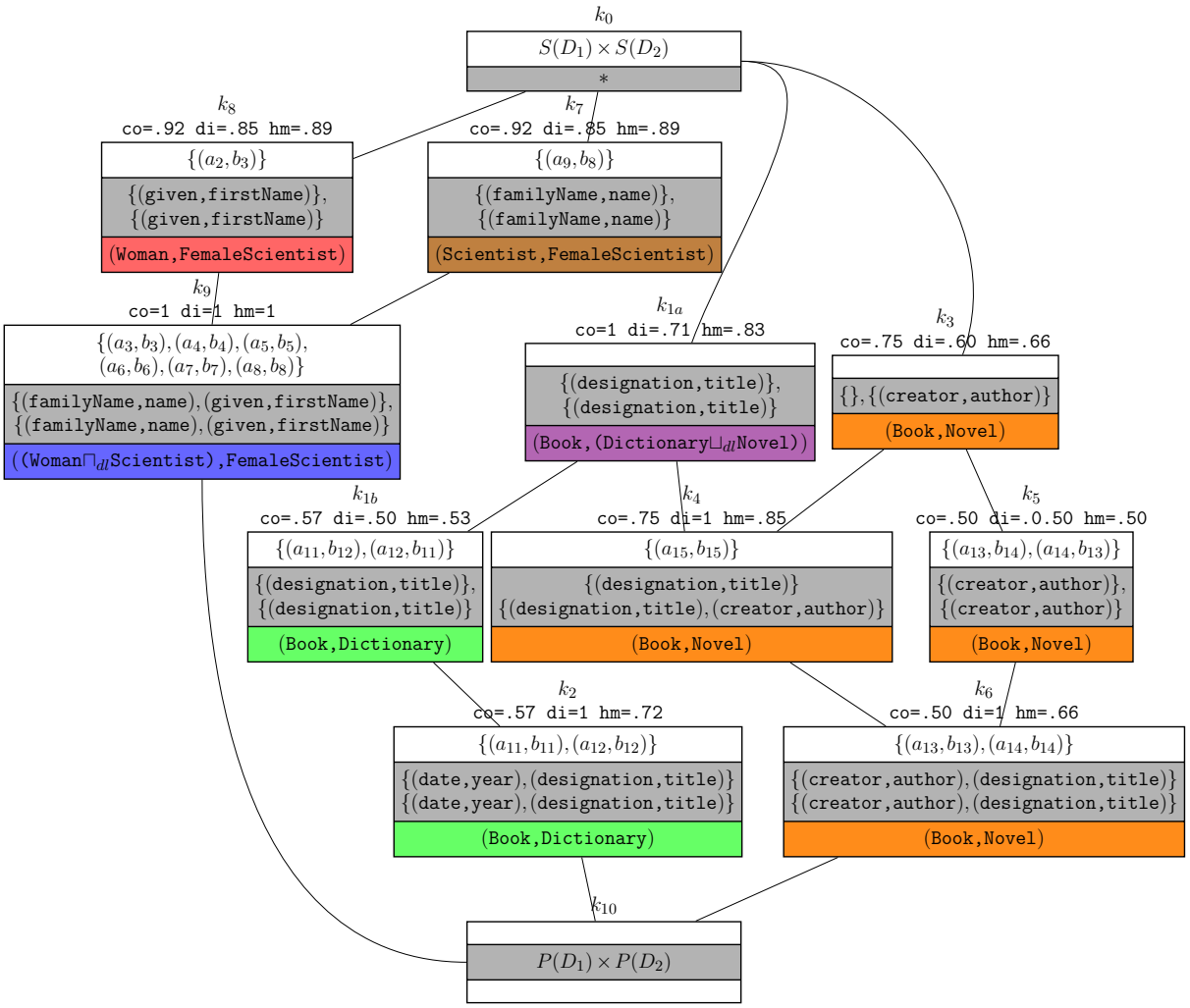


Figure 5.3: The LK-lattice generated from the $LK-$pattern structure in Table 5.3. Here the reduced representation applies only on objects.

Chapter 6

Selection of Relevant Link Key Candidates

Contents

6.1 Motivation 57
6.2 Link Key Selection Guided by the Classes 58
6.3 Link Key Selection Guided by the Lattice 64
6.4 The Sandwich algorithm at work 66
6.5 Conclusion and Discussion 69

This chapter is based on two papers:

Nacira Abbas, Jérôme David, and Amedeo Napoli. LKSA : un algorithme de sélectionde clés de liage dans des données RDF guidée par des paires de classes. In Jérôme Azé and Vincent Lemaire, editors, Extraction et Gestion des Connaissances, EGC 2021, 25-29 Janvier 2021, Montpellier, France, volume E-37 of RNTI, pages 205–216. Éditions RNTI, 2021 [6].

Nacira Abbas, Alexandre Bazin, Jérôme David, and Amedeo Napoli. Sandwich: An algorithm for discovering relevant link keys in an LKPS concept lattice. In Agnès Braud, Aleksey Buzmakov, Tom Hanika, and Florence Le Ber, editors, Proceedings of the International Conference on Formal Concept Analysis, ICFCA, volume 12733 of Lecture Notes in Computer Science, pages 243–251. Springer, 2021 [2].

6.1 Motivation

The link key discovery process consists of two primary stages. First, a set of link key candidates is identified, and then the relevant ones are selected from this set. When we refer to relevant link key candidates in this context, we mean candidates that are more likely to be valid link keys. This implies that all links produced by a specific candidate must be identity links.

To establish the relevance of a candidate, it is necessary to examine the links it generates. This evaluation can be done through various methods, such as seeking expert confirmation of link accuracy, exploiting existing identity links, or utilizing reasoning to ensure compliance with the ontology. However, these approaches are often not feasible due to the unavailability of experts, identity links, or ontology alignments for reasoning approaches. This chapter specifically focuses

on such situations. In this scenario, the work by Atencia et al. [7] relies on quality measures, i.e., coverage and discriminability. Specifically, the top-k candidates, determined by their harmonic mean of coverage and discriminability, are regarded as relevant and chosen as valid link keys.

Nevertheless, as illustrated in the previous chapter, these measures are less accurate when evaluating candidates in terms of the entire dataset rather than considering their associated classes. Indeed, even if a candidate is a valid link key, it may receive a low score for the harmonic mean. Moreover, candidates with the highest scores may only generate links for specific classes while disregarding others.

To address these challenges, Chapter 5 presented an LK-pattern structure that enables the identification of link key candidates with explicit specification of their associated classes. With the introduction of this Pattern Structure, it is now possible to evaluate candidates based on their classes, leading to an enhancement in the scores of the candidate for the harmonic mean. Nevertheless, relying solely on a simple top-k strategy for candidate selection is still inadequate. Even the top-k candidates may not be able to generate links for certain pairs of classes. Furthermore, there is still a possibility of selected candidates generating erroneous links.

Based on the previously introduced LK-pattern structure, in this chapter we introduce two algorithms to address the aforementioned challenges. The first algorithm, known as LKSA (Link Key Selection Algorithm) [6], is guided by the available pairs of classes provided by the LK-pattern structure. The second algorithm, called SANDWICH [2], leverages the structure of the LK-lattice and is based on the intuition that relevant link key candidates tend to reside in the middle of the lattice. In the following sections, we present a description of these algorithms.

6.2 Link Key Selection Guided by the Classes

In order to demonstrate the problem addressed in this chapter, we rely on the datasets depicted in Figure 6.1. Here the datasets to link are D_1 and D_2, with D_1 is populated with the instances of the classes `Newspaper` and `Person`. The dataset D_2 is populated with the instances of the classes `Journal` and `Personne`. The set of correct links considered here is, $L^{ref} = \{(a_i, b_i) \cup (a_6, b_5)$, such that $1 \leq i \leq 9\}$, any other link is considered as erroneous. Let LKC $= \{k_1, k_2, k_3, k_4, k_5, k_6\}$ be the set of link key candidates[23] over these datasets, retrieved from the LK-lattice over D_1 and D_2.
.

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

The scores of these link key candidates regarding coverage, discriminability, and their harmonic mean are provided in Table 6.1.

The first question of the selection problem is how to select a set of relevant link key candidates that generate identity links between all the classes in D_1 and D_2?

Let's consider an example where the top-2 link keys, labeled as k_1 and k_2, are selected based on the harmonic mean measure hm. Despite being the top candidates, k_1 and k_2 fail to generate

[23] In this example, we give only the In-candidates.

D_1

Newspaper

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

`Person`

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

D_2

Journal

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

`Personne`

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

Figure 6.1: Example of two datasets D_1 et D_2

Candidate	**Coverage** co	**Discriminability** di	**Harmonic mean** hm
k_1	1	0.83	0.90
k_2	0.81	0.80	0.80
k_3	0.36	1	0.53
k_4	0.20	1	0.33
k_5	0.60	1	0.75
k_6	0.20	1	0.33

Table 6.1: Link key candidate evaluation.

all the identity links between datasets D_1 and D_2. Specifically, they do not cover the pair (`Person,Personne`), resulting in the omission of identity links between these two classes.

One intuitive solution is to choose the best candidate, in terms of hm, for each pair of classes, namely k_1, k_3, k_4, and k_5. By considering these candidates collectively, we ensure the generation of all correct links between the datasets while covering all pairs of classes.

Nevertheless, these candidates result in the generation of erroneous links. One such example is the incorrect identification of instances a_1 (`Le Monde`) and b_{10} (`Paris Hilton`), which occurs due to the presence of candidate k_3 associated with the classes (`Newspaper,Personne`). These classes are regarded as "incompatible" due to the impossibility of an entity being simultaneously represented by both classes. This refers to the class disjointness which can be expressed using the OWL property `owl:disjointWith` within the Tbox, thus k_3 can be discarded as a valid link key. In the current scenario, our objective is to minimize the prior knowledge and solely consider the Abox. Consequently, information about the disjointedness of these classes is unavailable. The second question regarding the selection problem is how to identify and discard link key candidates associated with incompatible classes that lead to erroneous links?.

Drawing from these observations, we introduce the Link Key Selection Algorithm (LKSA), which operates by guiding the selection process using pairs of classes. The primary objective of this algorithm is to identify relevant candidates that generate links across all datasets' classes, except for the incompatible ones, in order to eliminate erroneous links.

Algorithm 1 represents the LKSA algorithm which relies on two key operations, namely TopLkClass (line 5) and RelevantClassForLk (line 12). This algorithm utilizes the candidates discovered thanks to the LK-pattern structure [5] introduced in Chapter 5.

TopLkClass

This function aims to select a subset of link key candidates, denoted as $KTop$, with the objective of linking all the classes. We define the set of pairs of classes associated with the link key candidates in LKC as CLASSES $= \{(c_1, c_2) \mid \exists k \in \text{LKC}$ such that $k = (In, (c_1, c_2))\}$. The TopLkClass function selects for each pair of classes $(c_1, c_2) \in$ CLASSES, the N best link key candidates based on hm (line 8). Here, N is a predetermined number specified by the user. On the current example, CLASSES $=\{$(`Newspaper,Journal`), (`Newspaper,Personne`), (`Person,Journal`), (`Person,Personne`)$\}$. For each pair in CLASSES the algorithm selects the $N = 1$ best link key candidate associated to it in LKC. Thus, the pair (`Newspaper,Journal`) is associated to k_1 and k_2, and k_1 is selected because $hm(k_1) > hm(k_2)$. The function is iterated for each of pair in CLASSES and returns the set of link keys $KTop = \{k_1, k_3, k_4, k_5\}$.

The candidates in $KTop$ identify all the correct links between the datasets D_1 and D_2 of the example. However, some $KTop$ link keys also generate erroneous links. For example k_4 identifies a_7 (`Nancy Reagan`) and b_2 (`Est R`) as identical, because they share the value `"Nancy"`. The candidate k_4 is associated with classes which are likely incompatible `Newspaper,Journal`.

If two classes are likely incompatible, we designate this pair as "irrelevant for data interlinking"; otherwise, we consider this pair as "relevant for data interlinking". The function RelevantClassForLk is precisely designed to select a set $KV \subseteq KTop$ containing candidates associated with pairs of classes relevant for data interlinking.

RelevantClassForLk

We hypothesize that the higher the number of entities shared by two classes, the greater the probability that this pair of classes is relevant for data interlinking. For this reason, we compute

Algorithm 1 LKSA

Input
LKC: The set of link key candidates over D_1 and D_2, along with their link sets
N: integer, the number of the best link keys to select per pair of classes
Output
KV: the set of the selected link key candidates for D_1 and D_2.

```
1: CLASSES ← the set of candidate pairs of classes
2: KTop ← TopLkClass(K, CLASSES, N)
3: KV ← RelevantClassForLk(KTop, CLASSES)
4: return KV
5: function TopLkClass(LKC, CLASSES, N)
6:     KTop ← ∅
7:     for each (c1, c2) ∈ CLASSES do
8:         KTop[(c1, c2)] ← N argmax_k hm(k) s.t. k = (In, (c1, c2)) ∈ LKC
9:     end for
10:    return KTop
11: end function
12: function RelevantClassForLk(KTop, CLASSES)
13:    Class1 ← {c1 | (c1, c2) ∈ CLASSES}
14:    Class2 ← ∅; KV ← ∅
15:    for each c1 ∈ Class1 do
16:        bestC2 ← argmax_c2 hm(KTop[(c1, c2)])
17:        Class2 ← Class2 ∪ {bestC2}
18:    end for
19:    for each c2 ∈ Class2 do
20:        bestC1 ← argmax_c1 hm(KTop[(c1, c2)])
21:        KV ← KV ∪ KTop[(bestC1, c2)]
22:    end for
23:    return KV
24: end function
```

the degree of overlap of each pair of classes associated with the candidates of $KTop$. The overlap of a pair of classes (c_1, c_2) associated with the candidates $KTop[(c_1, c_2)] = \{k \mid k = (In, (c_1, c_2)) \in \text{LKC}\}$ can be measured by $hm(KTop[(c_1, c_2)])$.

When $hm(KTop[(c_1, c_2)]) = 1$ it means that $KTop[(c_1, c_2)]$ identifies all entities in c_1 and all entities in c_2, while ensuring that each entity is represented by a single instance in c_1 and a single instance in c_2. The higher the overlap of a pair of classes, the greater the probability that this pair is relevant for data interlinking.

The function RelevantClassForLk allows to select in $KTop$ the link key candidate associated with the pairs of classes which are more likely to be relevant for the interlinking task. For each class c_1 of $Class1 = \{c_1 \mid (c_1, c_2) \in \text{CLASSES}\}$, RelevantClassForLk selects the class c_2 which gives the best overlap for c_1, denoted as $bestC2$ (line 16). This class $bestC2$ is then added to the set $Class2$ (line 17). Then, for each class $c_2 \in Class2$, the algorithm selects the class c_1 which gives the best overlap for c_2, denoted as $bestC1$ (line 20). The set of link keys $KTop[(bestC1, c_2)]$ is then added to KV (line 21).

In the example, given $Class1 = \{$`Newspaper,Personne`$\}$, the class `Newspaper` has a better overlap with the class `Journal` than the class `Personne` because $hm(\{k_1\})$ which is associated with the pair (`Newspaper,Journal`) is greater than $hm(\{k_3\})$ which is associated with the pair (`Newspaper,Personne`). The procedure is repeated for the class `Person` and the class `Personne` is selected, and $Class2 = \{$`Journal,Personne`$\}$.

The process is repeated with the classes of $Class2$. For the class `Journal`, the class `Newspaper` which gives the best overlap is selected and $\{k_1\}$ is added to KV. For the class `Personne`, the class `Person` is selected and $\{k_5\}$ is added to KV. Thus, the output of the algorithm is the set $KV = \{k_1, k_5\}$ which identifies all and only correct links between datasets D_1 and D_2, i.e., the set of links $\{(a_1, b_1), (a_2, b_2), (a_3, b_3), (a_4, b_4), (a_5, b_5), (a_6, b_5), (a_7, b_7), (a_8, b_8), (a_9, b_9)\}$.

Experiments

The aim of these experiments is to empirically verify the efficiency of the LKSA algorithm comparing to the top-k selection strategy. All our experiments[24] were carried out using a MacBook Pro Intel Core i7, 2.6 GHz with 16 GB of RAM allocated to the Java virtual machine.

We considered the datasets described in the Table 6.2.

- For the 2010 OAEI Restaurants and `Person1` datasets[25], identity links were provided between:
  - Instances of the class `Restaurant` for the `Restaurant1` and `Restaurant2` datasets.
  - Instances of the class `Person` for the `Person11` and `Person12` datasets.

  As our work deals with datasets with multiple classes, we also considered the discovery of identity links between the instances of the class `Address` present in these datasets. Moreover we considered the datasets `pr1` which is the fusion of `Restaurant1` and `Person1`, and `pr2` which is the fusion of `Restaurant2` and `Person2`.
- For the `Abox1` and `Abox2` datasets of `SPIMBench Sandbox` from OAEI 2018[26], we considered the classes `Program`, `BlogPost` and `NewsItem`.

[24] The details of the experiments are given in https://gitlab.inria.fr/nabbas/lksa.
[25] http://oaei.ontologymatching.org/2010/im/index.html
[26] http://oaei.ontologymatching.org/2018/spimbench.html

– The dataset `Db-Yago`[27], is composed of 10 classes from DBpedia and 10 classes from Yago which are: `Actor` , `Album`, `Book`, `City`, `Film`, `Mountain`, `Organization`, `Scientist`, `Museum` and `University`.

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

Table 6.2: Description of the datasets used in the experiments.

For each dataset, we generated the link key candidates thanks to the LK-pattern structure. We then selected a subset of link key candidates KV with the LKSA algorithm. The results of LKSA are compared with the best candidates top-k in terms of hm ensuring that $K = |KV|$. The performance of the two methods is calculated in terms of `precision`, `recall` and `F-measure`.

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

Table 6.3: The results of the selection.

The results of these experiments are given in Table 6.3. We notice that in all cases, we obtain with LKSA an `F-measure` greater than or equal to that of the top-k. LKSA obtains a better `F-measure` than top-k for the datasets `Person1`, `pr`, `SPIMBench` and `Db-Yago`. The recall is improved for `Person1` and `pr`. The precision is improved for `pr`, `SPIMBench` and `Db-Yago`.

The `F-measure` is not modified for the dataset `Restaurants` because the top-k link keys correspond exactly to the link keys selected by LKSA.

For the `Person1` and `pr` datasets, the top-k link keys are not associated with all possible pairs of classes, which explains lower recall. The LKSA algorithm on the other hand gives better recall because it selects link keys that are associated with more pairs of classes than the top-k

[27]https://github.com/lgalarra/vickey

link keys. For `Db-Yago` and `SPIMBench` datasets, we improve the precision because LKSA selects more link keys associated with relevant pairs of classes comparing to top-k, hence generating fewer erroneous links.

We have demonstrated the potential of the LKSA algorithm by experimenting on several datasets. Here, we note the importance of distinguishing the pairs of classes which are associated with the link keys, which is precisely the objective of the method proposed in Chapter 5.

Next, we introduce an alternative approach for selecting link keys, which is guided by the lattice structure.

6.3 Link Key Selection Guided by the Lattice

The evaluation criteria of a link key candidate, i.e., completeness and correctness, verify an "inverse" relationship, as correctness tends to decrease when completeness increases. We rely on this observation and we show that the upper part of an LK-lattice contains the most complete but the least correct link keys, while the lower part of the LK-lattice contains the most correct but the least complete link keys. Starting from this observation, we introduce an original pruning strategy combining a "bottom-up" and a "top-down" pruning strategies, while the most relevant link key candidates are lying "in the middle" of the lattice and achieve the best compromise between completeness and correctness. Accordingly, we propose the SANDWICH algorithm that traverse and prune the LK-lattice w.r.t. two measures estimating correctness and completeness. The input of this algorithm is an LK-lattice, a correctness measure and a threshold, and as well a completeness measure and a threshold. The output of the SANDWICH algorithm is a poset of relevant link key candidates.

Correctness and Completeness Measures

The measure of discriminability is used to estimate correctness. However, when datasets don't meet the Unique Name Assumption, this measure becomes unsuitable. Therefore, in this work, we introduce an alternative approach to assess the correctness of a link key candidate. Our hypothesis is that the more pairs of properties two instances a and b have in common, the higher the likelihood that the link (a,b) is correct.

Let k_1 and k_2 be two link key candidates such as [28]:
$k_1 = (\{(\texttt{country,pays})\},(\texttt{Person,Personne}))$ and
$k_2 = (\{(\texttt{country,pays}),(\texttt{lastName,nom}),(\texttt{firstName,prenom})\},(\texttt{Person,Personne}))$.

On one hand if k_1 generates a link (a,b) then this link is probably erroneous because two instances sharing the same value for (`country,pays`) doesn't mean that they represent the same person. The instances a, b represent in this case two people having the same country. On the other hand, if k_2 generates the link (a,b) then this link is probably correct because two instances sharing the same values for the pairs of properties (`country,pays`), (`lastName,nom`) and (`firstName,prenom`) means that they probably represent the same person.

We propose to measure the correctness of a link key candidate k by its size $|k|$.

Definition 11. *Let $k = (In,(c_1,c_2))$ be a link key candidate. The size of k is $|k| = |In|$, i.e., the number of its pairs of properties in In.*

The larger is the size of a link key, the higher is its probability of generating correct links. The correctness of k_1 is greater or equal to the correctness of k_2 if $|k_1| \geq |k_2|$ with k_1 and k_2 are comparable.

[28] Without loss of generality, we consider only In-link keys.

The size of a link key is a positive integer, lower-bounded by 1 since $In \neq \emptyset$ and upper-bounded by $|P(D_1) \times P(D_2)|$.

The size of link keys is monotone w.r.t. the intents of pattern concept intents (i.e. link key candidates) in an LK-lattice.

Property 1. *Let k_1 and k_2 be two link key candidates. The size of a link key is monotone w.r.t the intents in an LK- lattice.*

$$k_1 \sqsubseteq k_2 \Rightarrow |k_1| \leq |k_2|$$

Proof. Let $k_1 = (In_1, (c_1, c_2))$ and $k_2 = (In_2, (c_3, c_4))$ be two link key candidates. That is $(L(k_1), k_1)$ and $(L(k_2), k_2)$ are two pattern concepts of the LK-lattice.

$$k_1 \sqsubseteq k2 \Rightarrow In_1 \subseteq In_2 \tag{6.1}$$
$$\Rightarrow |k_1| \leq |k_2| \tag{6.2}$$

□

In [7], the measure of coverage was proposed to evaluate the completeness of a link key candidate. The coverage is locally monotone. That is, when the link key candidates are associated with the same pairs of classes, the coverage is monotone w.r.t. extents of these candidates in the LK-lattice. We recall that the coverage of a link key $k = (In, (c_1, c_2))$ is denoted by $co(k)$ and defined as:

$$co(k) = \frac{|\pi_1(L(k)) \cup \pi_2(L(k))|}{|I(c_1) \cup I(c_2)|}$$

with $\pi_1(L) = \{a | (a,b) \in L\}$ and $\pi_2(L) = \{b | (a,b) \in L\}$.

Property 2. *Let $k_1 = (In_1, (c_1, c_2))$ and $k_2 = (In_2, (c_1, c_2))$ be two link key candidates associated with the same pair of classes (c_1, c_2).*
The measure of coverage is monotone.

$$k_1 \sqsubseteq k_2 \Rightarrow co(k_1) \geq co(k_2)$$

Proof. Let $k_1 = (In_1, (c_1, c_2))$ and $k_2 = (In_2, (c_1, c_2))$ be two link key candidates. That is $(L(k_1), k_1)$ and $(L(k_2), k_2)$ are two pattern concepts of the LK-lattice.

$$k_1 \sqsubseteq k2 \Leftrightarrow L(k_1) \supseteq L(k_2) \tag{6.3}$$
$$\Leftrightarrow \pi_1(L(k_1)) \supseteq \pi_1(L(k_2)) \text{ and } \pi_2(L(k_1)) \supseteq \pi_2(L(k_2)) \tag{6.4}$$
$$\Leftrightarrow |\pi_1(L(k_1)) \cup \pi_2(L(k_1)| \geq |\pi_1(L(k_2) \cup \pi_2(L(k_2)| \tag{6.5}$$
$$\Leftrightarrow co(k_1) \geq co(k_2) \tag{6.6}$$

□

The Sandwich algorithm

Given an LK-lattice, we propose the `Sandwich` algorithm (Algorithm 2) that identifies the relevant link key candidates (intents) and prunes the irrelevant candidates in this lattice. The input of this algorithm is an LK-lattice, and two measures, namely a completeness measure σ_{comp} and a correctness measure σ_{cor}, and in addition two thresholds, namely a minimum completeness μ_{comp} and a minimum correctness μ_{cor}. The output of the `Sandwich` algorithm is a poset called KV of relevant link key candidates. To identify relevant link keys associated with each pair of classes, the first step in the `Sandwich` algorithm is to split the lattice into sub-lattices (function Cut (line 8) in the Algorithm 2) where all intents are link key candidates associated with the same pairs of classes.

In the next step, the `Sandwich` algorithm prunes the sub-lattices. As mentioned above, a relevant link key candidate has to satisfy two criteria:

1. Correctness: the algorithm retains link key candidates for which the score of correctness measure σ_{cor} is equal to or greater than the minimum correctness threshold μ_{cor}. The `Sandwich` algorithm is based on monotone correctness measures w.r.t. the intents. Therefore, the most correct link keys are those in the lower part of a given sub-lattice. In order to retain the most correct link keys, the `Sandwich` algorithm uses a Bottom-Up pruning strategy. It calls (line 4) the `Slice` algorithm (see Algorithm 3) which takes as input an sub-lattice, σ_{cor} and μ_{cor} and returns a set of pattern concepts $BottomSlice$ (the bottom part of the lattice in the input). The intents in $BottomSlice$ are link key candidates having score of correctness $\geq \mu_{cor}$.

2. Completeness: the algorithm retains link key candidates for which the score of completeness measure σ_{comp} is equal to or greater than the minimum completeness threshold μ_{comp}. The `Sandwich` algorithm requires monotone completeness measures w.r.t. the extents. Therefore, the most complete link keys are those in the upper part of a given sub-lattice. In order to retain the most complete link keys the `Sandwich` algorithm uses a Top-Down pruning strategy. It calls (line 5) the `Slice` algorithm (see Algorithm 3) which takes as input an sub-lattice, σ_{comp} and μ_{comp}, and returns a set of pattern concepts $TopSlice$ (the upper part of the lattice in the input). The intents in $TopSlice$ are link key candidates having score of completeness $\geq \mu_{cor}$.

In order to keep the link keys reaching a good compromise between the completeness and the correctness, the `Sandwich` algorithm retains in $KV\,[(c_1, c_2)]$ the concepts which are at the same time in the $BottomSlice$ and $TopSlice$ for a given pair of classes (c_1, c_2) (line 6). Finally, the output of the algorithm is the poset KV containing the relevant candidates for all the classes.

6.4 The Sandwich algorithm at work

Let the lattice[29], represented in Figure 6.2, be the LK-lattice over the RDF datasets D_1 and D_2. We aim at retaining in LK-lattice the relevant link key candidates and removing the irrelevant candidates thanks to the `Sandwich` algorithm. The input of the algorithm includes the LK-lattice and some other parameters which are made precise below:

[29]These datasets and the implementation generating the lattice can be checked at https://gitlab.inria.fr/nabbas/sandwich_algorithm.

Algorithm 2 Sandwich

Input D_1 and D_2 two RDF datasets.
The LK-lattice over D_1 and D_2.
CLASSES: the pairs of classes (c_1, c_2) appearing in the intents of the LK-lattice.
σ_{cor}: a monotone correctness measure w.r.t intents, μ_{cor}: minimum correctness.
σ_{comp}: a monotone completeness measure w.r.t extents, μ_{comp}: minimum completeness.

Output KV: a poset of pattern concepts from LK-lattice. Each concept's intent in KV is considered as a relevant link key candidate.

```
1: Relevant ← ∅
2: sub−lattice ← Cut(LK-lattice,CLASSES)
3: for each sub−lattice[(c1,c2)] do
4:     BottomSlice ← Slice(sub−lattice[(c1,c2)],σcor,μcor,Up)
5:     TopSlice ← Slice(sub−lattice[(c1,c2)],σcomp,μcomp,Down)
6:     KV[(c1,c2)] ← BottomSlice∩TopSlice
7: end for
8: function Cut(LK-lattice,CLASSES)
9:     for each (c1,c2) ∈ CLASSES do
10:        sub−lattice[(c1,c2)] ← all Concept_i ∈ LK-lattice such as Concept_i intent is k_i = (In_i,(c1,c2))
11:    end for
12:    return sub−lattice
13: end function
```

- σ_{cor} a monotone correctness measure w.r.t. intents. To evaluate correctness, we have the option to consider either the size or the discriminability of a link key. In this case, we opt for the size of the link key because it exhibits a monotone property, which is required by the SANDWICH algorithm. In contrast, the discriminability does not possess this property.

- The minimum size of a link key threshold is set to $\mu_{cor} = 3$.

- σ_{comp} is a monotone completeness measure w.r.t extents. Here we choose the coverage of a link key $co(k)$ as measure of completeness because it is monotone when link keys are associated with same pairs of classes.

- The minimum coverage threshold is set to $\mu_{comp} = 0.9$.

The first step (line 2) is to split the LK-lattice into sub-lattices where each sub-lattice is related to the same pair of classes (see the CUT function in Algorithm 2)) and contains link key candidates associated with this particular pair of classes. In the running example, the CUT function provides two sub-lattices. On one hand, a sub-lattice for the pair of classes (`Newspaper,Journal`) containing the pattern concepts k_0, k_1, k_2, k_3 and k_8. One the other hand, a sub-lattice for the pair of classes (`Person,Personne`) containing the pattern concepts k_0, k_4, k_5, k_6, k_7 and k_8.

Now, for each sub-lattice, the next step in the SANDWICH algorithm is to:

1. Apply a Bottom-Up pruning strategy by calling the algorithm `Slice` with the following parameters, ($sub-lattice$[(`Newspaper,Journal`)], $size$, 3, Up). The algorithm `Slice` with these parameters returns for each sub-lattice a set of pattern concepts $BottomSlice$ which contains all the pattern concepts in this sub-lattice, where each concept's intent is a link key candidate k for which $size(k) \geq 3$. That is the link key candidates in $BottomSlice$

Algorithm 3 Slice

Input $sub-lattice[(c_1,c_2)]$,σ,μ, direction

Output *Relevant*: a poset of concepts from $sub-lattice[(c_1,c_2)]$ where each concept's intent is a link key candidate having a score for the measure σ greater or equal to a threshold μ

```
1:  function SLICE(sub−lattice[(c1,c2)],σ,μ, direction)
2:      if direction=down then
3:          Relevant ← the Top concept of sub−lattice[(c1,c2)]
4:      else
5:          Relevant ← the Bottom concept of sub−lattice[(c1,c2)]
6:      end if
7:      processed ← ∅
8:      tested ← ∅
9:      for each Concept ∈ Relevant and Concept ∉ processed do
10:         processed ← processed ∪ {Concept}
11:         neighbors=NEIGHBORHOOD(Concept,direction)
12:         for each (L(k),k) ∈ neighbors and (L(k),k) ∉ tested do
13:             tested ← tested ∪{(L(k),k)}
14:             if σ(Concept)≥ μ then
15:                 Relevant ← Relevant ∪{(L(k),k)}
16:             end if
17:         end for
18:     end for
19:     return Relevant
20: end function
21: function NEIGHBORHOOD(Concept,direction)
22:     if direction=Down then
23:         return the set of all direct descendants of Concept in LK-lattice
24:     else
25:         return the set of all direct ascendants of Concept in LK-lattice
26:     end if
27: end function
```

are the most correct candidates. For example for the sub-lattice over the pair of classes (`Newspaper,Journal`), the algorithm `Slice` returns $BottomSlice = \{k_2, k_3\}$.

2. Apply a Top-Down pruning strategy by calling the algorithm `Slice` with the parameters ($sub-lattice$[(`Newspaper,Journal`)], co, 0.9, down). The algorithm `Slice` with these parameters returns for each sub-lattice a set of pattern concepts $TopSlice$ which contains all the pattern concepts in this sub-lattice, where each concept's intent is a link key candidate k for which $co(k) \geq 0.9$. This means that the link key candidates in $TopSlice$ are the most complete candidates. For example, in the sub-lattice over the pair of classes (`Newspaper,Journal`), the algorithm `Slice` returns $TopSlice = \{k_1, k_2\}$.

3. The final step for each sub-lattice is to select as relevant link keys the ones which are in $BottomSlice$ and in the $TopSlice$ at the same time. For example, for the sub-lattice over the pair of classes (`Newspaper,Journal`), the relevant link key candidates are the intents of the concepts in KV[(`Newspaper,Journal`)] $= BottomSlice \cap TopSlice = \{k_2\}$. Indeed the link key candidate k_2 achieves a good compromise between correctness and completness. For the sub-lattice over the pair of classes (`Person,Personne`), the relevant link key candidates for this pair are in KV[(`Person,Personne`)] $= BottomSlice \cap TopSlice = \{k_5\}$.

The output of the SANDWICH algorithm is the union of relevant link key candidates for each pair of classes. For the LK-lattice represented in Figure 6.2 the algorithm returns the relevant link key candidates $KV = \{k_2, k_5\}$. Subsequently, these link keys will be used to find identity links over the RDF datasets D_1 and D_2.

6.5 Conclusion and Discussion

This chapter centers around the difficulty of selecting relevant link key candidates, which are the ones which are likely valid link keys. This challenge particularly arises in scenarios where there are no experts, reference links, or reasoning available. We have demonstrated that selecting the top-k candidates, w.r.t. coverage and discriminability, can fail to retrieve relevant link key candidates. To tackle these issues, we have proposed two algorithms for selecting link key candidates.

The first algorithm, named LKSA, operates by considering pairs of classes. It prioritizes the selection of the top-k candidates, w.r.t. coverage and discriminability, for each pair of classes and then identifies and eliminates candidates associated with classes that are likely incompatible. This approach helps eliminate erroneous links. Experimental results have shown that the LKSA algorithm outperforms selection strategies solely based on ranking candidates using discriminability and coverage measures.

The second algorithm, known as the SANDWICH algorithm, leverages the structure of the LK-lattice. It operates based on the intuition that relevant link key candidates tend to be situated in the middle of the lattice. By following the duality principle and exploring the pattern concept lattice in both top-down and bottom-up manners, this algorithm aims to discover the candidates achieving a balance between correctness and completeness. Additionally, we introduced a new measure called the size of a link key, which is better suited for datasets that do not follow the Unique Name Assumption (UNA) compared to discriminability. It's worth noting that the SANDWICH algorithm is generic and can be applied with various quality measures, as long as they exhibit monotonicity or local monotonicity.

Both algorithms build upon the LK-lattice introduced in Chapter 5. Their objective is to select relevant candidates that establish links between different classes in the datasets. The

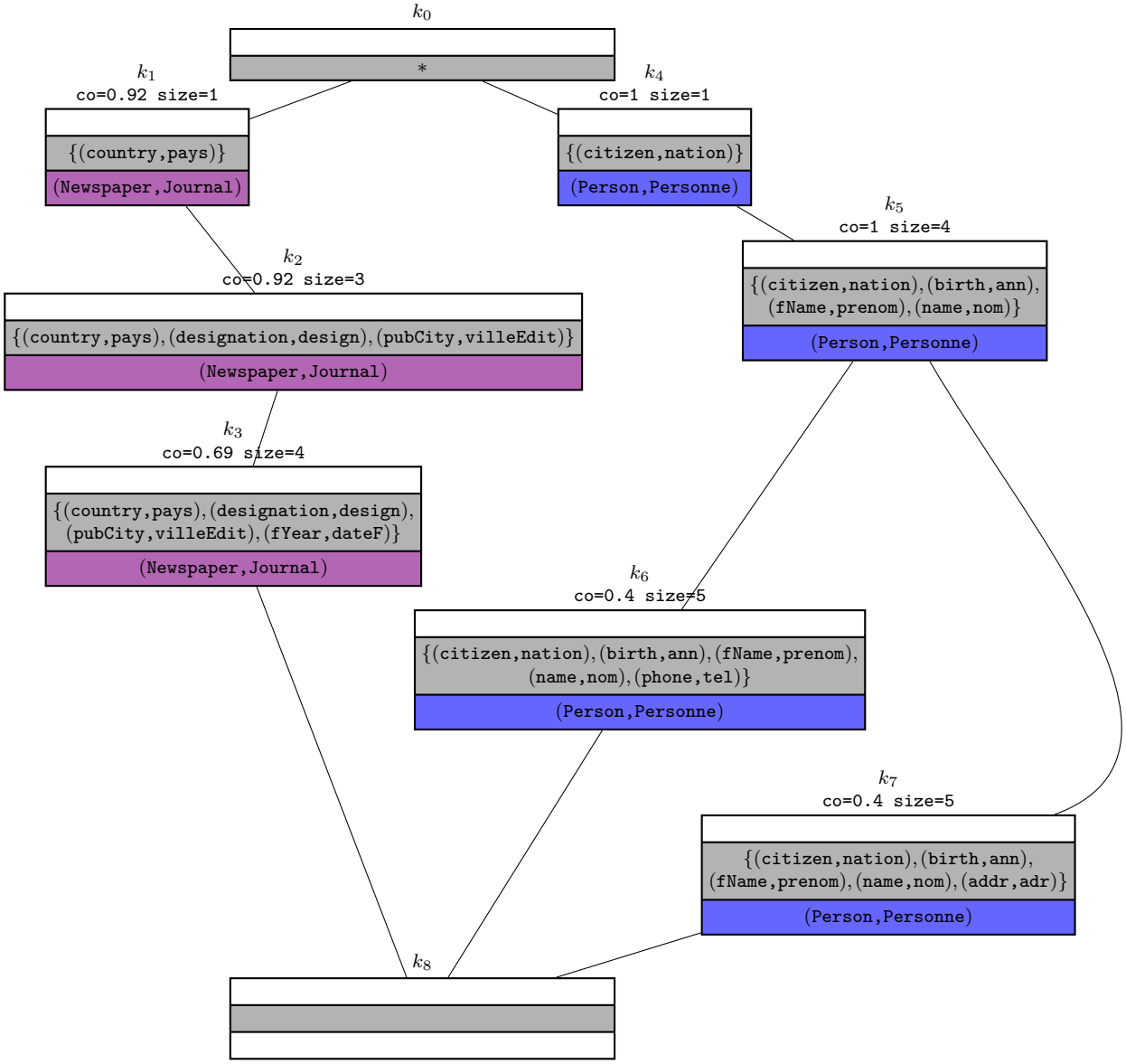


Figure 6.2: LK-lattice for link key candidate discovery over the datasets D_1 and D_2. Only the pattern concepts' intents, i.e. link key candidates, are represented here.

difference between these algorithms lies in how they discard candidates generating erroneous links: LKSA discards candidates associated with irrelevant classes for interlinking, while the Sandwich algorithm uses the size of a candidate as a criterion.

There are several directions to enhance the current research. While we have provided a proof of concept that the Sandwich algorithm is efficient, further experimentation on real datasets is required to establish its effectiveness. Additionally, a comparative experimental analysis of the two algorithms is necessary. The Sandwich algorithm selects link key candidates associated with pairs of classes, even those that may be incompatible. To discard these candidates, we can utilize the RelevantClassForLk function from LKSA, which retains only the candidates associated with relevant pairs of classes. It would also be beneficial to expand the range of measures used, including measures employed in Formal Concept Analysis and data mining communities, such as the "stability" of a concept [58]. These measures could provide valuable insights for characterizing link keys. Furthermore, scalability issues may arise due to the large size of the LK-lattice, which will be addressed in the subsequent chapter aiming to reduce the redundancy of link key candidates.

Chapter 7

Investigating Redundancy of Link Key Candidates

Contents

7.1 Motivation 73

7.2 Partition-based Redundancy 73

7.2.1 Partition Pattern Structures for non Redundant Candidate Discovery . 73

7.2.2 Partition-based Quality Measures 75

7.2.3 Experiments 79

7.3 Similarity-based Redundancy 83

7.3.1 Building an Approximate Basis of Link Keys 85

7.3.2 Experiments 88

7.4 Conclusion and Discussion 91

This chapter is based on three papers:

Nacira Abbas, Alexandre Bazin, Jérôme David, and Amedeo Napoli. Discovery of link keys in resource description framework datasets based on pattern structures. International Journal of Approximate Reasoning, 161:108978, 2023 [4].

Nacira Abbas, Alexandre Bazin, Jérôme David, and Amedeo Napoli. A study of the discovery and redundancy of link keys between two RDF datasets based on partition pattern structures. In Pablo Cordero and Ondrej Krídlo, editors, Proceedings of the Sixteenth International Conference on Concept Lattices and Their Applications (CLA 2022) Tallinn, Estonia, June 20-22, 2022, volume 3308 of CEUR Workshop Proceedings, pages 175–189. CEUR-WS.org, 2022 [3].

Nacira Abbas, Alexandre Bazin, Jérôme David, and Amedeo Napoli. Non-redundant link keys in RDF data: Preliminary steps. In Sergei O. Kuznetsov, Amedeo Napoli, and Sebastian Rudolph, editors, Proceedings of the 9th International Workshop "What can FCA do for Artificial Intelligence?" co-located with the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021), Montréal, Québec, Canada, August 21, 2021, volume 2972 of CEUR Workshop Proceedings, pages 125–130. CEUR-WS.org [1].

7.1 Motivation

Chapter 4 discusses how the existing algorithms for link key discovery do not exhaustively examine all link key expressions to determine their validity as link keys due to their large number. Indeed redundancy occurs when multiple link key expressions generate the same set of links. To address this issue, a maximal link key expression is chosen to represent each set of redundant link key expressions, resulting in a set of link key candidates. The method based on Formal Concept Analysis ensures that each link key candidate generates a unique set of links, since each intent, i.e. link key candidate, corresponds to only one extent, i.e. link set. However, despite this reduction, the set of candidates remains large due to the extensive amount of data. Consequently, this poses challenges for the visualization, analysis, and evaluation of link key candidates. Therefore, the objective of the work presented in this chapter is to determine if any remaining redundancy exists within the set of link key candidates. The goal is to identify and subsequently reduce this redundancy.

In order to address this issue, we examine the partition of instances induced by the owl:sameAs relation within the link set of each link key candidate. We identify candidates that generate identical partitions as redundant. Consequently, we introduce a Partition Pattern Structure [15] that detects and merges redundant candidates based on their partition equality.

However, our experiments on real datasets demonstrate that redundancy based on partitions is nonexistent or rare. As a result, we propose an alternative approach that identifies redundancy by considering candidates that generate similar sets of links. To detect and minimize this redundancy, we have developed a hierarchical clustering-based method. Within this approach, candidates that produce similar sets of links are grouped into clusters, and a representative is selected from each cluster. Experiments on real datasets show that this approach effectively reduces redundancy among the link key candidates.

7.2 Partition-based Redundancy

7.2.1 Partition Pattern Structures for non Redundant Candidate Discovery

To begin, we provide an illustrative example that explains how the partition-based redundancy is detected.

Let us examine the link sets generated by link key candidates in the pattern concept lattice in Figure 7.2. For example, the link set $L(k_{11}) = \{(a_1, b_1), (a_2, b_2), (a_2, b_1)\}$ is generated by k_{11}. In other terms, the link key candidate k_{11} states that $(a_1, \text{owl:sameAs}, b_1)$, $(a_2, \text{owl:sameAs}, b_2)$, and $(a_2, \text{owl:sameAs}, b_1)$. As the owl:sameAs relation is an equivalence relation, i.e., reflexive, symmetric, and transitive, we compute the transitive closure of it in $L(k_{11})$. This yields to the partition $\{\{a_1, a_2, b_1, b_2\}, \{a_i\}, \{b_i\}\}$ with $3 \leq i \leq 17$. Then it can be concluded that the four subjects a_1, a_2, b_1, and b_2, represent the same entity, they are equivalent. Whereas the other subjects in both datasets, i.e. the singletons in the partition, $\{a_i\}$ and $\{b_i\}$ are not linked to any subject by k_{11}. In the following, for the sake of readability, we omit the singletons in the partitions. Notice, that each element of this partition is an equivalence class of the owl:sameAs relation induced by k_{11} in the set of subjects. More formally, let D_1 and D_2 be two RDF datasets and k a link key candidate over these datasets. Let S be the union of the subjects of D_1 and D_2, $S = S(D_1) \cup S(D_2)$. The owl:sameAs relation induced by a link key candidate k is denoted by $=_k$. Then, *partition of subjects induced by* $=_k$ is defined as the quotient set $Part(k) = S/(=_k)$. We say that k generates the partition $Part(k)$.

In Figure 7.2, we can observe that the link key candidates k_{11} and k_{12} generate the same

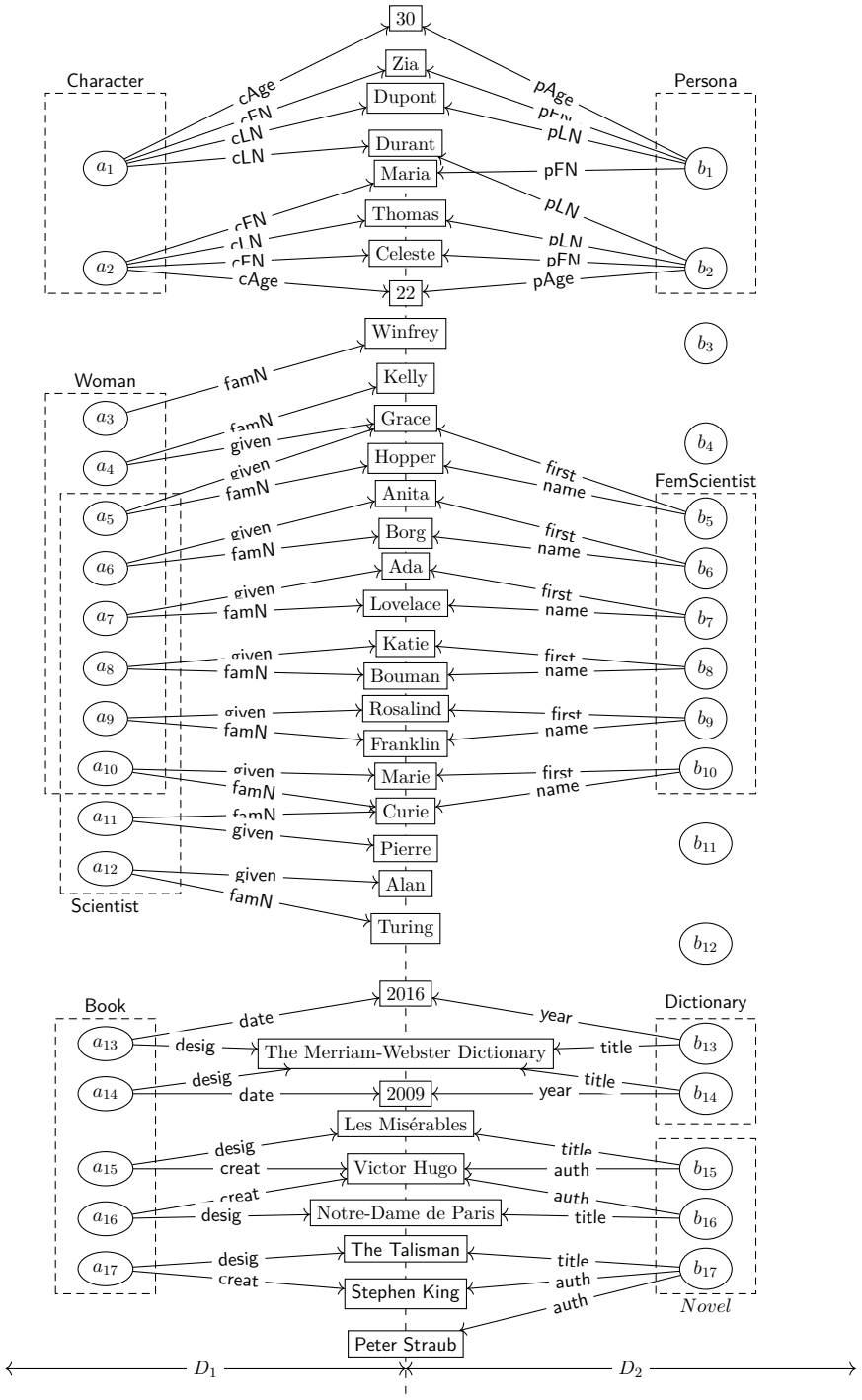


Figure 7.1: An example of two RDF datasets, with D_1 on the left-hand side and D_2 on the right-hand.

partition, i.e., $Part(k_{11}) = Part(k_{12}) = \{\{a_1, a_2, b_1, b_2\}\}$. Therefore, these two link key candidates are carrying the same information about identity links. Thus, they can be seen as redundant.

In order to detect and reduce this redundancy, we introduce an LK-partition pattern structure where the meet of two partitions $\sqcap_{part}$ is defined as the coarsest common refinement of the two partitions [15]. For example:

$\{\{a_{15}, a_{16}, b_{15}, b_{16}\}\} \sqcap_{part} \{\{a_{15}, b_{15}\}, \{a_{16}, b_{16}\}, \{a_{17}, b_{17}\}\} = \{\{a_{15}, b_{15}\}, \{a_{16}, b_{16}\}\}$.

Definition 12 (LK-partition pattern structure)**.** *Let D_1 and D_2 be two RDF datasets and let $S = S(D_1) \cup S(D_2)$ their set of subjects. The LK-*partition pattern structure *over D_1 and D_2 is defined as the triple $(\text{LKC}, (Partition(S)), \sqcap_{part}), \delta)$ such that:*

- LKC *is the set of link key candidates over D_1 and D_2 computed thanks to the LK-pattern structure introduced in Chapter 5,*
- *$Partition(S)$ is the set of partitions of the set of all subjects in D_1 and D_2, while $\sqcap_{part}$ is the "meet" of two partitions,*
- *δ maps a link key candidate k to its partition $Part(k) = S/(=_k)$.*

The LK-partition pattern structure corresponding to the RDF datasets displayed in Figure 7.1 is given in Table 7.1. The objects are the link key candidates $k_{1a}, k_{1b}, k_2, \ldots, k_9, k_{12}, k_{13}$, while the descriptions are the related partitions of subjects.

Pattern concepts from this LK-partition pattern structure are composed of a set of link keys (the extent) and a partition of the set of subjects (the intent) that is the coarsest refinement of the partitions induced by the link keys in the extent.

In order to detect redundant link keys, we want to find sets of link keys that induce the same partition of subjects, i.e., link keys k_n and k_m such that $Part(k_n) = Part(k_m)$. As such, we do not need all the concepts of this LK-partition pattern structure but only the concepts in which the intent is a partition of subjects induced by a single link key, i.e. concepts of the form $(k_m^{\square\square}, k_m^{\square})$. Those concepts are, by definition, the object concepts introducers (about the OC-poset, see Chapter 3). If the extent of these concepts contains more than one element, it contains redundant link keys.

Figure 7.3 shows the OC-poset of the running example completed into a lattice (for the sake of readability). We can observe that the link key candidates k_{11} and k_{12} are merged into the same concept, at the bottom left. This concept can be interpreted in the following way: k_{11} and k_{12} are inducing the same partition and they should be considered as redundant. By contrast, the extents of the other concepts contain only a single element as these link key candidates are not redundant.

7.2.2 Partition-based Quality Measures

We aim to leverage the defined partitions created by a candidate and establish novel quality measures centered around these partitions. First, we introduce direct extensions of the measures of discriminability and coverage, applied to the extended set of links inferred thanks to the owl:sameAs equivalence relation. Then, we propose measures based on the partitions .

Coverage and discriminability are calculated thanks to the set of links $L(k)$ directly generated by a candidate k. They may also be computed by considering all links generated thanks to transitivity and symmetry of the owl:sameAs relation. For a given candidate $k = (Eq, In, (c_1, c_2))$, the set of such links is defined as:

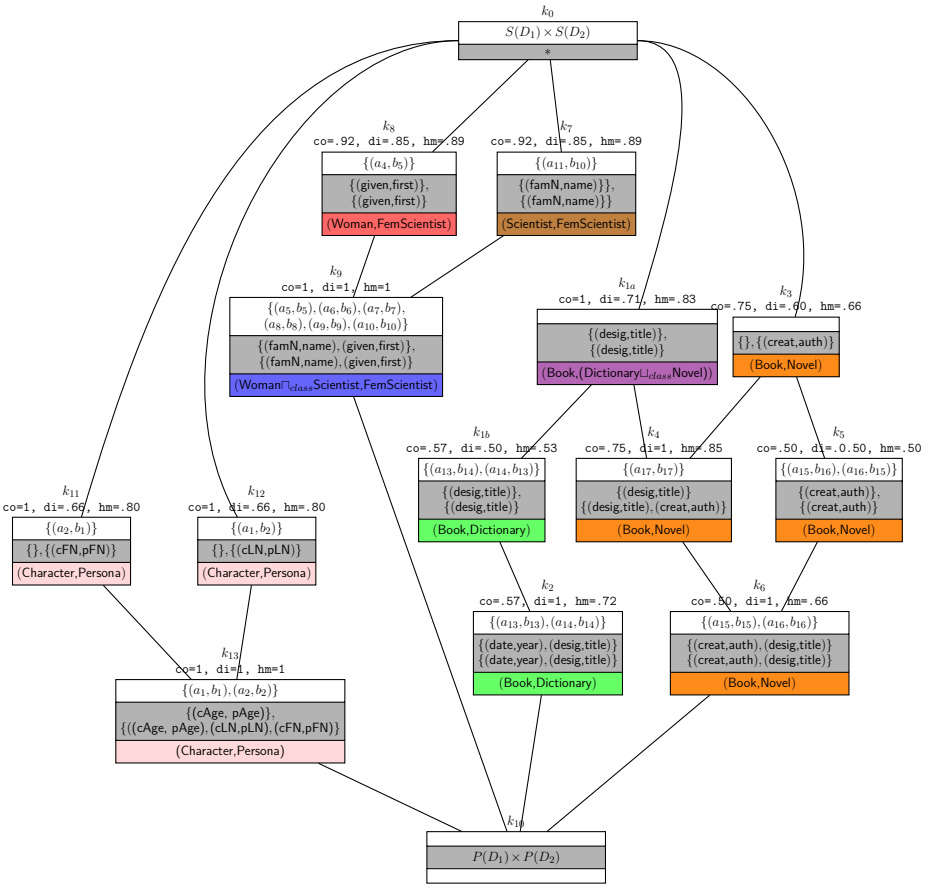


Figure 7.2: A pattern concept lattice constructed from the LK-pattern structure over the datasets in 7.1. This is a reduced representation for only the objects.

$$L^+(k) = \{(a,b) \in I(c_1) \times I(c_2) \mid a =_k b\}$$

Since $L(k) \subseteq L^+(k)$,

- $co(L(k), c_1, c_2) = co(L^+(k), c_1, c_2)$
- $di(L(k)) \geq di(L^+(k))$

The coverage value remains the same whenever it is calculated on the direct links or on the whole set of direct and inferred links because it relies on the (separate) sets of linked instances in classes c_1 and c_2, which do not vary. However, the value of discriminability can change:

$$di(L(k_{11})) = \frac{min(|\{a_1,a_2\}|, |\{b_1,b_2\}|)}{|\{(a_1,b_1),(a_2,b_1),(a_2,b_2)\}|} = \frac{2}{3}$$
$$di(L^+(k_{11})) = \frac{min(|\{a_1,a_2\}|, |\{b_1,b_2\}|)}{|\{(a_1,b_1),(a_2,b_1),(a_2,b_2),(a_1,b_2)\}|} = \frac{2}{4}$$

Given a link key candidate k, global measures of completeness and correctness compliant with the properties of owl:sameAs can be defined w.r.t. the partition generated by the equivalence relation $=_k$ induced by k. As we disscuss it in Chapter 5, it is more accurate to evaluate a

PPS objects (k_i)	Descriptions ($\delta(k_i) = Part(k_i)$)
k_{1a}	$\{\{a_{13}, a_{14}, b_{13}, b_{14}\}, \{a_{15}, b_{15}\}, \{a_{16}, b_{16}\}, \{a_{17}, b_{17}\}\}$
k_{1b}	$\{\{a_{13}, a_{14}, b_{13}, b_{14}\}\}$
k_2	$\{\{a_{13}, b_{13}\}, \{a_{14}, b_{14}\}\}$
k_3	$\{\{a_{15}, a_{16}, b_{15}, b_{16}\}, \{a_{17}, b_{17}\}\}$
k_4	$\{\{a_{15}, b_{15}\}, \{a_{16}, b_{16}\}, \{a_{17}, b_{17}\}\}$
k_5	$\{\{a_{15}, a_{16}, b_{15}, b_{16}\}\}$
k_6	$\{\{a_{15}, b_{15}\}, \{a_{16}, b_{16}\}\}$
k_7	$\{\{a_5, b_5\}, \{a_6, b_6\}, \{a_7, b_7\}, \{a_8, b_8\}, \{a_9, b_9\}, \{a_{10}, a_{11}, b_{10}\}\}$
k_8	$\{\{a_4, a_5, b_5\}, \{a_6, b_6\}, \{a_7, b_7\}, \{a_8, b_8\}, \{a_9, b_9\}, \{a_{10}, b_{10}\}\}$
k_9	$\{\{a_5, b_5\}, \{a_6, b_6\}, \{a_7, b_7\}, \{a_8, b_8\}, \{a_9, b_9\}, \{a_{10}, b_{10}\}\}$
k_{11}	$\{\{a_1, a_2, b_1, b_2\}\}$
k_{12}	$\{\{a_1, a_2, b_1, b_2\}\}$
k_{13}	$\{\{a_1, b_1\}, \{a_2, b_2\}\}$

Table 7.1: The LK-partition pattern structure corresponding to the lattice given in Figure 7.2.

link key candidate w.r.t. its associated classes. Thus, for the evaluation, instead of defining the equivalence relation $=_k$ over the set of all the subjects of the datasets $S = S(D_1 \cup D_2)$, we define it on $I = I(c_1) \cup I(c_2)$ the set of all instances of the classes associated to $k = (Eq, In, (c_1, c_2))$. Indeed, the number of the equivalence classes in the partition is such an important indicator.

We mentioned above that singletons are omitted for readability reasons. Moreover, the presence of the singleton $\{z\}$ in a partition $I/(=_k)$ indicates that z is not linked to any other instance through k. However, the number of singletons can give us an idea of the proportion of instances that are effectively linked through a link key candidate. Thus we will distinguish the whole set $I/(=_k)$ including singletons and the set $I^{linked}/(=_k)$ without singletons, such that:

$$I^{linked}/(=_k) = \{z \in I/(=_k) \mid |z| > 1\}$$

For example, $I^{linked}/(=_{k_3}) = \{\{a_{15}, a_{16}, b_{15}, b_{16}\}, \{b_{17}, a_{17}\}\}$ and $|I^{linked}/(=_{k_3})| = 2$. While $I/(=_{k_3}) = \{\{a_{15}, a_{16}, b_{15}, b_{16}\}, \{b_{17}, a_{17}\}, \{a_{13}\}, \{a_{14}\}\}$, and $|I/(=_{k_3})| = 4$, with k_3 is a link key candidate, from Figure 7.2 associated with the classes (`Book,Novel`) and a_{13} and a_{14} are the instances from the class `Book` that are not linked by k_3.

We are now ready to define two new quality measures based on partitions. Let us consider a link key candidate k and the related sets of equivalence classes $I/(=_k)$ and $I^{linked}/(=_k)$. The partition size measure, denoted $pSize$, is the number of equivalence classes in $I^{linked}/(=_k)$:

$$pSize(k) = |I^{linked}/(=_k)|$$

The "Jaccard index", denoted $jpi(k)$ (p for partition), is a normalization of $pSize$ and measures the proportion of instances linked through k:

$$jpi(k) = \frac{pSize(k)}{|I/(=_k)|}$$

Given a link key candidate k, $pSize(k)$ ranges from 1, since k generates at least one link, to $min(|I(c_1)|, |I(c_2)|)$ when $L^+(k)$ is a 1-1 mapping. By contrast, the Jaccard partition index

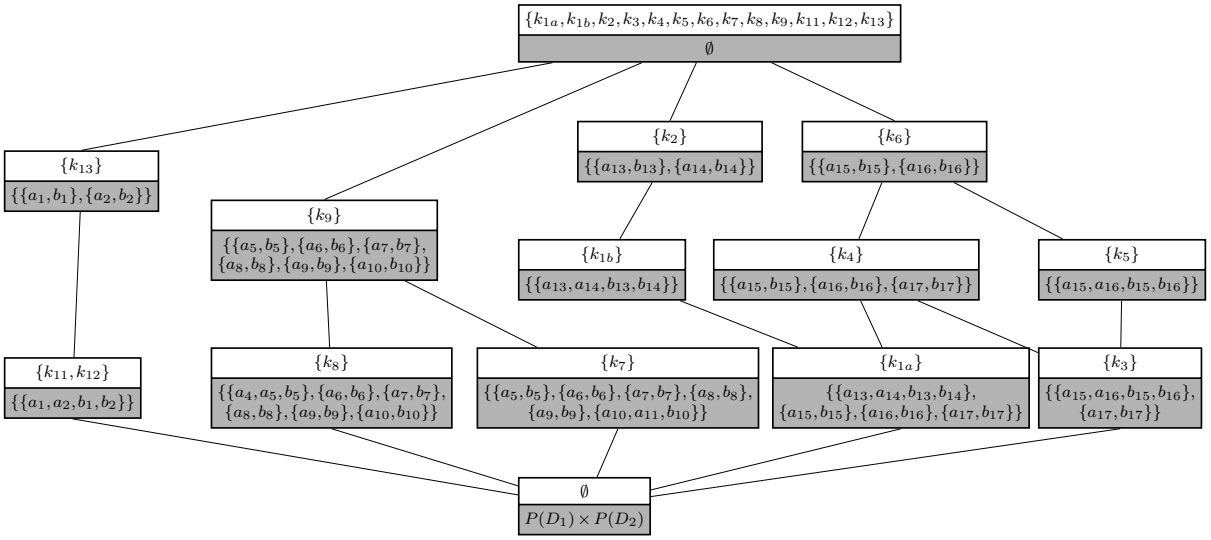


Figure 7.3: The OC-poset of the LK-partition pattern structure given in Table 7.1.

Candidate	$pSize$
k_3	2
k_4	3
k_5	1
k_6	2

Table 7.2: The $pSize$ of the candidates associated with the pair of classes (`Book,Novel`) depicted in Figure 7.2.

evaluates how close the classes c_1 and c_2 are w.r.t. $=_k$, $jpi(k)$ being maximal when all instances are linked.

For example, in Table 7.2, we give the $pSize$ of the candidates associated with the pair of classes (`Book,Novel`) depicted in Figure 7.2. By examining their harmonic mean of the coverage and discriminability score hm (in Figure 7.2), we can notice that the ranking of the candidates based on the $pSize$ aligns with that of the hm, which is a global measure as the $pSize$.

However, the Jaccard partition index may not always be satisfactory, especially when classes c_1 and c_2 are imbalanced in size. In this case, it is more accurate to maximize the weight of the "smallest" class as soon as all instances are linked. This is taken into account by the "overlap coefficient", also known as "Szymkiewicz–Simpson coefficient", and denoted here by $sspc(k)$. It can be calculated as

$$sspc(k) = \frac{pSize(k)}{min(|I(c_1)/(=_k)|, |I(c_2)/(=_k)|)}$$

$I(c_1)/(=_k)$ denotes the partition induced by $=_k$ over the set of instances of class c_1. For example, $I(\texttt{Book})/(=_{k_5}) = \{\{a_{15}, a_{16}\}, \{a_{13}\}, \{a_{14}\}, \{a_{17}\}\}$, stating that instances a_{15} and a_{16} in the class class `Book` are equivalent w.r.t. k_5. Then we have that:

$$sspc(k_5) = \frac{|\{\{a_{15}, a_{16}, b_{15}, b_{16}\}\}|}{min(|\{\{a_{15}, a_{16}\}, \{a_{13}\}, \{a_{14}\}\}|, |\{a_{17}\}\}|, |\{\{b_{15}, b_{16}\}, \{b_{17}\}\}|)} = \frac{1}{2}$$

Interlinking task	datasets	|triples|	|subjects|	|properties|	|LKC|	|NRLKC|	time
Actor	db:Actor yago:Actor	94 606 1 029 580	5 807 108 415	16 16	2 198	2 177	56s
Album	db:Album yago:Album	594 144 762 238	85 002 136 848	5 5	44	44	28s
Book	db:Book yago:Book	247 372 185 032	29 846 41 849	7 7	82	82	39s
Film	db:Film yago:Film	1 369 600 1 067 084	82 099 123 822	9 9	18 718	17 643	34m
Mountain	db:Mountain yago:Mountain	135 442 233 562	16 397 32 874	5 5	39	39	5m
Museum	db:Museum yago:Museum	15 940 163 342	1 826 21 050	7 7	48	48	9s
Organization	db:Organization yago:Organization	4 487 205 4 410 854	183 665 430 071	17 17	1 425	1 425	57m
Scientist	db:Scientist yago:Scientist	128 360 671 266	18 409 92 828	10 18	862	862	2m
University	db:University yago:University	241 838 263 624	10 352 23 334	9 9	213	213	56s

Table 7.3: DB-Yago datasets statistics, where |triples| denotes the number of triples in each dataset, |subjects| the number of instances, and |properties| the number of properties.

7.2.3 Experiments

In these experiments, our objectives are threefold:

1. Investigate the significance of partition-based redundancy among link key candidates in real-world datasets.

2. Compare the link-based quality measures with the newly introduced partition-based measures.

3. Assess the performance of link keys selected using the $pSize$ measure against other key-based interlinking approaches.

Datasets and Experimental Settings

We run experiments on DB-Yago datasets provided in [93]. These datasets are used by the approaches to which we compare our results in the last series of experiments. They are derived from DBpedia and Yago and organized into nine tasks. One particular task consists in finding links between instances of a class in DBpedia and instances of a class in Yago, e.g., `db:Actor` and `yago:Actor`. A set of reference links (i.e., owl:sameAs links) denoted by L^{ref} is provided for each task. The statistics of DB-Yago datasets are given in Table 7.3.

Experiments were run on a MacBook Pro 2018 with Intel Core i7-8850H@2,6 GHz, 16GB of RAM.

Redundancy of Link Key Candidates is not so Significant

In Table 7.3, column |LKC| represents the number of link key candidates discovered by the pattern structure algorithm, while column |NRLKC| represents the number of non redundant link key candidates discovered by the partition pattern structure algorithm. In most of the tasks, we can observe that |NRLKC| is equal to |LKC|, except for the tasks `Actor` and `Film` where |NRLKC| is

lower than |LKC| by 1% and by 5% respectively. This means that, in general, link key candidates produce different partitions and only a few are redundant. By contrast, if redundancy does not significantly reduce the number of link key candidates, it gives a good idea of the compactness of partitions related to link key candidates.

Nevertheless, even if redundancy is not so significant, we decided to experimentally compare the link-based measures, introduced in [7], against the partition-based measures that we introduce in this chapter.

Comparing Link-Based and Partition-Based Measures

This experiment compares the ability of quality measures to correctly rank the link key candidates. To do that, we compute the Kendall τ_b coefficient between the rankings provided by the different quality measures and the reference rankings provided a posteriori by precision and F-measure. For a set of reference links L^{ref}, precision is $pre(k) = |L(k) \cap L^{ref}|/|L(k)|$, recall is $rec(k) = |L(k) \cap L^{ref}|/|L^{ref}|$, and the F-measure is the harmonic mean of precision and recall. In particular, we perform two series of comparisons:

- The first series compares discriminability measures with precision *pre*. Here to distinguish between discriminability computed over $L(k)$ and the discriminability computed over $L^+(k)$, we denote the first one by *di* and the second one by di^+. Results are shown in Figure 7.4.

- The second series compares global quality measures, namely *hm* (harmonic mean of coverage and discriminability), *pSize* (partition size), *jpi* (Jaccard index), and *sspc* (overlap coefficient), with F-measure. The results are shown in Figure 7.5.

In Figures 7.4 and 7.5, the higher the τ_b coefficient of a measure m, the greater the ability of m to correctly rank link key candidates. The first series of experiments shows that *di* and di^+ have nearly the same τ_b coefficient values. The correlations are greater than 0.6 for seven tasks while the measures are poorly performing on the two tasks `Album` and `Scientist`. Then, it can be concluded that discriminability does not improve when considering `owl:sameAs` closure.

The second series of experiments shows that *pSize* and *hm* have similar Kendall τ_b scores in the three tasks `Film`, `Organization`, and `University`. In the other five tasks, *pSize* has a higher Kendall τ_b score than *hm*, e.g., by 21% in `Actor`. In `Album`, *pSize* has a lower score than *hm* while *hm* is surpassed by *jpi* and *sspc*. Then it can be concluded that *pSize* (partition-based) generally performs better than *hm* (link-based) w.r.t. the overall quality of link key candidates.

Link keys, Classical Keys and Conditional Keys

The *pSize* measure is used to evaluate and select the best link key candidates, –in this experiment the link key candidate ranked first– in every task listed in Table 7.3. Then, we compare recall, precision, and F-measure of the links generated by the best link key candidates selected by *pSize* against interlinking approaches providing classical keys and conditional keys as reported in [93].

In Figure 7.6 we can observe that: (i) recall of link keys is significantly better than recall of classical and conditional keys, (ii) precision of classical and conditional keys is slightly better than precision of link keys in most of the tasks, (iii) F-measure is much higher for selected link key candidates than for classical and conditional keys.

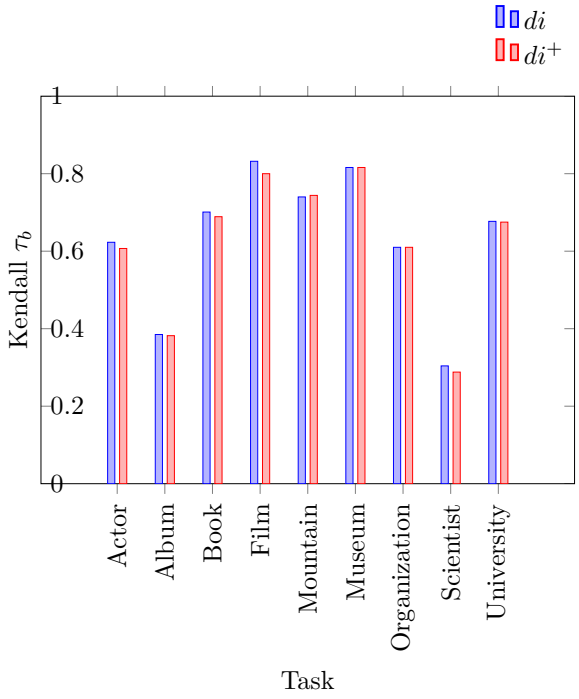


Figure 7.4: The rank correlation evaluated by Kendall τ_b between the ranking given by the precision of link keys and the ranking given by their correctness score measured by di and di^+. All p-values < 0.01.

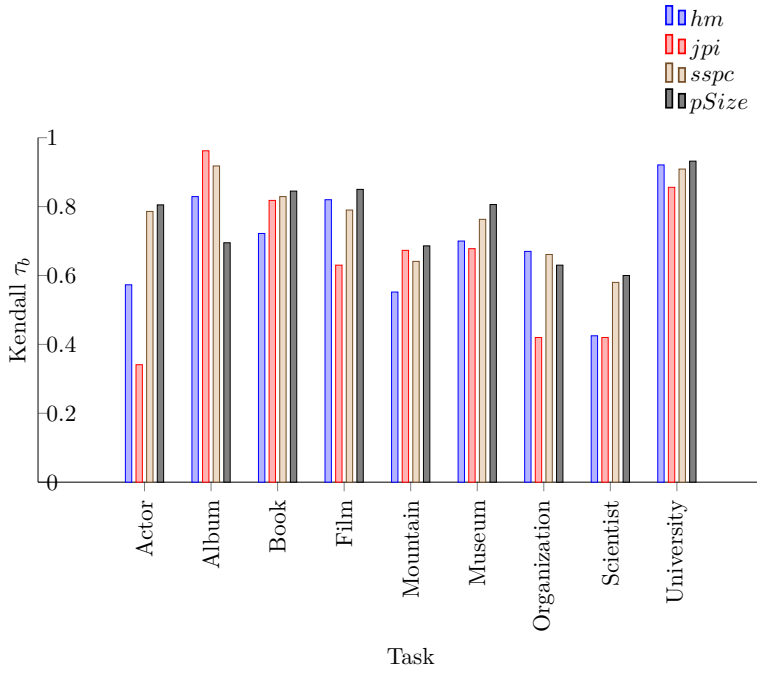


Figure 7.5: The rank correlation evaluated by Kendall τ_b between the ranking given by F-measure (recall, precision) of the link key candidates and the ranking given by global quality measures jpi, $sspc$, and $pSize$. All p-values are ≤ 0.01.

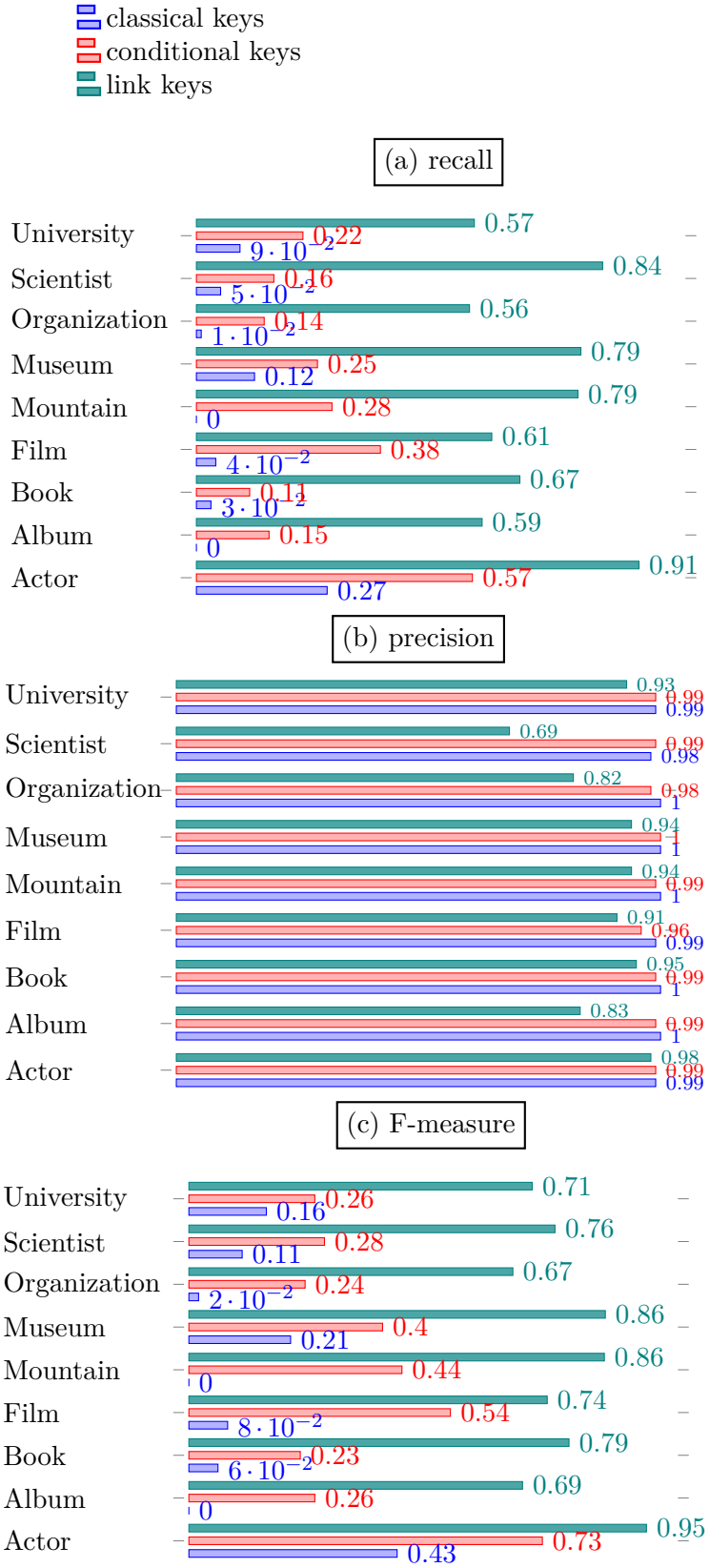


Figure 7.6: Interlinking performances of classical keys, conditional keys as given in [93] and link keys. A recall of 0 stands for a very small recall close to 0.

Link keys have a better recall because they are more flexible than classical keys, i.e., a link key is not necessarily a pair of keys and an instance of class c_1 may be linked to many instances of class c_2.

For summarizing, it can be concluded that considering the best link key candidates selected by $pSize$ will ensure a higher interlinking quality compared to classical and conditional keys. In addition, link keys do not require any prior knowledge such as property or class alignments comparing to key-based approaches.

We also observe that the best link key according to discriminability and coverage is the same as the best non redundant link key selected thanks to $pSize$ in these datasets. Nevertheless, the `Actor` dataset makes an exception: the link key candidate selected with $pSize$ obtains an F-measure of 0.95 contrasting the score of 0.34 obtained with coverage and discriminability.

7.3 Similarity-based Redundancy

In the previous section, we noted that partition-based redundancy is uncommon or non-existent in real-world datasets. Therefore, we aim to investigate if there exists another form of redundancy within the set of link key candidates. We start by providing a motivating example where we analyze the lattice of link key candidates shown in Figure 7.7. While examining this lattice, we observe that all the link key candidates produce distinct partitions, indicating an absence of partition-based redundancy. However, upon closer inspection of the link sets associated with candidates k_1, k_2, and k_3, we notice that they generate similar link sets, represented by the orange color in Figure 7.7. Hence, due to the fact that candidates k_1, k_2, and k_3 convey nearly identical information, we consider them as redundant. This type of redundancy is referred to as similarity-based redundancy.

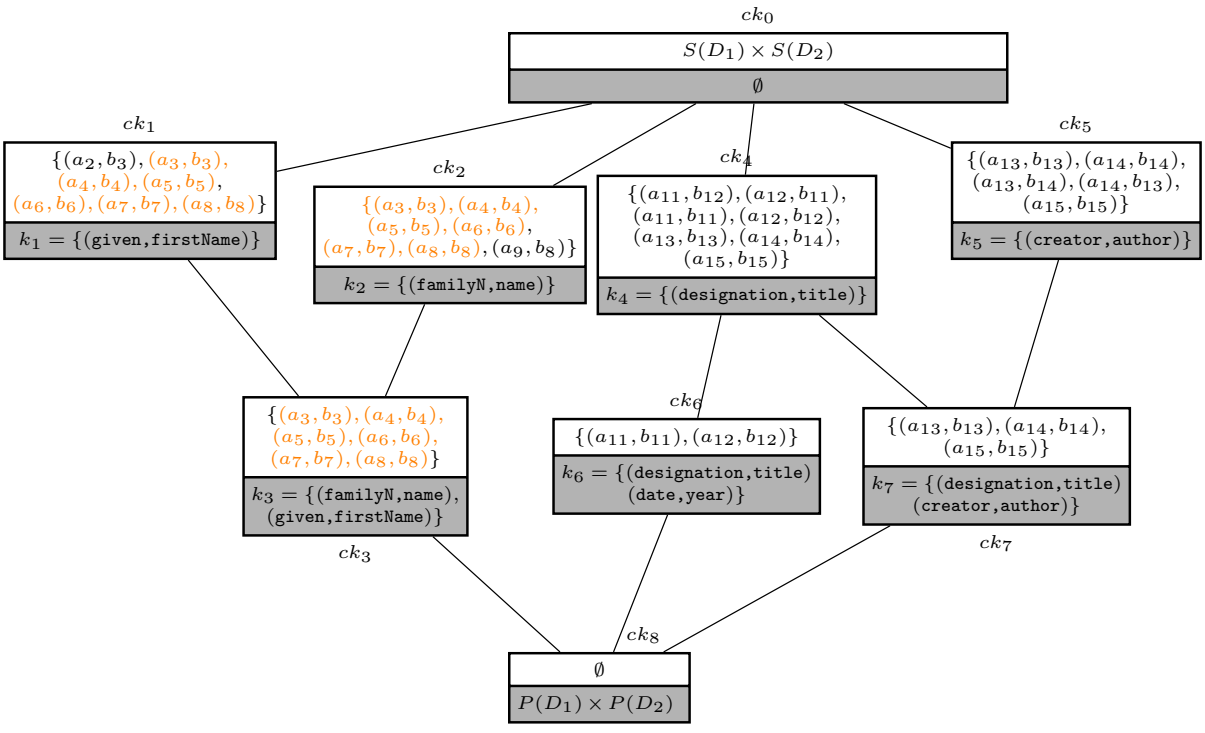
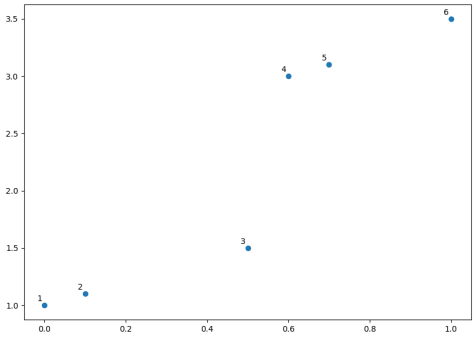


Figure 7.7: An example of a lattice of In-link key candidates.

In the following, we introduce a method for identifying redundancy, grouping redundant candidates into clusters, and subsequently selecting a representative from each cluster. This collection of selected representatives is referred to as the approximate basis of link keys, abbreviated as APPROX-LKB. To construct an APPROX-LKB, we propose the LK-CLUS algorithm, which utilizes agglomerative hierarchical clustering techniques [55, 66]. The details of the LK-CLUS algorithm will be outlined in the following sections.

Agglomerative Hierarchical Clustering

Hierarchical clustering aims at constructing a hierarchy of clusters based on a distance or a similarity index [55, 66]. Agglomerative hierarchical clustering follows a bottom-up strategy, starting with each data point as a singleton cluster and then successively merging pairs of clusters based on their similarity. The clustering stops when all clusters have been merged into one final cluster that contains all the data points.



$$\begin{array}{c} \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{pmatrix} 0.00 & 0.14 & 0.58 & 2.09 & 2.21 & 2.18 \\ 0.14 & 0.00 & 0.45 & 1.96 & 2.09 & 2.06 \\ 0.58 & 0.45 & 0.00 & 1.53 & 1.65 & 1.63 \\ 2.09 & 1.96 & 1.53 & 0.00 & 0.14 & 0.10 \\ 2.21 & 2.09 & 1.65 & 0.14 & 0.00 & 0.10 \\ 2.18 & 2.06 & 1.63 & 0.10 & 0.10 & 0.00 \end{pmatrix} \end{array}$$

Figure 7.8: On the left hand side, an example of input data points. On the right hand side, the distance matrix between the input data points with the Euclidean distance.

The distance between two clusters A and B depends on the distance d between the elements. It is usually defined by a linkage criterion as follows. The complete linkage (farthest neighbor) relies on the maximum distance between elements of each cluster, i.e., $distance(A,B) = \max_{a\in A, b\in B} d(a,b)$. The single linkage (nearest neighbor) relies on the minimum distance between elements of each cluster, i.e., $distance(A,B) = \min_{a\in A, b\in B} d(a,b)$. The average linkage relies on the mean distance between elements of each cluster, i.e., $distance(A,B) = \frac{1}{|A||B|}\sum_{a\in A}\sum_{b\in B} d(a,b)$. Other linkage criteria may be defined [66].

The ouput of an agglomerative hierarchical clustering can be visualized as a tree-based representation called a dendrogram, as illustrated in Figure 7.9. The clusters are determined by choosing a "cutting level" in the dendrogram. For example, cutting the dendrogram at the height 1.0 will provide two clusters, i.e., $A = \{0,1,2\}$ and $B = \{3,4,5\}$. Cutting the dendrogram at the height 0.4 provides three clusters, i.e., $A = \{0,1\}$, $B = \{3,4,5\}$, and $C = \{2\}$. The choice

of the cutting level mainly depends on the objectives of the clustering and the application [55].

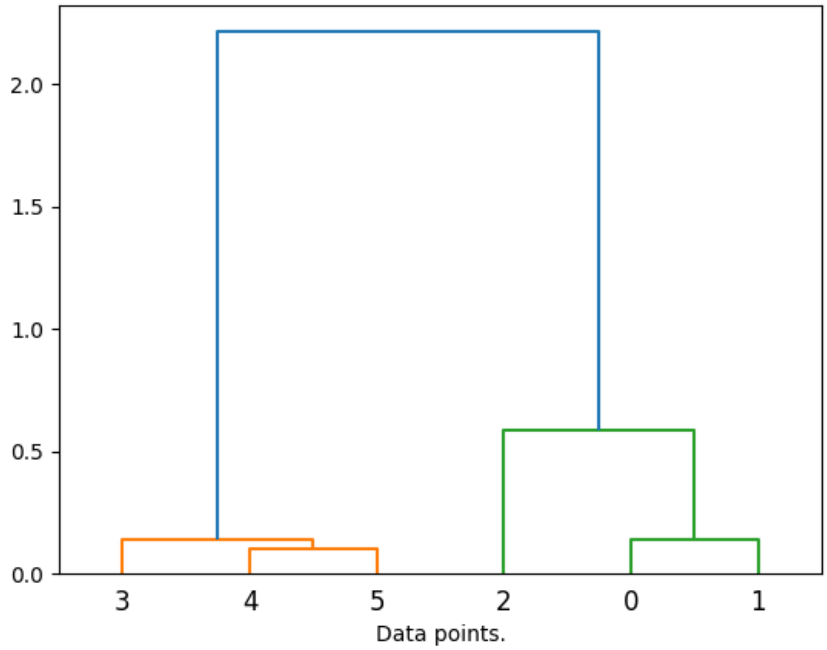


Figure 7.9: The dendrogram represents the output of the agglomerative hierarchical clustering of the data points in 7.8 based single linkage between-clusters.

7.3.1 Building an Approximate Basis of Link Keys

The Characteristics of an approx-lkb

The approximate basis of link keys APPROX-LKB contains a set of link key candidates that are non redundant w.r.t. the similarity-based redundancy. Accordingly, we define three characteristics for a "good" APPROX-LKB:

(i) *The compression related to the APPROX-LKB w.r.t. LKC.* In order to minimize redundancy, an APPROX-LKB should be minimal, i.e., it includes the smallest number of candidates, the smaller the number of the candidates the better is the "compression" in the APPROX-LKB. Given LKC the set of all link key candidates, this compression can be checked as follows:

$$\text{COMPRESS}\ (\text{APPROX-LKB},\text{LKC}) = 1 - |\text{APPROX-LKB}\ |/|\text{LKC}\ |$$

where the values are ranging from from 0 to 1 (excluded), best values being close to 1. The compression is equal to 0 when $|\text{APPROX-LKB}\ | = |\text{LKC}\ |$, i.e., there is no compression at all. The compression cannot be equal to 1 as APPROX-LKB cannot be empty.

For example, considering the $\text{APPROX-LKB} = \{k_3, k_6, k_7\}$ in Figure 7.10, we have that $|\text{LKC}\ | = 7$ and then the compression is $\text{COMPRESS} = 1 - 3/7 \simeq 0.57$.

(ii) *The proportion of preserved links.* An APPROX-LKB should preserve the maximum of identity links, i.e., the higher the number of identity links generated in APPROX-LKB the better the quality of the APPROX-LKB. The proportion of preserved links PPL is evaluated thanks to the formula:

$$\text{PPL (APPROX-LKB,LKC)} = |\bigcup_{k_i \in \text{APPROX-LKB}} L(k_i)| / |\bigcup_{k_j \in \text{LKC}} L(k_j)|.$$

It should be noticed that, thanks to the definition of a link key candidate, $\text{LKC} \neq \emptyset$ and as well $L(k_j) \neq \emptyset$. The proportion of preserved links PPL ranges in $]0,1]$, cannot be equal to zero, and is equal to 1 when all identity links are preserved.

(iii) *The representativeness of the APPROX-LKB.* Ideally, every link key candidate should be represented in an APPROX-LKB. A representative candidate say k_r should "represent" a set of link key candidates, say $k_i, 1 \leq i \leq N$. Then the set of identity links generated by k_r should include the largest proportion of identity links related to the represented candidates.

For example, an APPROX-LKB whose elements represent candidates generating links involving instances of all classes in D_1 an D_2, i.e., `Woman, Scientist` and `Book` in D_1, and `FemaleScientist, Dictionary` and `Novel` in D_2, is better than an APPROX-LKB whose elements only represent candidates generating links between `Book` and `Dictionary`. Moreover, it is preferable that a representative candidate represents more than only one link key candidate.

The "representativeness degree" of an APPROX-LKB, denoted by $\text{RDEG}(\text{APPROX-LKB}, \text{LKC})$, depends on the distance d between candidates, and is computed as follows:

$$\text{RDEG (APPROX-LKB,LKC,d)} = 1 - 1/|\text{LKC}| \sum_{k \in \text{LKC}} \min_{k_i \in \text{APPROX-LKB}} (d(k, k_i)).$$

Not all these three characteristics can be maximized simultaneously but a good balance can be achieved. It should be noticed that given an LK-lattice, the corresponding APPROX-LKB is not unique. In the following section, we make precise the process of building an APPROX-LKB starting with an LK-lattice and using agglomerative clustering.

The Construction of the approx-lkb with the lk-clus Process

In this section we make precise the process called LK-CLUS of building a "good" approximate basis of link keys using agglomerative hierarchical clustering. As input the process takes an LK-lattice, a distance measure d, a cutting level, and a linkage criterion.

lk-clus-1.

Firstly, the LK-CLUS process computes the distance matrix between link key candidates. The dissimilarity $dk(k_i, k_j)$ between two candidates k_i and k_j is computed thanks to the Jaccard index between the respective link sets $L(k_i)$ and $L(k_j)$ as follows:

$$dk(k_i, k_j) = 1 - |L(k_i) \cap L(k_j)| / |L(k_i) \cup L(k_j)|$$

where $dk(k_i, k_j) \in [0, 1]$.

Two distinct candidates k_i and k_j always verify that $dk(k_i, k_j) \neq 0$ because they cannot generate the same set of links. When $dk(k_i, k_j) = 1$, this means that dissimilarity is maximal, i.e., $L(k_i)$ and $L(k_j)$ are disjoint set of links and there is no subsumption relation between k_i and k_j (the lower bound of k_i and k_j in the lattice is $\perp$). For example this is the case for k_3 and k_7 in Figure 7.7 and we have that $dk(k_3, k_7) = 1$.

By contrast, let us compute now $dk(k_1, k_3)$, i.e. the dissimilarity between k_1 and k_3, knowing that there is a subsumption relation between k_1 and k_3. The two link sets are:
$L(k_1) = \{(a_2, b_3), (a_3, b_3), (a_4, b_4), (a_5, b_5), (a_6, b_6), (a_7, b_7), (a_8, b_8)\}$,
$L(k_3) = \{(a_3, b_3), (a_4, b_4), (a_5, b_5), (a_6, b_6), (a_7, b_7), (a_8, b_8)\}$
Then:
$dk(k_1, k_3) = 1 - |L(k_1) \cap L(k_3)| / |L(k_1) \cup L(k_3)| = 1 - 6/7 = 0.14$

lk-clus-2.

Based on the distance matrix and a selected linkage criterion (here complete linkage), the LK-CLUS process performs an agglomerative hierarchical clustering and outputs a dendrogram. Figure 7.10 shows an example of dendrogram associated with the LK-lattice.

lk-clus-3.

The next steps consists in choosing a "cutting level" in the dendrogram and to identify the related clusters. Every cluster consists of a set of link key candidates generating "approximatively" the same set of links.

For example, setting a cutting level to 0.7 and complete linkage criterion, the LK-CLUS process returns three clusters, namely: $cluster_1 = \{k_4, k_5, k_7\}$, $cluster_2 = \{k_6\}$, and $cluster_3 = \{k_1, k_2, k_3\}$.

lk-clus-4.

In this last step of the LK-CLUS process, a "representative" of the link key candidates lying in the cluster should be selected. In this study, the representative that is selected in each cluster corresponds to the "medoid" of the cluster, i.e., the candidate which minimizes the sum of the distances to all elements in the cluster [55]. In the following, we keep the medoid as the representative of a cluster but it should be noticed that other choices of a representative are possible.

Let $K = \{k_1, ..., k_n\}$ be a set of link key candidates in the cluster K and dk the associated dissimilarity measure. The medoid of the cluster K is defined as:

$$medoid(K) = argmin_{k \in K} \sum_{i=1}^{n} dk(k, k_i)$$

For example, in Figure 7.10, when the cutting level is set to 0.7, the LK-CLUS process returns the following representatives in each cluster, namely $medoid(cluster_1) = k_7$, $medoid(cluster_2) = k_6$, and $medoid(cluster_3) = k_3$. This set of medoids forms the APPROX-LKB, $\text{APPROX-LKB}_{0.7} = \{k_3, k_6, k_7\}$.

Cutting the dendrogram at different levels will identify different clusters and thus different APPROX-LKB. For example, if the cutting level is set to 0.5 in the previous dendrogram, four

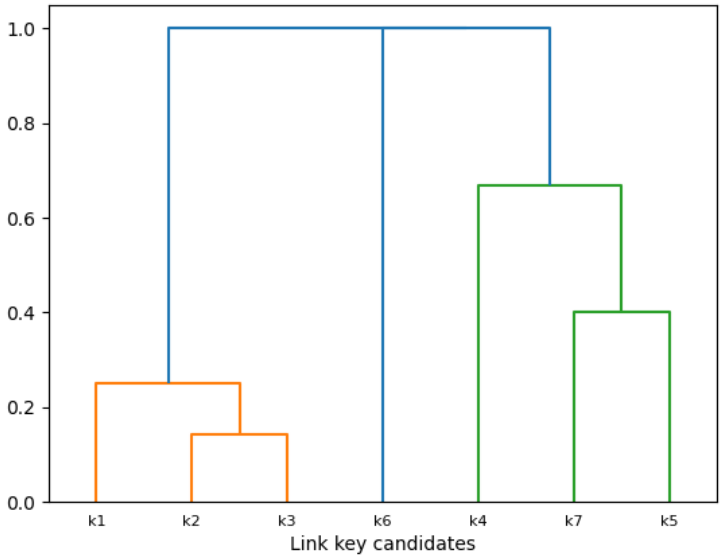


Figure 7.10: The dendrogram associated with the LK-lattice given in Figure 7.7.

APPROX-LKB	compression	link preservation	representativeness
APPROX-LKB$_{0.7}$	0.571	0.647	0.82
APPROX-LKB$_{0.5}$	0.429	0.765	0.902

Table 7.4: The comparison between the APPROX-LKB APPROX-LKB$_{0.7}$ and APPROX-LKB$_{0.5}$.

clusters, namely $\{k_1, k_2, k_3\}$, $\{k_4\}$, $\{k_6\}$, and $\{k_5, k_7\}$, will be considered. Then we will have that be APPROX-LKB$_{0.5} = \{k_3, k_4, k_6, k_7\}$.

Actually, the choice of the cutting level has a direct influence on the construction of the APPROX-LKB and on its quality. For example, a comparison of the two APPROX-LKB APPROX-LKB$_{0.7}$ and APPROX-LKB$_{0.5}$ is proposed in Table 7.4. In this comparison, one can see that APPROX-LKB$_{0.5}$ dominates APPROX-LKB$_{0.7}$ within two dimensions, i.e. preservation and representativeness. In this way, it could be concluded that the best APPROX-LKB is reached with a cutting level of 0.5. This shows that the APPROX-LKB is not unique and that several APPROX-LKB may be relevant, depending on the objectives of the experiments at hand. Accordingly, in the next section, we present a series of experiments and we study how the different choices are affecting the quality the APPROX-LKB that is constructed.

7.3.2 Experiments

We evaluate link key discovery based on LK-CLUS with two series of experiments. Firstly, we show that LK-CLUS is able to return good quality APPROX-LKB starting from an LK-lattice, and we also discuss how the cutting level affects this quality. The second series of experiments evaluates the potential of medoids and LK-CLUS w.r.t. precision and recall. The results of these

second experiments confirm the good capabilities of the LK-CLUS approach in link key discovery.

Datasets and Protocol

The experiments are performed using OAEI datasets and two datasets from the library domain: (1) The `Restaurants`, `Person1`, and `Person2` from OAEI 2010 datasets. (2) The Doremus datasets about cultural institutions from OAEI 2016. (3) The SPIMBench Sandbox from OAEI 2018. (4) Libraries datasets provided by two French libraries, namely "Bibliothèque Nationale de France" (BnF, `https://data.bnf.fr/`) and "Agence Bibliographique de l'Enseignement Supérieur" (Abes `https://www.idref.fr/`). Statistics about these datasets are provided in Table 7.5. Any pair of datasets from OAEI relies on the same ontology/schema while the two datasets from the library domain depend on different ontologies.

The LK-lattices are generated thanks to `Linkex` which performs a basic normalization of data values and deals with property's composition. The column #candidates in Table 7.5 represents the number of link key candidates discovered for each task (a task consists in considering two datasets and then to discover link key candidates).

Task	datasets	#instances	#properties	#triples	#candidates
Restaurants	Restaurant1	339	7	1 130	22
	Restaurant2	2 256	7	7 520	
Person1	Person11	2 000	14	9 000	535
	Person12	1 000	13	7 000	
Person2	Person21	2 400	14	10 800	469
	Person22	800	13	5 600	
Doremus1	PP-1	797	52	2 530	20
	BnF-1	692	48	2 189	
Doremus2	PP-2	4 053	52	12 757	83
	BnF-2	3 384	54	10 622	
Doremus3	PP-3	940	52	2 970	27
	BnF-3	822	53	2 610	
SPIMBench	Abox1	1 126	47	10 001	5 589
	Abox2	1 130	67	10 022	
Libraries	BnF	78 076	414	989 278	216
	Abes	290 247	128	1 864 636	

Table 7.5: The datasets' statistics

The Evaluation of an approx-lkb

In the first series of experiments, we evaluate the quality of an APPROX-LKB in terms of compression, number of preserved links, and representativeness. To do that, we run LK-CLUS on the LK-lattice associated with each task, using the complete linkage criterion and we varied the cutting level for obtaining different APPROX-LKB. In every case, we compute the compression of the APPROX-LKB, the number of preserved links, and the representativeness degrees. Moreover, we also carried out the same experiment with single linkage and average linkage criteria. We do not present the results here because they are very close to the results returned with the complete linkage criterion. The dendrogram related to each task is depicted n Figure 7.13;

As expected, results presented in Figure 7.11 show that the compression increases and the link preservation decreases when the cutting level increases. However, contrasting compression, link preservation is quite stable at the beginning and then dramatically decreases when cutting level varies between 0.7 and 0.9. The representativeness decreases slower than the cutting level and than the compression ratio. This shows that LK-CLUS strongly reduces the number of

Task	dis. threshold	selection	recall	precision	F-measure
Restaurants	1	LK-CLUS	0.63	0.8	0.71
		BASELINE	0.74	0.04	0.08
	0.5	LK-CLUS	0.74	0.7	0.72
		BASELINE	0.74	0.04	0.077
Person1	1	LK-CLUS	1	0.96	0.98
		BASELINE	1	0.49	0.66
	0.5	LK-CLUS	1	0.49	0.66
		BASELINE	1	0.49	0.66
Person2	1	LK-CLUS	0.36	0.91	0.52
		BASELINE	1	0.06	0.11
	0.5	LK-CLUS	0.97	0.71	0.82
		BASELINE	1	0.03	0.06
Doremus1	1	LK-CLUS	0.34	1	0.51
		BASELINE	0.81	0.36	0.5
	0.5	LK-CLUS	0.78	0.81	0.79
		BASELINE	0.84	0.36	0.51
Doremus2	1	LK-CLUS	0.24	0.78	0.37
		BASELINE	0.91	0.04	0.08
	0.5	LK-CLUS	0.77	0.81	0.79
		BASELINE	0.91	0.04	0.08
Doremus3	1	LK-CLUS	0.34	0.82	0.48
		BASELINE	0.93	0.12	0.21
	0.5	LK-CLUS	0.83	0.56	0.67
		BASELINE	0.93	0.12	0.21
SPIMBench	1	LK-CLUS	0.49	0.76	0.6
		BASELINE	0.63	0.48	0.55
	0.5	LK-CLUS	0.63	0.51	0.56
		BASELINE	0.63	0.48	0.55
Libraries	1	LK-CLUS	0	0.05	0
		BASELINE	0.73	0.1	0.18
	0.5	LK-CLUS	0.44	0.46	0.45
		BASELINE	0.73	0.01	0.03

Table 7.6: The comparison between LK-CLUS and the BASELINE selection w.r.t recall, precision and F-measure.

link key candidates without loosing too much information. In the worst case, we can obtain a compression ratio of at least 50% without loosing any link.

In every task, the APPROX-LKB produced from an LK-lattice is of good quality. For example, considering the task `SPIMBench`, the LK-lattice contains 5589 link key candidates with a representativeness threshold of 0.73, LK-CLUS returns an APPROX-LKB compressing the lattice by 92% while preserving 86% of the links. This shows the potential of LK-CLUS to reduce a set of candidates while ensuring a high degree of link preservation.

The Evaluation of the Links lying in an approx-lkb

In this second series of experiments, we test the quality of the links generated by link key candidates in the APPROX-LKB discovered by LK-CLUS. To that extent, we compare the links generated by APPROX-LKB against available gold standards and we compute the associated F-measure.

In a first experiment, we test whether medoids are the best link keys in a cluster in terms of F-measure. We compute the mean of the differences between the best F-measures among candidates in a cluster say `CL` and the F-measure of the medoid in `CL`. The green curve on Figure 7.11 shows that this difference remains low when the cutting level is varied. This supports the hypothesis that medoids are, in average, always close to the best candidates in a cluster.

Moreover, there is no reason that all link keys in an APPROX-LKB are good link keys in terms

of F-measure. Actually, LK-CLUS builds APPROX-LKB, where good and bad link keys w.r.t. F-measure may be present. Thus we want to effectively check the quality of candidates in an APPROX-LKB w.r.t. F-measure. Accordingly, in this second experiment, we select all candidates in an APPROX-LKB with a discriminability score higher than a given threshold. We compute the recall and the precision of the union of the links generated by the selected candidates. These results are then compared to those returned by a so-called "baseline strategy" that selects the same number of link key candidates in the whole lattice ordered thanks to harmonic mean.

Table 7.6 presents the results for APPROX-LKB having a representativeness degree above 90% and maximizing the compression. The selection strategy is tested with two discriminability thresholds, i.e., 1 and 0.5. The F-measure for LK-CLUS is significantly better than for BASELINE, except for `Libraries` in which good candidates have a discriminability lower than the threshold 1. We also observe that the discriminability threshold affects the performance of LK-CLUS. For example, considering `Doremus2`, F-measure is equal to 0.37 with a threshold of 1 and to 0.79 with a threshold of 0.5.

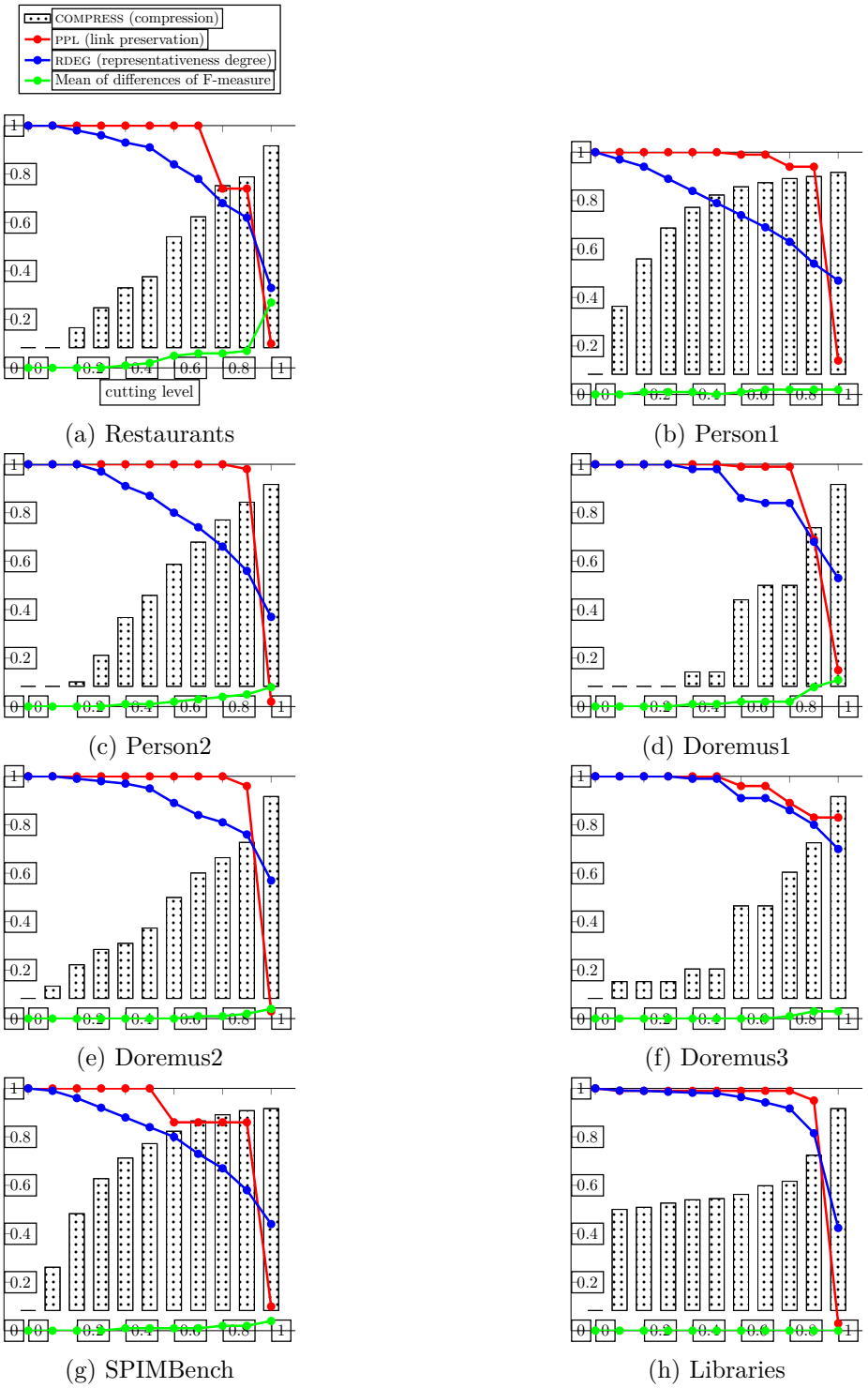
7.4 Conclusion and Discussion

Discovery algorithms designed to identify link key candidates typically produce a large number of them, which poses a challenge for domain experts to validate the most suitable ones. To simplify the selection process and address this issue, our work presents two methods for identifying redundant link key candidates.

The first method involves detecting partition-based redundancy by utilizing the owl:sameAs predicate, which serves as an equivalence relation to define partitions related to each candidate's link set. By identifying candidates that generate the same partitions, we classify them as redundant. To identify and reduce this redundancy, we introduce Partition Pattern Structures, where the link key candidates act as objects and the induced partitions serve as descriptions. The corresponding lattice of this Partition Pattern Structures merges redundant link key candidates into a single pattern concept, effectively reducing redundancy.

However, since real datasets tend to exhibit less significant partition-based redundancy, we propose an alternative approach based on similarity-based redundancy. In this approach, link key candidates that generate similar link sets are considered redundant. To identify and utilize this form of redundancy, we propose a method based on agglomerative hierarchical clustering, which groups redundant candidates into clusters and selects a representative candidate from each cluster. Through experiments conducted on various RDF datasets, we demonstrate that these approaches result in a concise set of link key candidates with minimal loss of links.

Regarding future perspectives, leveraging the lattice, we can optimize the clustering algorithm which computes the similarity between all link key candidates. In the LK-lattice, if a candidate is dissimilar to its immediate neighbor, it implies that it is also dissimilar to the neighbors of this neighbor. Additionally, we aim to explore the discovery of non-redundant link key candidates directly from RDF datasets, without the need to first identify redundant ones using Formal Concept Analysis, and then discover the non-redundant candidates. Finally, this method can be generalized to summarize any given formal concept lattice.



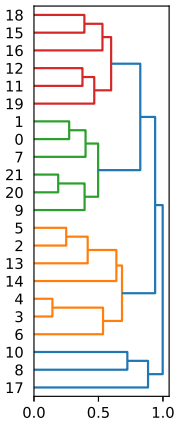
(a) Restaurants (b) Person1

(c) Person2 (d) Doremus1

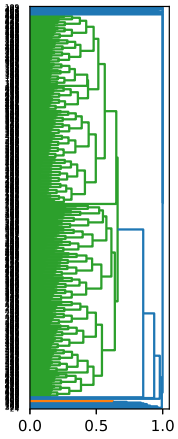
(e) Doremus2 (f) Doremus3

(g) SPIMBench (h) Libraries

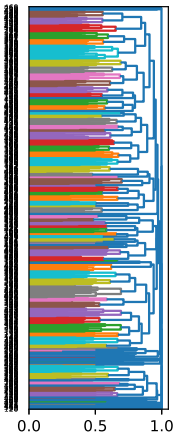
Figure 7.11: The variation of the compression, link preservation, the representativeness and the mean of differences of the F-measures w.r.t the cutting level.



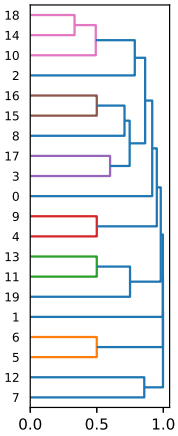
(a) Restaurants



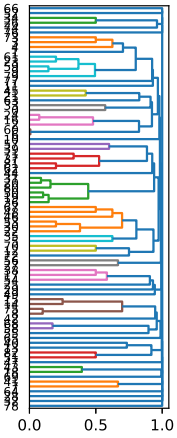
(b) Person1



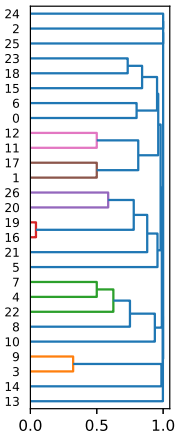
(c) Person2



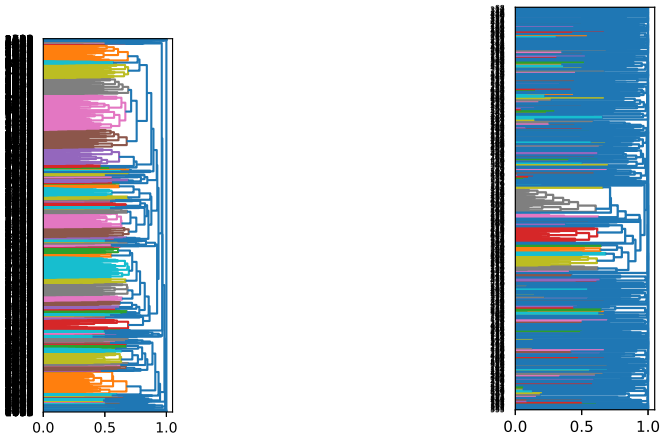
(d) Doremus1



(e) Doremus2



(f) Doremus3



(a) SPIMBench (b) Libraries

Figure 7.13: The dendrograms of link key candidates of each task obtained thanks to [79].

Chapter 8

Conclusion and perspectives

Contents

8.1 Summary of Contributions 95

8.2 Perspectives 96

8.1 Summary of Contributions

This thesis explores the data interlinking task within the Web of Data, with a primary objective to discover link keys that generate identity links across RDF datasets.

Link keys can be considered as a generalization of keys applied to different datasets. They provide an explicit and unambiguous method of linking individuals. The use of link keys can be particularly advantageous because they function as rules, directly determining whether newly added individuals in the datasets are the same. Additionally, one of the significant advantages of link keys lies in their application as logical axioms in ontologies, which involves handling semantics. Moreover, link keys are applicable in data interlinking scenarios that involve diverse ontologies or when no ontology alignment is accessible. This flexibility allows link keys to be utilized in a broader range of situations.

In this thesis, we have made several contributions to the discovery of link keys. Firstly, we have proposed a method to discover link key candidates while explicitly specifying their associated classes. Secondly, we have improved the selection process for link key candidates. Lastly, we have proposed methods for detecting and reducing the redundancy among link key candidates. The following is a summary of the contributions made in this thesis.

Our initial contribution introduces a Pattern Structure for the discovery of link key candidates. This method brings significant improvements to the candidate discovery process in various aspects. Firstly, it enables the identification of link key candidates while explicitly specifying their pairs of classes. This enables a more precise evaluation of the candidates and enhances the effectiveness of reasoning. Secondly, the definition of a link key candidate has been extended from being associated solely with a pair of atomic classes to being associated with a pair of conjunctions and disjunctions of classes. This extension allows for the consideration of different levels of abstraction within the datasets.

Then, we focused on the selection of relevant link key candidates, leveraging the proposed Pattern Structure. To accomplish this, we put forth two methods:The first method, based on the LKSA algorithm, is guided by the pairs of classes associated with the link key candidates.

This algorithm discovers candidates that generate identity links between all relevant classes in the datasets while discarding candidates associated to irrelevant pairs of classes, thus avoiding erroneous links. The second method takes guidance from the lattice of link key candidates. It operates on the intuition that the most "complete" candidates reside in the upper part of the lattice, while the most "correct" ones are situated at the bottom. Consequently, link key candidates that satisfy both criteria are typically found in the "middle" of the lattice. This approach is based on the SANDWICH algorithm, which follows a pruning strategy incorporating both "bottom-up" and "top-down" approaches. It identifies link keys that strike the best balance between completeness and correctness.

Finally, our attention was directed towards tackling the issue of redundancy among the link key candidates. By reducing this redundancy, we can obtain a smaller set of link key candidates, thereby simplifying the analysis, evaluation, and selection process for analysts. Consequently, we introduced two methods aimed at identifying and minimizing this redundancy. Initially, our approach involved examining the partition induced by the owl:sameAs relation among the link set of each link key candidate. Then candidates that generate the same partition are considered as redundant. Consequently, we introduced a Partition Pattern Structure, which allowed us to identify and merge link key candidates generating the same partitions. However, based on our experiments on real datasets, we observed that redundancy based on partitions was relatively rare. As a result, we proposed considering candidates generating similar link sets as redundant. To identify and minimize this redundancy, we developed an approach utilizing hierarchical clustering. Within this approach, candidates that produced similar link sets were grouped together into clusters. Subsequently, a representative for each cluster is selected. This effectively minimizes redundancy among the link key candidates.

8.2 Perspectives

In the associated chapters, we discussed the perspectives of each contribution. In the following, we aim to provide a global perspectives on our research work.

Atencia et al. [9] presented a study where they introduced various semantic categories for link keys, based on two dimensions: (i) how the link key handles multiple values of a specific property, and (ii) whether its underlying properties are considered keys. In our thesis, we specifically focus on addressing the problem of identifying weak link keys within this second dimension. Consequently, a significant perspective of our research involves developing an algorithm that can identify all potential types of link key candidates under these semantics, i.e. weak, strong and plain link keys. We propose to investigate the use of Partition Pattern Structure for this purpose.

In Chapter 4, we highlighted the utilization of extensional or instance-based methods in ontology matching, which can benefit from data interlinking [32, 85]. Alternatively, through a careful examination of the link key candidates discovered within datasets, we can identify alignments that could potentially contribute to ontology matching. For instance, the link key candidate $(\{(\texttt{given},\texttt{firstName})\},\{(\texttt{name},\texttt{familyName})\},(\texttt{Woman}\sqcap_{dl}\texttt{Scientist}),\texttt{FemaleScientist}))$ indicates an alignment between the properties `given` and `firstName`, as well as between `name` and `familyName`. Additionally, the class $\texttt{Woman}\sqcap_{dl}\texttt{Scientist}$ aligns with the class `FemaleScientist`. The semantics of such alignments need to be investigated, along with how the evaluation of link key candidates relates to the quality of these alignments. Consequently, link keys can be employed not only for identifying identity links but also for aligning properties and classes across datasets.

The link key discovery method proposed in this thesis diminishes the reliance on exter-

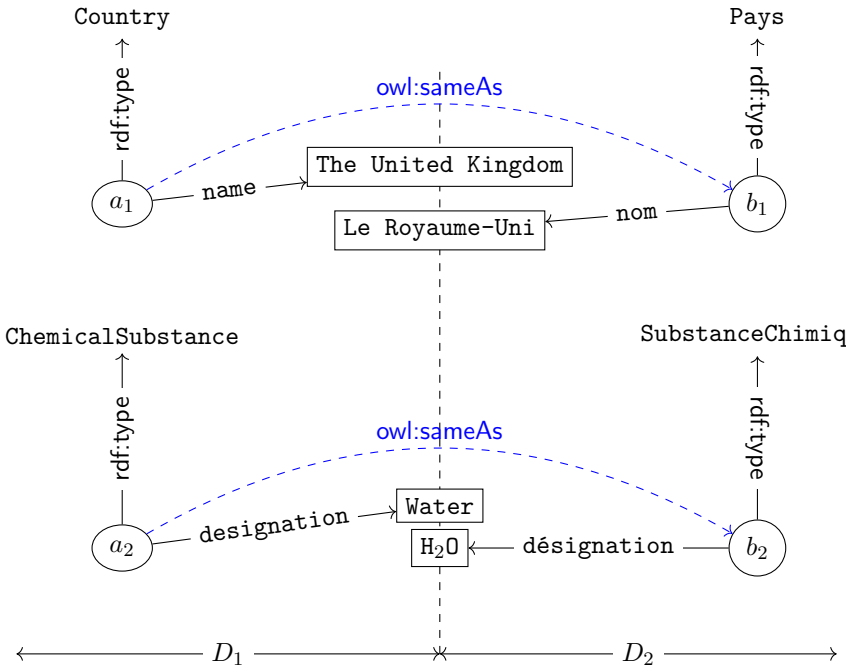


Figure 8.1: An example two datasets that do not have shared values but contain identity links.

nal knowledge. Notably, it eliminates the necessity of aligning properties and classes to discover link keys, which is a significant advantage. However, in datasets where the values are expressed differently, the discovery of link key candidates, based on this method, is not possible. Take Figure 8.1 as an example, where the link key $(\{(\texttt{name},\texttt{nom})\},\{\},(\texttt{Country},\texttt{Pays}))$ cannot be discovered using our method if the shared values are expressed in different languages. For instance, the value "`The United Kingdom`" is in English, while "`Le Royaume-Uni`" is in French. Moreover, values can also be expressed through synonyms such as "`famous`" and "`well-known`", or they may contain errors like "`fish `" and "`fishe`". Similarly, in Figure 8.1, the link key $(\{(\texttt{designation},\texttt{désignation})\},\{\},(\texttt{ChemicalSubstance},\texttt{SubstanceChimique}))$ cannot be discovered because water is given as a plain word in D_1, whereas in D_2, water is expressed by its chemical formula "$\texttt{H}_2\texttt{O}$". Other examples of the same values expressed differently are: "`Pfizer-BioNTech COVID-19 Vaccine`" and "`Pfizer-BioNTech`", "`25`" and "`Twenty-Five`", or "`over 18`" and "`20`". Therefore, there is a need for a value interlinking, i.e., identifying values that are the same. This can be achieved using external knowledge sources such as bilingual dictionaries for language translation, thesauri for synonym identification, and similarity functions for value comparison.

Furthermore, using link keys for data interlinking is not feasible when the values differ across datasets. In Figure 8.1, even if an expert provides the link key $(\{(\texttt{name},\texttt{nom})\},\{\},(\texttt{Country},\texttt{Pays}))$, the identity link $(a_1,\mathsf{owl{:}sameAs},b_1)$ would never be generated by this link key due to the difference in language used to express the values of properties `name` and `nom`. Consequently, there might be a need to enhance a link key by incorporating a similarity function, thereby giving rise to a link specification.

Redescription Mining (RM) [41] is a data mining technique that enables the discovery of multiple distinct characterizations, known as redescriptions, for the same entities. RM takes a set of entities as input, each described by different sets of attributes belonging to different views.

The output of RM is a collection of redescriptions, representing alternative descriptions for the same set of entities. To illustrate this, let's consider an example from [41] where the entities are geographical regions. These regions are described using two different views: (i) the species inhabiting these regions, and (ii) the climate of these regions. An example of a redescription in this context could be: *"The areas inhabited by either the Eurasian lynx or the Canada lynx are approximately the same areas as those where the maximum March temperature ranges from 24 degree C to 3.4 degree C"*. These redescription enable ecologists to examine the specific bioclimatic environments required by different mammal species. The application of Redescription Mining in the context of data interlinking has emerged as a natural question. Indeed, in the Web of Data, entities can be described in various datasets using different vocabularies. Hence, the challenge of link key discovery in Redescription Mining can be approached by considering the individuals as entities, each dataset as a view, and the redescriptions as potential link keys. However, this formalization is not straightforward as the identification of individuals representing the same entities is precisely the problem at hand. Formalizing link key discovery in the context of Redescription Mining remains a research question that we aim to investigate.

Privacy and security are of utmost importance in the Web of Data [56]. The capability to share and link data from diverse sources can lead to concerns regarding the potential misuse of personal and sensitive information. As a result, it is essential for data interlinking methods to carefully address this matter. Taking advantage of the flexibility of link keys, one can envision a scenario where a property is integrated into the link key to indicate whether the data provider has authorized the linking of their data.

Appendix A

List of Publications

Journal

Nacira Abbas, Alexandre Bazin, Jérôme David, and Amedeo Napoli. Discovery of link keys in resource description framework datasets based on pattern structures. International Journal of Approximate Reasoning, 161:108978, 2023.

Conferences (with program committee)

Nacira Abbas, Alexandre Bazin, Jérôme David, and Amedeo Napoli. A study of the discovery and redundancy of link keys between two RDF datasets based on partition pattern structures. In Pablo Cordero and Ondrej Krídlo, editors, Proceedings of the Sixteenth International Conference on Concept Lattices and Their Applications (CLA 2022) Tallinn, Estonia, June 20-22, 2022, volume 3308 of CEUR Workshop Proceedings, pages 175–189. CEUR-WS.org, 2022.

Nacira Abbas, Jérôme David, and Amedeo Napoli. LKSA : un algorithme de sélection de clés de liage dans des données RDF guidée par des paires de classes. In Jérôme Azé and Vincent Lemaire, editors, Extraction et Gestion des Connaissances, EGC 2021, 25-29 Janvier 2021, Montpellier, France, volume E-37 of RNTI, pages 205–216. Éditions RNTI, 2021 [6].

Nacira Abbas, Alexandre Bazin, Jérôme David, and Amedeo Napoli. Sandwich: An algorithm for discovering relevant link keys in an LKPS concept lattice. In Agnès Braud, Aleksey Buzmakov, Tom Hanika, and Florence Le Ber, editors, Proceedings of the International Conference on Formal Concept Analysis, ICFCA, volume 12733 of Lecture Notes in Computer Science, pages 243–251. Springer, 2021 [2].

Nacira Abbas, Jérôme David, and Amedeo Napoli. Discovery of Link Keys in RDF Data Based on Pattern Structures: Preliminary Steps. In Proceedings of International Conference on Concept Lattices and Their Applications, CLA, CEUR Workshop Proceedings 2668, pages 235–246, 2020.

Workshops (with program committee)

Nacira Abbas, Alexandre Bazin, Jérôme David, and Amedeo Napoli. Non-redundant link keys in RDF data: Preliminary steps. In Sergei O. Kuznetsov, Amedeo Napoli, and Sebastian Rudolph, editors, Proceedings of the 9th International Workshop "What can FCA do for Artificial Intelligence?" co-located with the 30th International Joint Conference on Artificial Intelligence

(IJCAI 2021), Montréal, Québec, Canada, August 21, 2021, volume 2972 of CEUR Workshop Proceedings, pages 125–130. CEUR-WS.org .

Nacira Abbas, Jérôme David, and Amedeo Napoli. Linkex: A tool for link key discovery based on pattern structures. In : ICFCA 2019-workshop on Applications and tools of formal concept analysis. 2019. p. 33-38.

International Semantic Web Research Summer School

Nacira Abbas, et al. Knowledge Graphs Evolution and Preservation–A Technical Report from ISWS 2019. arXiv preprint arXiv:2012.11936, 2020.

Bibliography

[1] Nacira Abbas, Alexandre Bazin, Jérôme David, and Amedeo Napoli. Non-redundant link keys in RDF data: Preliminary steps. In Sergei O. Kuznetsov, Amedeo Napoli, and Sebastian Rudolph, editors, *Proceedings of the 9th International Workshop "What can FCA do for Artificial Intelligence?" co-located with the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021), Montréal, Québec, Canada, August 21, 2021*, volume 2972 of *CEUR Workshop Proceedings*, pages 125–130. CEUR-WS.org, 2021.

[2] Nacira Abbas, Alexandre Bazin, Jérôme David, and Amedeo Napoli. Sandwich: An algorithm for discovering relevant link keys in an LKPS concept lattice. In Agnès Braud, Aleksey Buzmakov, Tom Hanika, and Florence Le Ber, editors, *Proceedings of ICFCA*, volume 12733 of *Lecture Notes in Computer Science*, pages 243–251. Springer, 2021.

[3] Nacira Abbas, Alexandre Bazin, Jérôme David, and Amedeo Napoli. A study of the discovery and redundancy of link keys between two RDF datasets based on partition pattern structures. In Pablo Cordero and Ondrej Krídlo, editors, *Proceedings of the Sixteenth International Conference on Concept Lattices and Their Applications (CLA 2022) Tallinn, Estonia, June 20-22, 2022., Tallinn, Estonia, June 20-22, 2022*, volume 3308 of *CEUR Workshop Proceedings*, pages 175–189. CEUR-WS.org, 2022.

[4] Nacira Abbas, Alexandre Bazin, Jérôme David, and Amedeo Napoli. Discovery of link keys in resource description framework datasets based on pattern structures. *International Journal of Approximate Reasoning*, 161:108978, 2023.

[5] Nacira Abbas, Jérôme David, and Amedeo Napoli. Discovery of Link Keys in RDF Data Based on Pattern Structures: Preliminary Steps. In *Proceedings of CLA*, CEUR Workshop Proceedings 2668, pages 235–246, 2020.

[6] Nacira Abbas, Jérôme David, and Amedeo Napoli. LKSA : un algorithme de sélection de clés de liage dans des données RDF guidée par des paires de classes. In Jérôme Azé and Vincent Lemaire, editors, *Extraction et Gestion des Connaissances, EGC 2021, 25-29 Janvier 2021, Montpellier, France*, volume E-37 of *RNTI*, pages 205–216. Éditions RNTI, 2021.

[7] Manuel Atencia, Jérôme David, and Jérôme Euzenat. Data interlinking through robust linkkey extraction. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 15–20. IOS Press, 2014.

[8] Manuel Atencia, Jérôme David, and Jérôme Euzenat. What can fca do for database linkkey extraction? In *3rd ECAI workshop on What can FCA do for Artificial Intelligence?(FCA4AI)*, pages 85–92. No commercial editor., 2014.

[9] Manuel Atencia, Jérôme David, and Jérôme Euzenat. On the relation between keys and link keys for data interlinking. *Semantic Web*, 12(4):547–567, 2021.

[10] 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.

[11] Manuel Atencia, Jérôme David, and François Scharffe. Keys and pseudo-keys detection for web datasets cleansing and interlinking. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 144–153. Springer, 2012.

[12] Manuel Atencia, Jérôme David, and Jérôme Euzenat. Several link keys are better than one, or extracting disjunctions of link key candidates. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 61–68. ACM.

[13] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, Daniele Nardi, et al. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[14] Franz Baader and Baris Sertkaya. Applying formal concept analysis to description logics. In *ICFCA*, pages 261–286. Springer, 2004.

[15] Jaume Baixeries, Mehdi Kaytoue, and Amedeo Napoli. Characterizing functional dependencies in formal concept analysis with pattern structures. *Annals of Mathematics and Artificial Intelligence*, 72:129–149, 2014.

[16] Matthias Baumgartner, Daniele Dell'Aglio, Heiko Paulheim, and Abraham Bernstein. Towards the web of embeddings: Integrating multiple knowledge graph embedding spaces with fedcoder. *Journal of Web Semantics*, 75:100741, 2023.

[17] Tim Berners-Lee. Linked data - design issues. `https://www.w3.org/DesignIssues/LinkedData.html`. Accessed: 2022-09-23.

[18] Tim Berners-Lee and Mark Fischetti. *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*. Harper San Francisco, 1999.

[19] Timothy J Berners-Lee. Information management: A proposal. Technical report, 1989.

[20] Anne Berry, Alain Gutierrez, Marianne Huchard, Amedeo Napoli, and Alain Sigayret. Hermes: a simple and efficient algorithm for building the aoc-poset of a binary relation. *Annals of Mathematics and Artificial Intelligence*, 72:45–71, 2014.

[21] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[22] Dan Brickley and R.V. Guha. RDF Schema 1.1. Recommendation, W3C, 2014. https://www.w3.org/TR/rdf-schema/.

[23] Vannevar Bush et al. As we may think. *The atlantic monthly*, 176(1):101–108, 1945.

[24] Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. *arXiv preprint arXiv:1611.03954*, 2016.

[25] Víctor Codocedo and Amedeo Napoli. A proposition for combining pattern structures and relational concept analysis. In *International Conference on Formal Concept Analysis*, pages 96–111. Springer, 2014.

[26] Richard Cyganiak, David, and Markus Lanthaler. RDF 1.1 concepts and abstract syntax. Recommendation, W3C, 2014. https://www.w3.org/TR/rdf11-concepts/.

[27] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[28] Harry Deutsch and Pawel Garbacz. Relative Identity. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2018 edition, 2018.

[29] Martin Dürst and Michel Suignard. Internationalized resource identifiers (iris). Technical report, 2005.

[30] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48(1-4):2, 2016.

[31] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, 19(1):1–16, 2006.

[32] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching, Second Edition*. Springer, 2013.

[33] M Fabian, Kasneci Gjergji, WEIKUM Gerhard, et al. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *16th International world wide web conference, WWW*, pages 697–706, 2007.

[34] Alfio Ferrara, Andriy Nikolov, and François Scharffe. Data linking for the semantic web. *Int. J. Semantic Web Inf. Syst.*, 7(3):46–76, 2011.

[35] Sébastien Ferré. A Proposal for Extending Formal Concept Analysis to Knowledge Graphs. In *Proceedings of the 13th International Conference on Formal Concept Analysis (ICFCA)*, LNCS 9113, pages 271–286. Springer, 2015.

[36] Sébastien Ferré. Exploring the application of graph-fca to the problem of knowledge graph alignment. In Pablo Cordero and Ondrej Krídlo, editors, *Proceedings of the Sixteenth International Conference on Concept Lattices and Their Applications (CLA 2022) Tallinn, Estonia, June 20-22, 2022., Tallinn, Estonia, June 20-22, 2022*, volume 3308 of *CEUR Workshop Proceedings*, pages 79–92. CEUR-WS.org, 2022.

[37] Sébastien Ferré, Marianne Huchard, Mehdi Kaytoue, Sergei O Kuznetsov, and Amedeo Napoli. Formal concept analysis: from knowledge discovery to knowledge processing. *A Guided Tour of Artificial Intelligence Research: Volume II: AI Algorithms*, pages 411–445, 2020.

[38] Sébastien Ferré and Olivier Ridoux. A logical generalizationof formal concept analysis. In *International Conference on Conceptual Structures*, pages 371–384. Springer, 2000.

[39] Sébastien Ferré and Peggy Cellier. Graph-FCA in Practice. In *Proceedings of the 22nd International Conference on Conceptual Structures (ICCS)*, LNCS 9717, pages 107–121. Springer, 2016.

[40] Peter Forrest. The Identity of Indiscernibles. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2020 edition, 2020.

[41] Esther Galbrun and Pauli Miettinen. *Redescription Mining*. Springer Briefs in Computer Science. Springer, 2017.

[42] André Gallois. *Occasions of identity: A study in the metaphysics of persistence, change, and sameness*. Oxford University Press, 2003.

[43] Fabien Gandon. Pour tout le monde: Tim berners-lee, lauréat du prix turing 2016 pour avoir inventé... le web. *1024: Bulletin de la Société Informatique de France*, (11), 2017.

[44] Bernhard Ganter and Sergei O. Kuznetsov. Pattern Structures and Their Projections. In *Proceedings of ICCS*, LNCS 2120, pages 129–142. Springer, 2001.

[45] Bernhard Ganter and Rudolf Wille. Conceptual scaling. In *Applications of combinatorics and graph theory to the biological and social sciences*, pages 139–167. Springer, 1989.

[46] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1999.

[47] Vijay K Garg. *Introduction to lattice theory with computer science applications*. John Wiley & Sons, 2015.

[48] The W3C SPARQL Working Group. SPARQL 1.1 overview. Recommendation, W3C, 2013. https://www.w3.org/TR/sparql11-overview/.

[49] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[50] Yanchao Hao, Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. A joint embedding method for entity alignment of knowledge bases. In *Knowledge Graph and Semantic Computing: Semantic, Knowledge, and Linked Big Data: First China Conference, CCKS 2016, Beijing, China, September 19-22, 2016, Revised Selected Papers 1*, pages 3–14. Springer, 2016.

[51] Tom Heath and Christian Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.

[52] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs. 2020.

[53] Yka Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal*, 42(2):100–111, 1999.

[54] Khadija Jradeh. *Optimised tableau algorithms for reasoning in the description logic ALC extended with link keys*. PhD thesis, Université Grenoble Alpes, 2022.

[55] Leonard Kaufman and Peter J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.

[56] Sabrina Kirrane, Serena Villata, and Mathieu d'Aquin. Privacy, security and policies: A review of problems and solutions with semantic web technologies. *Semantic Web*, 9(2):153–161, 2018.

[57] Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. A description logic primer. *arXiv preprint arXiv:1201.4089*, 2012.

[58] Sergei O. Kuznetsov and Tatiana P. Makhalova. On interestingness measures of formal concepts. *Inf. Sci.*, 442-443:202–219, 2018.

[59] Fritz Lehmann and Rudolf Wille. A triadic approach to formal concept analysis. In *International conference on conceptual structures*, pages 32–43. Springer, 1995.

[60] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[61] Hector J Levesque. Knowledge representation and reasoning. *Annual review of computer science*, 1(1):255–287, 1986.

[62] Deborah L. McGuinness and Frank van Harmelen. OWL web ontology language overview. Recommendation, W3C, 2004. https://www.w3.org/TR/owl-features/.

[63] Dean Van Der Merwe, Sergei Obiedkov, and Derrick Kourie. Addintent: A new incremental algorithm for constructing concept lattices. In *International Conference on Formal Concept Analysis*, pages 372–385. Springer, 2004.

[64] Boris Motik, Peter F. Patel-Schneider, and Bernardo Cuenca Grau. OWL 2 web ontology language: Direct semantics (second edition). Recommendation, W3C, 2012. http://www.w3.org/TR/owl2-direct-semantics.

[65] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 web ontology language structural specification and functional-style syntax (second edition). Recommendation, W3C, 2012. https://www.w3.org/TR/owl-syntax/.

[66] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: DMKD*, 2(1):86–97, 2012.

[67] Theodor Holm Nelson. Complex information processing: A file structure for the complex, the changing and the indeterminate. In *Proceedings of the 1965 20th National Conference*, ACM '65, page 84–100, New York, NY, USA, 1965. Association for Computing Machinery.

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

[69] Axel-Cyrille Ngonga Ngomo. Link discovery with guaranteed reduction ratio in affine spaces with minkowski measures. In *ISWC (1)*, pages 378–393, 2012.

[70] Axel-Cyrille Ngonga Ngomo and Sören Auer. Limes-a time-efficient approach for large-scale link discovery on the web of data. *integration*, 15(3), 2011.

[71] Axel-Cyrille Ngonga Ngomo, Jens Lehmann, Sören Auer, and Konrad Höffner. RAVEN - active learning of link specifications. In Pavel Shvaiko, Jérôme Euzenat, Tom Heath, Christoph Quix, Ming Mao, and Isabel F. Cruz, editors, *Proceedings of the 6th International Workshop on Ontology Matching, Bonn, Germany, October 24, 2011*, volume 814 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.

[72] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. Unsupervised learning of link specifications: deterministic vs. non-deterministic. In *Proceedings of the Ontology Matching Workshop*, 2013.

[73] Axel-Cyrille Ngonga Ngomo. Orchid–reduction-ratio-optimal computation of geo-spatial distances for link discovery. In *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I 12*, pages 395–410. Springer, 2013.

[74] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. EAGLE: Efficient Active Learning of Link Specifications Using Genetic Programming. In *Proceedings of ESWC*, Springer LNCS 7295, pages 149–163, 2012.

[75] Axel-Cyrille Ngonga Ngomo, Klaus Lyko, and Victor Christen. COALA - Correlation-Aware Active Learning of Link Specifications. In *Proceedings of ESWC*, Springer LNCS 7882, pages 442–456, 2013.

[76] Axel-Cyrille Ngonga Ngomo, Mohamed Ahmed Sherif, Kleanthi Georgala, Mofeed Hassan, Kevin Dreßler, Klaus Lyko, Daniel Obraczka, and Tommaso Soru. LIMES - A Framework for Link Discovery on the Semantic Web. *KI-Künstliche Intelligenz, German Journal of Artificial Intelligence - Organ des Fachbereichs "Künstliche Intelligenz" der Gesellschaft für Informatik e.V.*, 2021.

[77] Harold Noonan and Ben Curtis. Identity. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2022 edition, 2022.

[78] Bijan Parsia, Peter Patel-Schneider, and Boris Motik. OWL 2 web ontology language structural specification and functional-style syntax (second edition). W3C recommendation, W3C, December 2012. https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.

[79] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

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

[81] Jan Portisch, Nicolas Heist, and Heiko Paulheim. Knowledge graph embedding for data mining vs. knowledge graph embedding for link prediction–two sides of the same coin? *Semantic Web*, 13(3):399–422, 2022.

[82] Daniel Ringler and Heiko Paulheim. One knowledge graph to rule them all? analyzing the differences between dbpedia, yago, wikidata & co. In *KI 2017: Advances in Artificial Intelligence: 40th Annual German Conference on AI, Dortmund, Germany, September 25–29, 2017, Proceedings 40*, pages 366–372. Springer, 2017.

[83] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–49, 2021.

[84] Mohamed Rouane-Hacene, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. Relational concept analysis: mining concept lattices from multi-relational data. *Annals of Mathematics and Artificial Intelligence*, 67:81–108, 2013.

[85] François Scharffe and Jérôme Euzenat. Linked data meets ontology matching: enhancing data linking through ontology alignments. In *Proc. 3rd international conference on Knowledge engineering and ontology development (KEOD)*, pages 279–284. No commercial editor., 2011.

[86] August Th Schreiber, Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Nigel Shadbolt, Robert de Hoog, Walter Van de Velde, and Bob Wielinga. *Knowledge engineering and management: the CommonKADS methodology*. MIT press, 2000.

[87] Mohamed Ahmed Sherif, Axel-Cyrille Ngonga Ngomo, and Jens Lehmann. Wombat - A generalization approach for automatic link discovery. In *Proc. 14th European semantic web conference (ESWC), Portorož (SL)*, volume 10249 of *Lecture Notes in Computer Science*, pages 103–119. Springer, 2017.

[88] 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.

[89] Tommaso Soru and Axel-Cyrille Ngonga Ngomo. Rapid execution of weighted edit distances. In *Proceedings of the Ontology Matching Workshop*. Citeseer, 2013.

[90] Tommaso Soru and Axel-Cyrille Ngonga Ngomo. Active learning of domain-specific distances for link discovery. In *Joint International Semantic Technology Conference*, pages 97–112. Springer, 2012.

[91] Gerd Stumme and Alexander Maedche. FCA-MERGE: bottom-up merging of ontologies. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 225–234. Morgan Kaufmann, 2001.

[92] Danai Symeonidou, Vincent Armant, Nathalie Pernelle, and Fatiha Saïs. Sakey: Scalable almost key discovery in RDF data. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig A. Knoblock, Denny Vrandecic, Paul Groth, Natasha F. Noy, Krzysztof Janowicz, and Carole A. Goble, editors, *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, volume 8796 of *Lecture Notes in Computer Science*, pages 33–49. Springer, 2014.

[93] Danai Symeonidou, Luis Galárraga, Nathalie Pernelle, Fatiha Saïs, and Fabian M. Suchanek. VICKEY: mining conditional keys on knowledge bases. In d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange, and Jeff Heflin, editors, *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, volume 10587 of *Lecture Notes in Computer Science*, pages 661–677. Springer, 2017.

[94] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk-a link discovery framework for the web of data. *LDOW*, 538, 2009.

[95] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[96] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[97] Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 349–357, 2018.

[98] Rudolf Wille. The basic theorem of triadic concept analysis. *Order*, 12(2):149–158, 1995.

[99] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[100] Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Iterative entity alignment via joint knowledge embeddings. In *IJCAI*, volume 17, pages 4258–4264, 2017.

Résumé

Le Web des données est un espace de données global qui peut être considéré comme une couche supplémentaire au-dessus du Web des documents. Le liage des données est la tâche de découverte des liens d'identité entre les ensembles de données RDF (Resource Description Framework) sur le Web des données. Nous nous intéressons à une approche spécifique pour le liage des données, qui repose sur les "clés de liage". Cette clé a la forme de deux ensembles de paires de propriétés associées à une paire de classes. Par exemple, la clé de liage ({(designation,titre)},{(designation,titre),(createur,auteur)},(Livre,Roman)) indique que si une instance "a" de la classe "Livre" et "b" de la classe "Roman" partagent au moins une valeur pour les propriétés "createur" et "auteur" et que "a" et "b" ont les mêmes valeurs pour les propriétés "designation" et "titre", alors "a" et "b" désignent la même entité. Ainsi, (a, owl:sameAs, b) est un lien d'identité sur les deux ensembles de données. Cependant, les clés de liage ne sont pas toujours fournies, et divers algorithmes ont été développés pour découvrir automatiquement ces clés. Les algorithmes découvrent d'abord des "clés de liage candidates". La qualité de ces candidates est ensuite évaluée à l'aide de mesures appropriées, et les clés de liage valides sont sélectionnées en conséquence. L'Analyse Formelle des Concepts (AFC) a été étroitement associée à la découverte de clés de liage candidates, ce qui a conduit à la proposition d'un algorithme basé sur l'AFC à cette fin. Cependant, les algorithmes de découverte de clés de liage présentent certaines limitations. Premièrement, ils ne spécifient pas explicitement les paires de classes associées aux candidates découvertes, ce qui peut conduire à des évaluations inexactes. De plus, les stratégies de sélection utilisées par ces algorithmes peuvent également produire des résultats moins précis. On observe aussi une redondance parmi les ensembles de candidates découvertes, ce qui complique leur visualisation, évaluation et analyse. Pour remédier à ces limitations, nous proposons d'étendre les algorithmes existants sur plusieurs aspects. Tout d'abord, nous introduisons une méthode basée sur les Pattern Structures, une généralisation de l'AFC pour les données non binaires. Cette approche permet de spécifier explicitement les paires de classes associées à chaque clé de liage candidate. Deuxièmement, basée sur la Pattern Structure proposée, nous présentons deux méthodes de sélection de clés de liage. La première méthode est guidée par les paires de classes associées aux candidates, tandis que la deuxième méthode utilise le treillis générée par la Pattern Structure. Ces deux méthodes améliorent la sélection par rapport à la stratégie existante. Enfin, pour remédier à la redondance, nous introduisons deux méthodes. La première méthode est basée sur une Partition Pattern Structure, qui identifie et fusionne les candidates générant les mêmes partitions. La deuxième méthode est basée sur le clustering hiérarchique, qui groupe les candidates produisant des ensembles de liens similaires en clusters et sélectionne un représentant pour chaque cluster. Cette approche réduit efficacement la redondance parmi les clés de liage candidates.

Mots-clés: Découverte de clés de liage, Ensemble de données RDF, Liage des données, Pattern Structure, Mesures de qualité pour les clés de liage, Clustering.

Abstract

The Web of data is a global data space that can be seen as an additional layer interconnected with the Web of documents. Data interlinking is the task of discovering identity links

across RDF (Resource Description Framework) datasets over the Web of data. We focus on a specific approach for data interlinking, which relies on the "link keys". A link key has the form of two sets of pairs of properties associated with a pair of classes. For example the link key ({(designation,title)},{(designation,title),(creator,author)},(Book,Novel)), states that whenever an instance "a" of the class "Book" and "b" of the class "Novel", share at least one value for the properties "creator" and "author" and that, "a" and "b" have the same values for the properties "designation" and "title", then "a" and "b" denote the same entity. Then (a,owl:sameAs,b) is an identity link over the two datasets. However, link keys are not always provided, and various algorithms have been developed to automatically discover these keys. First, these algorithms focus on finding "link key candidates". The quality of these candidates is then evaluated using appropriate measures, and valid link keys are selected accordingly. Formal Concept Analysis (FCA) has been closely associated with the discovery of link key candidates, leading to the proposal of an FCA-based algorithm for this purpose. Nevertheless, existing algorithms for link key discovery have certain limitations. First, they do not explicitly specify the associated pairs of classes for the discovered link key candidates, which can lead to inaccurate evaluations. Additionally, the selection strategies employed by these algorithms may also produce less accurate results. Furthermore, redundancy is observed among the sets of discovered candidates, which presents challenges for their visualization, evaluation, and analysis. To address these limitations, we propose to extend the existing algorithms in several aspects. Firstly, we introduce a method based on Pattern Structures, an FCA generalization that can handle non-binary data. This approach allows for explicitly specifying the associated pairs of classes for each link key candidate. Secondly, based on the proposed Pattern Structure, we present two methods for link key selection. The first method is guided by the associated pairs of classes of link keys, while the second method utilizes the lattice generated by the Pattern Structure. These two methods improve the selection compared to the existing strategy. Finally, to address redundancy, we introduce two methods. The first method involves Partition Pattern Structure, which identifies and merges link key candidates that generate the same partitions. The second method is based on hierarchical clustering, which groups candidates producing similar link sets into clusters and selects a representative for each cluster. This approach effectively minimizes redundancy among the link key candidates.

Keywords: Link key discovery, RDF dataset, Data interlinking, Pattern Structure, Quality measures for link keys, Clustering.