UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Matching and mining in knowledge graphs of the Web of data Applications in pharmacogenomics

Appariement et fouille dans les graphes de connaissances du Web des données

Applications en pharmacogénomique

THÈSE

présentée et soutenue publiquement le 16 décembre 2020

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Pierre Monnin

Composition du jury

Présidente :	Marianne Clausel	Professeure, Université de Lorraine
Rapporteurs :	Claudia d'Amato	Associate Professor, University of Bari
	Fabien Gandon	Directeur de recherche, Inria
Examinateurs :	Céline Rouveirol	Professeure, Université Sorbonne Paris Nord
	Matthias Samwald	Associate Professor, Medical University of Vienna
Directeurs :	Amedeo Napoli	Directeur de recherche émérite, CNRS
	Adrien Coulet	Maître de Conférences HDR, Université de Lorraine

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

Mis en page avec la classe thesul.

Acknowledgments

Over the past years, I had the chance to meet, learn, work, or share moments of my (professional or personal) life with some persons I would like to acknowledge here. As with knowledge graphs of the Web of data, these acknowledgments may suffer from incompleteness. "Categories" are also not disjoint.

First of all, I would like to thank Claudia d'Amato and Fabien Gandon for reviewing my thesis. It was a privilege for me to receive their positive and detailed reports. Their insightful remarks offered me new perspectives on my work. I also thank Marianne Clausel, Céline Rouveirol, and Matthias Samwald for being my thesis examiners. I was deeply honored by the interest, questions, and encouragement of this committee of experts. I express my gratitude to all of them for making the remote defense as warm and human as possible. I hope future years will offer us many possibilities to meet and exchange ideas in-person.

I thank my supervisors, Amedeo Napoli and Adrien Coulet, for this thesis opportunity. They guided and supported me during this journey with unwavering enthusiasm, encouragement, and trust. This manuscript both ends a chapter of our work and opens some new perspectives that I hope to share with them.

I express a special thank to Miguel Couceiro for his interest, advice, knowledge, and cheerfulness. It was both a pleasure and enlightening to work with him in research and teaching. I also thank all my co-authors, Emmanuel Bresso, Clément Jonquet, Joël Legrand, Mario Lezoche, Chedy Raïssi, Patrice Ringot, Malika Smaïl-Tabbone, and Andon Tchechmedjiev. I substantially learned from discussing and working with such passionate and knowledgeable colleagues. Thanks to all members of PractiKPharma, Orpailleur, and Capsid, with whom I shared convivial moments. Thanks to my teaching colleagues from TELECOM Nancy and IDMC, especially Rémi Badonnel, Geoffray Bonin, Christophe Bouthier, Thibault Cholez, Olivier Festor, and Marc Tomczak. I would also like to thank my students and interns for their interest, curiosity, and good mood during our many classes and discussions. I cannot possibly cite all of them here. Thanks to Delphine Hubert for her support in all administrative formalities. I extend my thanks to the French National Research Agency[1] for financing this thesis through the PractiKPharma project. I thank my colleagues from Orange for their interest in my work and their support during the last months, especially Yoan Chabot.

During my educational background, I learned from passionate teachers that somehow inspired and made possible the present work. Thus, I would like to acknowledge Isabelle Chrisment, Suzanne Collin, Rana Farha, Marie-Noëlle Flavenot, Isabelle Frey, Kamel Hamdache, Jean-Marie Moureaux, Houari Mechkour, Nassima Nacer, and Emmanuel Schneider.

I am grateful to all my friends for all the support, laughs, sports sessions, New Year's Eve parties, coffees, meals, vacations, etc. I especially thank Paul, Sophie, Alessandra, Soline & Pierre-Olivier, Safae & Axel, Marion & Cédric, Imane & Martin, Vincent & Morgane, Damien & Julie & Gabriel, Sara, Clémence, Christèle & Maëlys, Anne-Claire, and Anne Julie. The time we spend together always turns into a good memory.

Last but not least, I would like to thank my family for their presence, their interest in my work, and their encouragement. In particular, I thank my brother, Nicolas, for our strong brotherly bond, Caroline, for emphasizing "sister" in "sister-in-law", and Joséphine, for her always-lightening smile. And I thank my parents for their unconditional support, love, and for all they taught me.

[1] "Agence Nationale de la Recherche" (ANR)

To my parents
To my brother
To Joséphine and Maëlys

Contents

Résumé étendu **1**

Chapter 1 Introduction **11**

1.1 From data to knowledge 11
1.2 Knowledge graphs 12
1.3 Research problems about and using knowledge graphs 14
1.3.1 A brief overview 14
1.3.2 Matching knowledge graphs 14
1.3.3 Mining knowledge graphs 16
1.4 Application context: pharmacogenomics 17
1.5 Outline of this thesis and contributions 19

Chapter 2
Ecosystem, refinement, and mining of knowledge graphs

2.1 Ecosystem of knowledge graphs 23
2.1.1 Ontology, knowledge base, knowledge-based system, and knowledge graph 23
2.1.2 The Web of data and knowledge graphs 26
2.1.3 The Open World Assumption and its consequences 29
2.1.4 Examples of ontologies and knowledge graphs 31
2.2 Refinement and mining of knowledge graphs 32
2.2.1 Refinement of knowledge graphs 32
2.2.2 Mining of knowledge graphs 37
2.3 Two frameworks to refine and mine knowledge graphs 40
2.3.1 Formal Concept Analysis and its extensions 40
2.3.2 Knowledge graph embedding 45

Chapter 3
Representing and integrating pharmacogenomic knowledge

3.1 The need for a knowledge graph focused on integration 54

3.2 PGxO: a sufficient ontology for pharmacogenomics 54
3.2.1 Specification 55
3.2.2 Conception 55
3.2.3 Diffusion 57
3.2.4 Evaluation 57
3.3 PGxLOD: a knowledge graph for pharmacogenomics 58
3.3.1 Population with preexisting LOD 58
3.3.2 Population with PharmGKB data 59
3.3.3 Population with the biomedical literature 63
3.3.4 Population with Electronic Health Records and linked biobanks studies . 67
3.3.5 Generation of mappings 68
3.4 Discussion and perspectives 69

Chapter 4
Knowledge-based matching of n-ary tuples with preorders and rules

4.1 Motivation and matching task definition 74
4.2 Ontology-based preorders 75
4.2.1 Preorder $\preccurlyeq^{\mathsf{P}}$ based on links between individuals 75
4.2.2 Preorder $\preccurlyeq^{\mathcal{O}}$ based on instantiation and subsumption 76
4.3 Using preorders to define matching rules 78
4.4 Application to pharmacogenomic knowledge 81
4.5 Discussion and perspectives 83

Chapter 5
Rediscovering alignment relations with Graph Convolutional Networks

5.1 Motivation and learning task definition 88
5.2 Matching nodes with Graph Convolutional Networks and clustering 90
5.2.1 Approach outline 90
5.2.2 Learning node embeddings with Graph Convolutional Networks and the Soft Nearest Neighbor Loss 90
5.2.3 Matching nodes by clustering their embeddings 92
5.3 Evaluating the influence of applying inference rules associated with domain knowledge 92
5.4 Experiments 94
5.4.1 Knowledge graph and gold clusters of similar nodes 94
5.4.2 Learning node embeddings 96
5.4.3 Clustering 98

5.4.4 Distance analysis 99
5.5 Discussion and perspectives 99

Chapter 6
Tackling scalability issues in mining path patterns from knowledge graphs

6.1 Motivation and mining task definition 110
6.2 Towards a scalable approach to mine interesting paths and path patterns 112
6.2.1 Canonicalizing $\mathcal{K}$ 112
6.2.2 Mining interesting neighbors and types 114
6.2.3 Mining interesting paths and path patterns 116
6.2.4 Optional and domain-dependent filtering 122
6.3 Experimental setup 122
6.4 Discussion and perspectives 125

Chapter 7
Knowledge graph refinement and mining with Concept Annotation

7.1 Basics about Concept Annotation 128
7.2 Discovering subsumption axioms between ontology classes 128
7.2.1 Classifying entities in a concept lattice with regard to their outgoing predicates 129
7.2.2 Annotating the concept lattice with classes from the ontology $\mathcal{O}$ 130
7.2.3 Comparing discovered axioms with existing axioms in the ontology $\mathcal{O}$ . . 132
7.2.4 Preliminary experiment with DBpedia 133
7.3 Suggesting alignments between classes of ontologies 134
7.3.1 Preliminary: a pattern structure for an ontology $\mathcal{O}_i$ 134
7.3.2 Classifying entities with regard to $\mathcal{O}_{\texttt{ref}}$ in a concept lattice 135
7.3.3 Annotating the concept lattice with classes from $\mathcal{O}_1$ and $\mathcal{O}_2$ 136
7.3.4 Reading alignments from the annotated lattice 136
7.4 Discovering domain $\rightarrow$ range associations for predicates 137
7.4.1 Classifying predicate instantiations in a concept lattice 139
7.4.2 Annotating the concept lattice with $\mathcal{O}_1$ and $\mathcal{O}_2$ 139
7.4.3 Reading domain $\rightarrow$ range associations from the annotated lattice 140
7.5 Discussion and perspectives 141

Chapter 8 Conclusion and perspectives **143**
8.1 Summary of contributions 143
8.2 Perspectives 144

Appendix A List of publications **147**

References **149**

Résumé étendu

Des données aux connaissances

Dans cette thèse, nous nous intéressons aux *graphes de connaissances* et aux deux tâches d'*appariement* et de *fouille*. Avant d'introduire le concept de graphe de connaissances et les tâches étudiées, nous proposons de préciser la notion de *connaissances*. Bien qu'il soit fréquent d'utiliser de manière interchangeable les trois mots *données*, *informations* et *connaissances*, Schreiber et al. [147] proposent la distinction suivante, dans laquelle chaque notion est construite en se basant sur les précédentes :

Les données sont des symboles non-interprétés, tels que des chaînes de caractères, des entiers, etc.

Les informations sont des données associées à une signification ou une description qui permet leur interprétation, par exemple par un humain. Ainsi, un entier peut être associé à des métadonnées qui indiquent à un humain que ce nombre représente l'âge d'un utilisateur.

Les connaissances sont définies comme des informations et données assimilées qui peuvent être utilisées pour accomplir des tâches et créer davantage d'informations. Ainsi, les connaissances sont *actionnables* car elles peuvent aider à la réalisation d'une tâche ou la prise d'une décision. Les connaissances ont également une *capacité générative* puisqu'elles peuvent être utilisées pour créer davantage d'informations. Schreiber et al. [147] soulignent que cette génération de nouvelles informations est une des fonctions majeures des connaissances. Par exemple, les connaissances peuvent prendre la forme de règles qui indiquent quelles actions accomplir lorsque des conditions sont validées.

Plusieurs langages de représentation des connaissances ont été développés, comme les logiques de description [9]. Ces langages sont associés à une *sémantique formelle* qui permet aux connaissances d'être représentées de manière non-ambiguë tout en contraignant leur interprétation à la fois par les humains et les agents logiciels. Ces langages basés sur la logique permettent de *raisonner* à propos des connaissances représentées en utilisant la déduction logique. Ainsi, étant donné des connaissances représentées dans de tels langages basés sur la logique, un système d'informatique décisionnelle peut utiliser des mécanismes de raisonnement pour conclure sur le choix d'une action.

Cependant, la définition des connaissances est parfois assouplie. Par exemple, Ehrlinger and Wöß [57] considèrent que les connaissances peuvent prendre la forme de règles, de définitions, mais aussi de faits et de déclarations. Cet assouplissement est similaire à la définition des connaissances utilisée par Hogan et al. [79] dans leur tutoriel à propos des graphes de connaissances :

Définition (Connaissances, selon la définition de Hogan et al. [79])**.** *Les connaissances* sont des éléments *connus*.

Par exemple, les relations d'amitié ou les effets secondaires causés par des médicaments peuvent être vus comme des faits connus par l'expert qui les représente. Ainsi, ils constituent des connaissances selon la définition précédente. La similarité avec la définition de Ehrlinger and Wöß [57] vient de l'idée que les faits et les déclarations peuvent être vus comme des unités qui sont connues. Dans ce manuscrit, nous adoptons cette définition de Hogan et al. [79] car elle est suffisamment flexible à la fois par rapport à la définition des graphes de connaissances adoptée et pour notre contexte applicatif (présentés ci-dessous).

Les graphes de connaissances

Les connaissances peuvent être représentées de différentes manières, parmi lesquelles les graphes qui permettent une représentation flexible associée à des algorithmes puissants (*e.g.*, de parcours). En effet, de nombreux faits et déclarations peuvent être vus de façon relationnelle, dans laquelle les entités impliquées sont décrites par les relations qui les lient. Par exemple, un fait pourrait être "Alice est amie avec Bob". Ce fait peut être représenté d'une façon *tabulaire*, utilisée dans les bases de données relationnelles, comme illustré sur la Figure 1.1a page 12. Alternativement, ce fait peut être vu comme un triplet $\langle$`Alice`, `est-amie-avec`, `Bob`$\rangle$. Ce triplet peut quant-à-lui être représenté par un graphe dont les nœuds représentent les entités impliquées (*i.e.*, `Alice` et `Bob`) et dont l'arête représente la relation qui les lie (*i.e.*, leur amitié). Un tel graphe est illustré sur la Figure 1.1b page 12.

Tout comme la définition de *connaissances*, plusieurs définitions de *graphes de connaissances* co-existent. Des auteurs suggèrent que les graphes de connaissances acquièrent et intègrent de l'information dans le but de produire de nouvelles connaissances [57]. Cet objectif correspond à la capacité générative des connaissances soulignée par Schreiber et al. [147]. D'autres auteurs proposent ou discutent des définitions plus flexibles. Par exemple, Paulheim [132] liste plusieurs critères pour définir un graphe de connaissances mais indique que n'importe quel graphe représentant des connaissances pourrait être un graphe de connaissances en adoptant une perspective large. Dans ce manuscrit, nous choisissons d'adopter la définition proposée par Hogan et al. [79] :

Définition (Graphe de connaissances, selon la définition de Hogan et al. [79])**.** Un *graphe de connaissances* est un graphe de données destiné à accumuler et transmettre des connaissances du monde réel, dont les nœuds représentent des entités d'intérêt et dont les arêtes représentent des relations entre ces entités.

Il est intéressant de noter l'absence de contraintes sur les types de graphes éligibles. Par exemple, le graphe de la Figure 1.1b page 12 est un *multigraphe orienté et étiqueté* : les arcs sont orientés, les nœuds et les arcs sont étiquetés, et des relations différentes peuvent exister entre deux nœuds. Cependant, il est aussi possible d'utiliser d'autres formalismes, comme les *graphes avec propriétés* ("*property graphs*") [107] dans lesquels des paires (propriété, valeur) sont associées aux nœuds et aux arcs, ou encore les *hypergraphes* dont les arêtes connectent des ensembles de nœuds au lieu de paires de nœuds.

Cette définition est donc plus générale et applicable en particulier à des "jeux de données" représentés avec les standards et les paradigmes développés dans le cadre de la vision du Web Sémantique [15]. En effet, ces jeux de données prennent la forme de données ouvertes et liées ("*Linked Open Data*" ou LOD) [17] et reposent sur des standards et des technologies comme les "*Uniform Resource Identifiers*" (URIs) et le langage "*Resource Format Description Language*" (RDF). Les URIs sont des identifiants uniques pour les entités d'un monde représenté (*e.g.*, une

personne, un médicament), des classes d'entités (*e.g.*, la classe de tous les médicaments qui sont des analgésiques), ou des relations (*e.g.*, `est-amie-avec`). RDF est un langage permettant de représenter des données et des connaissances. Les éléments atomiques du langage RDF sont des déclarations sous la forme de triplets $\langle \texttt{sujet}, \texttt{prédicat}, \texttt{objet} \rangle$, indiquant qu'une relation spécifique identifiée par le `prédicat` existe entre le `sujet` et l'`objet`. Le `sujet` est une URI, le `prédicat` est une URI, et l'`objet` est soit une URI soit un litéral (*e.g.*, une chaîne de caractères, un entier). Comme évoqué précédemment, ces triplets peuvent être représentés sous la forme d'un graphe. En plus des déclarations RDF basiques, il est possible d'étendre un graphe RDF avec des déclarations RDFS et OWL qui sont généralement utilisées pour définir des ontologies, *i.e.*, des représentations formelles d'un domaine [74]. Les ontologies sont composées de classes et de prédicats. Une entité peut instancier une ou plusieurs classes et ces classes peuvent alors être vues comme les "types" de l'entité. Par exemple, `Alice` pourrait instancier la classe `Personne`. Les classes et les prédicats sont chacun organisés dans une hiérarchie définie par la relation de subsomption. Cette relation est habituellement représentée par le symbole $\sqsubseteq$ et indique qu'une classe (respectivement un prédicat) est plus spécifique qu'une autre, *i.e.*, toutes ses instances sont aussi des instances de l'autre classe. RDFS et OWL sont deux langages qui sont associés à des règles d'inférence[2], équipant les graphes de connaissances représentés à l'aide de standards du Web Sémantique avec des mécanismes de raisonnement. Ainsi, ces graphes de connaissances ont une capacité inhérente à générer davantage de connaissances.

Les jeux de données LOD constituent le "Web des données" qui matérialise les idées du Web Sémantique. Ces idées ont été proposées dans un article de Berners-Lee et al. [15] en 2001. A cette époque, le Web pouvait être vu comme un "Web des documents" conçu pour être lu par des humains. Ainsi, le Web Sémantique a été imaginé comme une extension permettant aux agents logiciels de comprendre et interagir avec le Web pour mener à bien des tâches complexes. Dans ce but, les principes des données ouvertes et liées ("*Linked Open Data principles*") [17] ont été établis et offrent un ensemble de bonnes pratiques pour publier et connecter des données sur le Web. La large adoption de ces principes conduit à un nombre grandissant de jeux de données LOD accessibles au sein du Web des données, comme représenté sur la Figure 1.2 page 14. De par la nature décentralisée du Web, ces graphes de connaissances sont publiés, édités, consultés, et interprétés de manière concurrente par des agents humains et logiciels. Par conséquent, le Web des données présente une hétérogénéité inhérente : des graphes de connaissances peuvent avoir différentes caractéristiques (*e.g.*, taille, granularité, ontologies associées, etc.) et peuvent capturer des connaissances de domaines différents. Par exemple, DBpedia [99] est un graphe de connaissances générique construit à partir du contenu semi-structuré de Wikipédia. D'autres graphes de connaissances ciblent les sciences de la vie, comme les graphes issus du projet Bio2RDF [54] qui sont le résultat d'une transformation de plusieurs sources dans d'autres formats (*e.g.*, des bases de données relationnelles).

Questions de recherche associées aux graphes de connaissances

Une brève vue d'ensemble

Le développement du Web des données et le nombre croissant de graphes de connaissances en son sein sont associés à plusieurs questions de recherche impliquant différents champs de recherche comme la recherche d'information, le traitement du langage naturel, les bases de don-

[2] Le langage OWL permet d'encoder certaines logiques de description [9] que nous avons mentionnées précédemment.

nées, l'apprentissage automatique, et la représentation de connaissances et le raisonnement [22]. Par exemple, la *construction* des graphes de connaissances implique la *représentation* des connaissances d'un domaine. Des faits, des entités, et des relations peuvent potentiellement être *automatiquement extraites* de sources existantes. Ces sources peuvent prendre la forme de données semi-structurées (*e.g.* l'arbre DOM de pages HTML), ou non-structurées (*e.g.*, du texte brut). Cependant, les graphes de connaissances construits par extraction automatique ou production participative ("*crowdsourcing*") peuvent ne pas atteindre une couverture complète (*i.e.*, contenir toutes les informations concernant toutes les entités du monde) et peuvent ne pas être entièrement corrects [132, 176]. De ce fait, des méthodes doivent être établies pour *raffiner* ces graphes de connaissances, par exemple pour les compléter ou détecter des déclarations erronées. Les graphes de connaissances peuvent également servir de support pour des tâches associées aux moteurs de recherche telles que l'interprétation de requêtes, la réponse aux questions, et l'enrichissement des résultats de recherche (voir Figure 1.3 page 15). Dans cette thèse, nous nous concentrons sur les deux problèmes spécifiques d'*appariement* et de *fouille* des graphes de connaissances que nous décrivons plus précisément dans les paragraphes suivants.

L'appariement dans les graphes de connaissances

La nature décentralisée du Web des données entraîne la publication et l'édition concurrentes des graphes de connaissances. Par conséquent, des chevauchements existent, *i.e.*, des graphes de connaissances peuvent décrire des unités similaires (*e.g.*, des entités, des faits, etc.) tout en ayant des différences en termes de qualité, complétude, granularité et vocabulaires. Ces chevauchements peuvent aussi apparaître au sein d'un même graphe de connaissances car le processus de construction implique souvent l'intégration et l'agrégation de différentes sources. Les unités chevauchantes peuvent être exactement identiques, faiblement apparentées, complémentaires, ou une unité peut être plus spécifique qu'une autre. Par exemple, un graphe de connaissances peut indiquer que "Alice est la mère de Carl", ce qui est plus spécifique que la déclaration "Alice est membre de la même famille que Carl" contenue dans un autre graphe de connaissances.

En conséquence, une tâche majeure consiste à *apparier* les graphes de connaissances, *i.e.*, identifier des unités identiques, plus spécifiques, ou similaires entre et au sein de graphes de connaissances. Le processus d'appariement produit des *alignements* qui sont des correspondances entre unités. Les unités appariées peuvent être des entités, des classes d'ontologie, des prédicats, des faits, etc. Ce processus est difficile car il faut faire face à l'hétérogénéité inhérente aux différents graphes de connaissances en termes de vocabulaires, langues, granularités, etc. Néanmoins, pour surmonter les problèmes d'hétérogénéité, les approches d'appariement peuvent utiliser différentes caractéristiques des graphes de connaissances comme les mécanismes de raisonnement et les connaissances formalisées au sein d'ontologies. La tâche d'appariement est au centre du domaine de recherche appelé appariement d'ontologies ("*ontology matching*") [58]. Elle est aussi liée à la réconciliation dans les bases de données [1] dont l'objectif est de gérer des répliques co-existantes et indépendamment modifiées d'une même base de données. Nous détaillons les approches existantes en Sous-section 2.2.1.

De nombreuses applications en aval bénéficient des alignements issus du processus d'appariement, ce qui illustre l'importance de cette tâche. Par exemple, les approches de vérification d'information ("*fact-checking*") peuvent souligner des accords ou des contradictions entre différentes sources à l'aide de ces alignements. Les alignements connectent différents graphes de connaissances et offrent donc une vue consolidée et unifiée des connaissances que ces graphes représentent et fournissent conjointement. Les graphes de connaissances peuvent être complétés avec des faits contenus dans d'autres graphes, fusionnés (*i.e.*, créant un nouveau graphe de

connaissances), ou réconciliés (*i.e.*, leurs contenus respectifs sont harmonisés) [58]. Nous introduirons dans la sous-section suivante le processus de fouille des graphes de connaissances qui permet de découvrir de nouvelles connaissances. En alignant les graphes de connaissances en amont, puis en fouillant la vue consolidée qui en résulte, il est possible de s'attendre à un plus grand nombre de résultats ou une amélioration de leur qualité. Des versions différentes du même graphe de connaissances peuvent aussi être alignées pour évaluer, *e.g.*, la dérive conceptuelle ("*concept drift*") entre les versions.

La fouille de graphes de connaissances

Précédemment, nous avons indiqué que les notions de données, d'informations et de connaissances peuvent être vues comme construites de manière pyramidale. En accord, d'une certaine manière, avec la définition de connaissances comme des informations et des données assimilées, il est possible de découvrir de nouvelles connaissances à partir de données. Ce processus est appelé Extraction de Connaissances à partir de Bases de Données (ECBD) et a été défini par Frawley et al. [62] comme "*l'extraction non-triviale d'informations implicites, précédemment inconnues et potentiellement utiles à partir de données*". Fayyad et al. [59] a modélisé ce processus par les étapes successives représentées sur la Figure 1.4 page 16. Tout d'abord, une partie des données disponibles est *sélectionnée* pour le processus d'ECBD, en fonction des objectifs et des connaissances du domaine d'un analyste. Cette sélection peut réduire le nombre d'échantillons ou les attributs considérés. Les données sélectionnées sont ensuite *préparées*, par exemple pour gérer les valeurs manquantes ou supprimer les données bruitées, avant d'être *transformées* dans un format accepté par l'*algorithme de fouille*. Le choix de l'algorithme de fouille dépend de la tâche, *e.g.*, régression, classement, ou classification. Les motifs produits par l'algorithme de fouille sont *interprétés* et *évalués* par l'expert pour établir, *e.g.*, leur validité ou leur utilité. Il est intéressant de noter que le terme de *fouille* est associé à une étape spécifique du processus d'ECBD mais est parfois utilisé pour évoquer le processus dans son ensemble. Nous adoptons une telle simplification dans cette thèse.

Le processus d'ECBD est itératif, interactif et dépend des connaissances que l'*analyste* possède à propos du domaine considéré. Nous avons vu que les connaissances peuvent être formalisées pour être utilisées et interprétées par des agents logiciels. De ce fait, des travaux de recherche proposent d'intégrer ces connaissances formalisées dans le processus d'ECBD, aboutissant à l'Extraction de Connaissances guidée par les Connaissances du Domaine (ECCD) [100, 143]. Dans ce cadre, les graphes de connaissances peuvent alors être impliqués dans toutes les étapes du processus d'ECBD (voir Sous-section 2.2.2) [143]. En particulier, il est possible de considérer les graphes de connaissances comme l'entrée du processus d'ECCD pour découvrir des connaissances à partir de connaissances, ou des méta-connaissances à propos des connaissances. Nous rappelons que, suivant la définition adoptée de connaissances, les faits et les déclarations représentés dans les graphes de connaissances sont eux-mêmes considérés comme des connaissances. Cependant, la fouille de graphes de connaissances soulève plusieurs questions de recherche, telles que :

- Comment faire face aux problèmes de scalabilité associés à la fouille de graphes de connaissances dont la taille s'accroît constamment ?
- Quelles caractéristiques extraites des graphes de connaissances peuvent être utilisées en tant qu'attributs dans les algorithmes de fouille ? [134, 140]
- Comment s'assurer que les algorithmes de fouille considèrent la sémantique formelle et les connaissances du domaine associées aux graphes de connaissances ?

- Étant donné que les graphes de connaissances sont conçus pour être utilisés à la fois par des humains et des agents logiciels, peuvent-ils participer à l'élaboration d'attributs interprétables, et donc jouer un rôle dans l'intelligence artificielle explicable ?

Contexte applicatif de la thèse : la pharmacogénomique

Le travail présenté dans cette thèse a été mené dans le cadre du projet *PractiKPharma*[3] [38] (2016–2020), financé par l'Agence Nationale de la Recherche. Ce projet a pour but d'étudier des approches informatiques pour extraire et comparer des connaissances dans le domaine biomédical de la pharmacogénomique ("*pharmacogenomics*" en anglais, classiquement abrégé PGx).

La pharmacogénomique étudie l'influence des facteurs génétiques dans les réponses aux médicaments. Les connaissances de ce domaine sont particulièrement utiles en médecine de précision dont l'objectif est d'adapter les traitements aux patients pour réduire le risque d'effets secondaires et maximiser l'efficacité des traitements [30]. Une unité de connaissances en PGx prend la forme d'une relation n-aire, liant un ensemble de médicaments, un ensemble de facteurs génétiques, et un ensemble de phénotypes (voir Figure 1.5 page 18). Une telle relation décrit qu'un patient traité avec les médicaments spécifiés tout en ayant les facteurs génétiques spécifiés aura une plus grande probabilité de subir les phénotypes indiqués, *e.g.*, des effets secondaires. Nous illustrons cette notion sur la Figure 1.6 page 18. Trois relations PGx sont représentées et indiquent que des patients avec des variants différents du gène CYP2D6 pourront subir des réactions différentes à un même traitement à la codéine. Les relations PGx peuvent être vues comme des déclarations, et donc constituent des connaissances d'après la définition précédemment adoptée. Les connaissances PGx ont pour origine des sources distinctes :

Les connaissances de l'état l'art peuvent être trouvées dans des bases de données spécialisées ou des articles de la littérature biomédicale. Parmi ces bases de données, il convient de mentionner PharmGKB [173] qui est la base de données de référence en PGx. Elle est manuellement complétée par des biologistes qui ajoutent des unités de connaissances dans PharmGKB après une revue de la littérature.

Les connaissances observationnelles proviennent de la fouille de données observationnelles comme les dossiers patients électroniques des hôpitaux (DPE).

Les connaissances de l'état de l'art peuvent ne pas être validées ou partiellement validées [81], par exemple à cause de cohortes de patients trop réduites. De telles connaissances doivent être davantage étudiées avant d'être intégrées dans la pratique clinique. Par exemple, la Figure 1.7 page 18 montre que seulement 10% des relations PGx contenues dans PharmGKB ont un niveau d'évidence de "1" ou "2", ce qui correspond à des relations intégrées dans des recommandations cliniques, ou des études montrant à minima un niveau modéré d'association. En revanche, les 90% restants ont un niveau d'évidence de "3" ou "4", ce qui correspond à des études non-répliquées, plusieurs études avec un manque de preuves, ou des études non-significatives.

Le projet PractiKPharma formule l'hypothèse que la PGx peut bénéficier des approches d'appariement développées en informatique. En effet, apparier les sources de connaissances en PGx pour les réconcilier pourrait fournir une vue consolidée des connaissances disponibles dans ce domaine, permettant potentiellement de valider des connaissances de l'état de l'art avec des dossiers patients électroniques. Dans l'objectif d'une telle approche d'appariement, le projet PractiKPharma propose de se baser sur les graphes de connaissances et les ontologies

[3] `http://practikpharma.loria.fr`

existantes et ciblant les sciences de la vie. Ceux-ci constituent une majeure partie des données ouvertes et liées disponibles [143] (en rouge sur la Figure 1.2 page 14). Ces travaux préexistants facilitent la représentation des relations PGx car ils définissent déjà des connaissances à propos des composants de ces relations, *i.e.*, les médicaments, les facteurs génétiques et les phénotypes. Le projet PractiKPharma propose donc de tirer parti des technologies et travaux de recherche autour du Web Sémantique et de l'Intelligence Artificielle pour la réconciliation des sources de connaissances en PGx.

Réciproquement, il est intéressant de noter que cette application réelle profite aux approches informatiques en soulevant des questions et problèmes importants et en motivant le développement de nouvelles approches qui les abordent. En effet, les relations PGx illustrent le besoin de développer des approches d'appariement qui ciblent les relations n-aires. Du fait de la variété des sources, un autre défi du processus d'appariement consiste à être capable de dépasser leur hétérogénéité inhérente en termes de langues, vocabulaires, granularités, et complétude (*i.e.*, composants inconnus ou manquants). La Figure 1.8 page 19 illustre ce défi avec quelques exemples de problèmes d'hétérogénéité dans la représentation de relations PGx ainsi que les résultats que nous souhaiterions obtenir d'un processus d'appariement. Cette application réelle et ses défis inhérents ont motivé les contributions détaillées dans ce manuscrit.

Résumé des chapitres et des contributions

Le présent manuscrit est organisé comme suit.

Au sein du **Chapitre 2**, nous faisons une revue de l'écosystème des graphes de connaissances, de certaines tâches qui leur sont associées, et des approches existantes pour les mener à bien. En particulier, nous discutons certaines définitions existantes des *graphes de connaissances* et des concepts associés, et nous identifions les définitions qui seront utilisées dans les chapitres suivants. Nous introduisons ensuite la perspective du *raffinement* des graphes de connaissances avant de nous concentrer sur la tâche d'*appariement* entre et au sein de graphes de connaissances. Nous détaillons également la tâche de *fouille* des graphes de connaissances pour découvrir de nouvelles connaissances. Nous donnons un aperçu de quelques travaux existants traitant de ces deux tâches, ce qui nous permet d'illustrer la variété des approches associées au raffinement et à la fouille des graphes de connaissances.

Le **Chapitre 3** détaille nos travaux sur la représentation des connaissances en PGx. Dans ce chapitre, nous commençons par motiver le besoin d'un nouveau graphe de connaissances ciblant l'intégration de ces connaissances. Nous utilisons les technologies du Web Sémantique pour représenter, indiquer la provenance et intégrer les connaissances en PGx de trois sources : la base de données de référence PharmGKB, la littérature biomédicale et les dossiers patients électroniques. Dans ce but, nous proposons PGxO, une ontologie réduite pour représenter les unités de connaissances en PGx. Nous instancions PGxO avec des connaissances provenant des trois sources susmentionnées, ce qui aboutit au graphe de connaissances que nous avons nommé PGxLOD. Du fait de sa taille et des particularités associées aux connaissances en PGx (*e.g.*, arité, hétérogénéité), PGxLOD nous offre un cadre expérimental intéressant et unifié pour tester les approches présentées dans les autres chapitres. Ce chapitre est basé sur deux contributions : un article d'atelier publié dans *NETTAB 2017* qui introduit une version préliminaire de PGxO [114], et un article de journal publié dans *BMC Bioinformatics* (2019) qui détaille PGxO and PGxLOD [115].

Dans le **Chapitre 4**, nous présentons une approche basée sur les connaissances pour apparier des relations n-aires représentées au sein d'un graphe de connaissances. Nous propose un cadre

général qui modélise ces relations comme des tuples n-aires dont les arguments sont des ensembles d'individus. Nous accomplissons l'appariement entre les tuples grâce à la définition de cinq règles qui sont mathématiquement bien fondées. Ces règles comparent les arguments de deux tuples et concluent sur leur niveau d'association choisi dans une échelle de cinq niveaux, tels que "équivalents", "plus spécifiques", ou "faiblement similaires". La principale originalité de ces règles vient de l'utilisation de préordres issus des connaissances du domaine pour parer aux problèmes d'hétérogénéité lors de la comparaison des tuples. Ce chapitre est basé sur un article de conférence publié dans *ICCS 2020* [113].

Les résultats de cette approche à base de règles ont souligné le besoin d'une comparaison flexible pour surmonter les problèmes d'hétérogénéité. Aussi, le **Chapitre 5** explore l'utilisation de plongements de graphe ("*graph embedding*"), et en particulier des réseaux convolutifs pour les graphes ("*Graph Convolutional Networks*", ou GCNs) [92, 146], pour apparier des entités dans les graphes de connaissances. Les GCNs permettent d'apprendre une représentation vectorielle pour chaque entité [27] à partir de son voisinage dans un graphe de connaissances. Dans notre cas d'usage, notre objectif est d'apprendre des vecteurs dont la distance reflète la similarité entre les entités. Cette méthode est davantage flexible qu'une approche à base de règles du fait de la représentation continue des vecteurs. De plus, les conditions de similarité sont apprises à partir d'exemples et non spécifiées par des règles. Ainsi, les vecteurs appris peuvent être plus résilients à des représentations hétérogènes et peuvent amener à davantage de résultats. Dans le cadre de cette approche, nous examinons l'interaction entre les connaissances du domaine et les modèles à base de GCNs avec les deux points d'attention suivants. Premièrement, nous mesurons l'amélioration des résultats d'appariement lors de la considération de différentes règles d'inférence associées aux connaissances du domaine, indépendamment ou combinées. Deuxièmement, alors que notre modèle à base de GCNs ignore les différentes relations d'alignement utilisées (*e.g.* équivalence, similarité faible), nous observons des distributions de distances différentes dans l'espace vectoriel pour chacune de ces relations, ce qui correspond d'une certaine manière à leur redécouverte par le modèle. Ce chapitre est une extension de résultats préliminaires décrits dans un article d'atelier publié dans *DL4KG 2019* [120].

Dans le **Chapitre 6**, nous considérons un ensemble spécifié de nœuds d'intérêt, appelé les *nœuds grains*, et nous extrayons des caractéristiques associées à ceux-ci, comme leurs voisins ou les chemins dont ils sont les racines dans un graphe de connaissances. Les entités dans de tels chemins peuvent instancier des classes d'ontologie. De ce fait, nous remplaçons ces entités par les classes qu'elles instancient pour générer des motifs de chemins qui pourront caractériser davantage de nœuds grains que les chemins. Dans le cadre de notre exemple, le chemin $\xrightarrow{\texttt{est-ami-avec}}$ `Bob` pourrait seulement caractériser `Alice`, alors que le motif de chemin $\xrightarrow{\texttt{est-ami-avec}}$ `Personne` pourrait être associé avec davantage d'entités. La fouille de ces motifs soulève naturellement des problèmes de scalabilité du fait de la nature combinatoire des graphes de connaissances. Nous répondons à ces problèmes en proposant une approche de fouille de motifs qui se base sur un ensemble de contraintes (*e.g.*, seuil de support, de degré), un élagage des motifs redondants basé sur la hiérarchie des classes d'ontologie, et la *monotonicité* du support des chemins et des motifs de chemins. Ce chapitre est basé sur un papier de conférence publié dans *ALGOS 2020* [112].

Dans le **Chapitre 7**, nous proposons de raffiner et fouiller des graphes de connaissances avec l'Annotation de Concepts, une extension de l'Analyse Formelle de Concepts (AFC) que nous proposons. L'AFC [70] est un cadre mathématique qui groupe des objets au sein de concepts formels en fonction de leurs attributs communs. Ces concepts sont organisés au sein d'une structure hiérarchique appelée treillis de concepts. Le treillis permet d'identifier les objets équivalents

(au sein d'un même concept) ou plus précis (dans les concepts subsumés). Nous utilisons cette organisation hiérarchique pour *(i)* organiser hiérarchiquement des classes d'une ontologie, *(ii)* suggérer des alignements entre les classes de différentes ontologies, et *(iii)* découvrir des associations domaine $\rightarrow$ image fréquentes pour les prédicats, *i.e.*, des associations $C_1 \rightarrow C_2$ où C_1 et C_2 sont deux classes d'ontologie apparaissant fréquemment en tant que domaine et image dans les assertions d'un prédicat. Ce chapitre est basé sur un article de conférence publié dans *ISMIS 2017* [116], un poster publié dans *BDA 2017* [117], et un article d'atelier publié dans *FCA4AI 2018* [118].

Enfin, le **Chapitre 8** conclut ce manuscrit en proposant un résumé de nos contributions et en esquissant des perspectives de recherche. La liste de toutes nos publications est disponible en **Annexe A**.

Chapter 1

Introduction

Contents

1.1 From data to knowledge 11
1.2 Knowledge graphs 12
1.3 Research problems about and using knowledge graphs 14
1.3.1 A brief overview 14
1.3.2 Matching knowledge graphs 14
1.3.3 Mining knowledge graphs 16
1.4 Application context: pharmacogenomics 17
1.5 Outline of this thesis and contributions 19

1.1 From data to knowledge

In this thesis, we are interested in *knowledge graphs* and the two tasks of *matching* and *mining* them. Before introducing the concept of knowledge graph and the studied tasks, we need to define the notion of *knowledge*. Although it is frequent to interchangeably use the three words *data*, *information*, and *knowledge*, Schreiber et al. [147] propose the following distinction in which each notion builds on top of the previous ones:

Data are uninterpreted symbols, such as strings of characters, integers, etc.

Information consists in data equipped with a meaning allowing its interpretation, for example by a human being. To illustrate, an integer can be equipped with metadata that indicates to a human being that the number represents the age of a user.

Knowledge is defined as assimilated information and data that can be used to carry out tasks and create more information. Hence, knowledge is *actionable* since it can support the achievement of a task or the making of a decision. Knowledge has also a *generative capability* since it can be used to create more information. Schreiber et al. [147] underline that this generation of new information is one of the major functions of knowledge. For example, knowledge can consist of rules that can indicate the action to achieve when some conditions are valid.

Various knowledge representation languages have been developed, such as Description Logics [9]. Such languages are associated with a *formal semantics* that ensures knowledge is unambiguously



(a) Tabular representation

(b) Graph representation

Figure 1.1: Two possibles representations of relational data illustrated with the fact "Alice is a friend of Bob". In the tabular representation, the table corresponds to the relation `is-a-friend-of` and each line corresponds to an instantiation or an assertion of this relation between two specific entities (here, `Alice` and `Bob`). Alternatively, in the graph representation, each assertion is represented as an edge labeled by the relation between two nodes representing the entities.

represented and constraints its interpretation by both human and software agents. Such logic-based representation languages enables to *reason* about the represented knowledge, using logical deduction. Hence, for example, given some knowledge represented in such logic-based languages, a decision support system can use reasoning mechanisms to conclude on the choice of an action.

However, the definition of knowledge is sometimes relaxed. For example, Ehrlinger and Wöß [57] consider that knowledge can be formed by rules, definitions, but also facts and statements. This relaxation is similar to the definition of knowledge used by Hogan et al. [79] in their tutorial about knowledge graphs:

Definition 1 (Knowledge as defined by Hogan et al. [79])**.** *Knowledge* is something that is *known*.

For example, friendship relations or side effects caused by drugs can be seen as facts known to the expert that represents them, and thus constitute knowledge in view of Definition 1. The similarity with the definition of Ehrlinger and Wöß [57] comes from the idea that facts and statements can be seen as units that are known. In the following, we adopt this definition of Hogan et al. [79] because it is flexible enough for the adopted definition of knowledge graph (see Section 1.2) and for our application context (see Section 1.4).

1.2 Knowledge graphs

Knowledge can be represented in several forms, among which graphs that provide a flexible representation associated with powerful algorithms (*e.g.*, traversal). Indeed, many facts and statements can be seen from a relational perspective in which involved entities are described by the relations existing between them. For example, a fact could be "Alice is a friend of Bob". Such a fact can be represented in a *tabular* form used in relational databases, as illustrated in Figure 1.1a. Alternatively, this fact can be seen as a triple $\langle$`Alice`, `is-a-friend-of`, `Bob`$\rangle$. This triple can, in turn, be represented by a graph whose nodes depict the involved entities (*i.e.*, `Alice` and `Bob`) and whose edge depicts the relation holding between them (*i.e.*, their friendship). Such a graph is illustrated in Figure 1.1b.

Similarly to the definition of *knowledge*, several definition of *knowledge graphs* co-exist. Some authors require knowledge graphs to acquire and integrate information in the aim of deriving new knowledge [57]. This purpose corresponds to the generative capability of knowledge as defined by Schreiber et al. [147]. Other authors provide or discuss more flexible definitions. For example, Paulheim [132] lists several criteria to define a knowledge graph but indicates that from

a broader perspective, any graph representing some knowledge could be a knowledge graph. In this manuscript, we decide to use the definition proposed by Hogan et al. [79]:

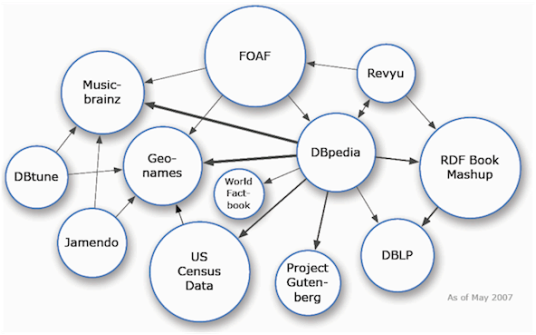
Definition 2 (Knowledge graph as defined by Hogan et al. [79])**.** A *knowledge graph* is a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities.

Interestingly, there is no constraint on the eligible types of graph. For example, the graph in Figure 1.1b is a *directed labeled multigraph*: arcs are directed, nodes and arcs are labeled, and different relations can hold between two nodes. However, it is also possible to use other formalisms, such as *property graphs* [107] in which pairs (property, value) are associated with nodes and arcs, or *hypergraphs* whose edges connect sets of nodes instead of pairs.

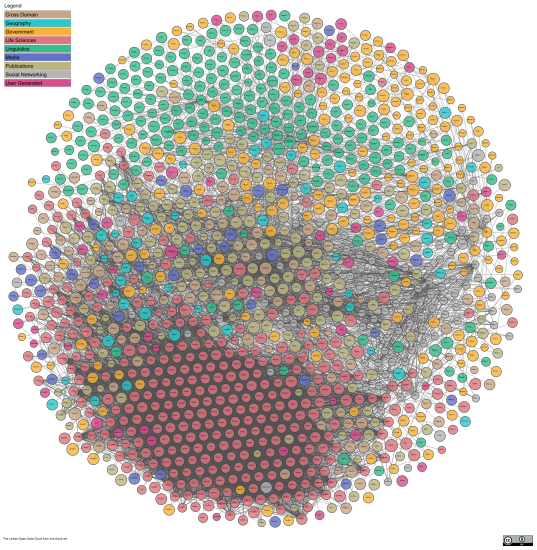
In particular, this definition is more general and thus applicable to "data sets" represented using standards and paradigms developed within the Semantic Web vision [15]. Indeed, these data sets have the form of Linked Open Data (LOD) [17] and rely on standards and technologies such as Uniform Resource Identifiers (URIs) and the Resource Format Description language (RDF). URIs provide unique identifiers for entities of a world (*e.g.*, a person or a drug), classes of entities (*e.g.*, the class of all drugs that are analgesics), or relations (*e.g.*, is-a-friend-of). RDF is a language for representing data and knowledge. Atomic RDF statements are triples $\langle$`subject`, `predicate`, `object`$\rangle$, indicating that a specific relation identified by the `predicate` holds between the `subject` and the `object`. The `subject` is a URI, the `predicate` is a URI, and the `object` is either a URI or a literal (*e.g.*, a string or an integer). As aforementioned, such triples can be represented in the form of a graph. Besides basic RDF statements, it is possible to extend an RDF graph with RDFS and OWL statements usually used to define ontologies, *i.e.*, formal representation of a domain [74]. Ontologies consist of classes and predicates. An entity can instantiate one or several classes and such classes can then be seen as the "types" of the entity. For example, `Alice` could instantiate a class `Person`. Classes and predicates are organized in two respective hierarchies by the subsumption relation. This relation is usually denoted by $\sqsubseteq$ and states that a class (respectively a predicate) is more specific than another, *i.e.* that all its instances are also instances of the other class. Both the RDFS and OWL languages provide inferences rules[4], thus equipping knowledge graphs expressed in Semantic Web standards with reasoning mechanisms. Hence, such knowledge graphs have an inherent capability to generate more knowledge.

LOD sets compose the "Web of data" that materializes ideas of the Semantic Web. This idea was envisioned in an article by Berners-Lee et al. [15] in 2001. At the time, the Web was a "Web of documents" designed for humans to read. Hence, the Semantic Web was conceived as an extension to enable software agents to comprehend and interact with the Web to carry out complex tasks. To this aim, the Linked Open Data principles [17] were established and offer some best practices for publishing and connecting data on the Web. The wide adoption of these principles leads to an ever-growing number of LOD sets being accessible in the Web of data, as depicted in Figure 1.2. Due to the decentralized nature of the Web, these knowledge graphs are concurrently published, edited, accessed, and interpreted by human and software agents. Consequently, there is an inherent heterogeneity in the Web of data: knowledge graphs may have different characteristics (*e.g.*, size, granularity, linked ontologies, etc.) and capture knowledge from different domains. For example, DBpedia [99] is a generic knowledge graph built from the semi-structured content of Wikipedia. Other knowledge graphs focus on Life Sciences, such as the ones of the Bio2RDF project [54] that result from the transformation of various sources in other formats (*e.g.*, relational databases).

[4]The OWL language enables to encode Description Logics [9] that we previously mentioned.



(a) In May, 2007 (12 knowledge graphs)



(b) In May, 2020 (1,255 knowledge graphs)

Figure 1.2: The Linked Open Data cloud from `https://lod-cloud.net` – CC BY. Nodes represent data sets and edges represent existing interconnections between them. Red nodes are data sets related to Life Sciences. These clouds can be zoomed and browsed on the LOD cloud website.

1.3 Research problems about and using knowledge graphs

1.3.1 A brief overview

The development of the Web of data and the increasing number of available knowledge graphs come along with several research problems involving various research fields such as information retrieval, natural language processing, databases, machine learning, and knowledge representation and reasoning [22]. For example, the *construction* of knowledge graphs involves the *representation* of the knowledge of a domain. Facts, entities, and relations can potentially be *automatically extracted* from existing sources. Such sources can be semi-structured data (*e.g.*, the HTML DOM trees of pages) or unstructured data (*e.g.*, plain text). However, knowledge graphs resulting from automatic extraction or crowdsourcing may not reach full coverage (*i.e.*, contain information about every entity of the world) and may not be fully correct [132, 176]. Hence, methods should be designed to *refine* these knowledge graphs, for example to complete them or detect erroneous statements. Knowledge graphs can also support tasks related to search engines such as interpreting the meaning of queries, question answering, and enhancing search results (see Figure 1.3). In this thesis, we focus on the two specific problems of *matching* and *mining* knowledge graphs that we describe more precisely in the following paragraphs.

1.3.2 Matching knowledge graphs

The decentralized nature of the Web of data leads to the concurrent publication and edition of knowledge graphs. Consequently, there are overlaps, *i.e.*, knowledge graphs may describe similar units (*e.g.*, entities, facts, etc.) while differing in quality, completeness, granularity, and vocabularies. This overlap may also arise within a knowledge graph as the construction process

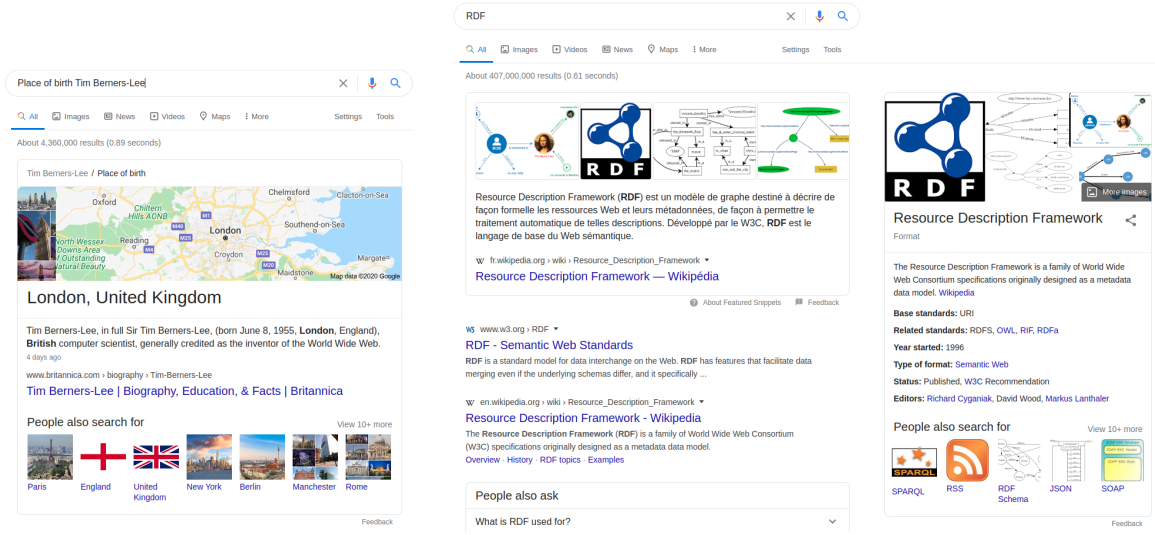


Figure 1.3: Examples of how knowledge graphs can be used for question answering (query: "Place of birth Tim Berners-Lee" – on the left) or to enhance search results by displaying additional information (query: "RDF" – on the right).

often integrates and aggregates different sources. Overlapping units may be exact duplicates, weakly related, complementary, or one unit may be more specific than another. For example, a knowledge graph may state that "Alice is the mother of Carl", which is more specific than the statement "Alice is a relative of Carl" from another knowledge graph.

Consequently, one major task lies in *matching* knowledge graphs, *i.e.*, identifying identical, more specific, or similar units within and across knowledge graphs. The matching process results in *alignments* that are correspondences between units. Matched units can be entities, ontology classes, predicates, facts, etc. This process is challenging as it needs to tackle the inherent heterogeneity of different knowledge graphs in terms of vocabularies, languages, granularities, etc. However, to overcome such heterogeneity issues, matching approaches can use different features offered by knowledge graphs such as reasoning mechanisms and knowledge formalized in ontologies. The matching task is at the center of the ontology matching research field [58]. It is also related to the reconciliation in databases [1] that aims at managing co-existing and independently modified replicas of the same database. We further detail existing approaches in Subsection 2.2.1.

Numerous downstream applications benefit from alignments resulting from the matching process, which illustrates the importance of this task. For example, fact-checking approaches can use these alignments to highlight agreements or contradictions between different sources. Alignments interlink knowledge graphs and thus offer a consolidated view of the knowledge the graphs conjointly express. Knowledge graphs can then be completed with facts from others, merged (*i.e.*, creating a new knowledge graph fusing them), or reconciled (*i.e.*, their content can be harmonized) [58]. We will introduce in Subsection 1.3.3 that knowledge graphs can be mined to discover additional knowledge. By aligning knowledge graphs and then mining the consolidated view, we can also expect an increased number of results or an improvement in their quality. Different versions of the same knowledge graph can also be aligned to evaluate, *e.g.*, the concept drift between the versions.

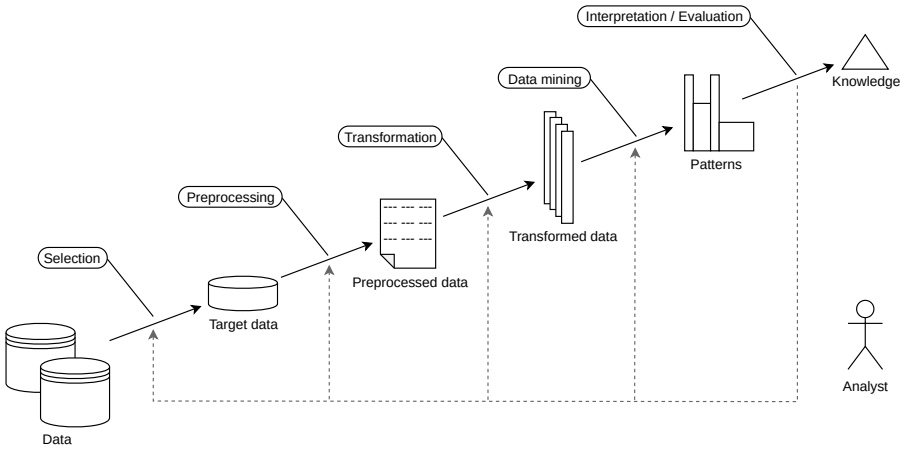


Figure 1.4: The process of Knowledge Discovery from Databases (KDD) (from Fayyad et al. [59]).

1.3.3 Mining knowledge graphs

Previously, we mentioned that the notions of data, information, and knowledge can be seen as built on top of each other. Somewhat in accordance with the definition of knowledge as assimilated information and data, it is possible to discover new knowledge from data. Such a process is called Knowledge Discovery from Databases (KDD) and was defined by Frawley et al. [62] as *"the non-trivial extraction of implicit, previously unknown, and potentially useful information from data"*. Fayyad et al. [59] modeled this process by the successive steps depicted in Figure 1.4. First, a part of the available data is *selected* for the KDD process, based on the objectives and the domain knowledge of an analyst. This selection can reduce the data samples or the considered attributes. Selected data are then *preprocessed*, for example to handle missing values or remove noisy data, before being *transformed* into a format accepted by the *mining algorithm*. The choice of this algorithm depends on the task, *e.g.*, regression, classification, or clustering. The patterns output by the mining algorithm are *interpreted* and *evaluated* by the expert to assess, *e.g.*, their validity or their usefulness. It should be noted that the term *mining* denotes a specific step of the KDD process but is sometimes used to refer to the whole process itself. We adopt this simplification in this thesis.

It is noteworthy that the KDD process is iterative, interactive, and depends on the knowledge that the *analyst* possesses about the considered domain. We saw that knowledge can also be formalized to be consumed and interpreted by software agents. That is why research works propose to integrate such formalized knowledge in the KDD process, leading to Knowledge Discovery guided by Domain Knowledge (KDDK) [100, 143]. In this framework, knowledge graphs can be involved in all the steps of the KDD process (see Subsection 2.2.2) [143]. In particular, it is possible to consider knowledge graphs as the input of the KDDK process to discover additional knowledge from knowledge, or meta-knowledge about knowledge. Recall that, according to Definition 1, even facts and statements represented in knowledge graphs are considered knowledge. However, the mining of knowledge graphs raises several research questions such as:

- How to tackle scalability issues arising when mining ever-growing knowledge graphs?
- What features can be mined from knowledge graphs to be used as attributes in mining

algorithms? [134, 140]

- How to ensure mining algorithms consider the formal semantics and domain knowledge associated with knowledge graphs?
- Since knowledge graphs are designed to be consumed both human and software agents, can they help building interpretable features, and thus play a role in explainable artificial intelligence?

1.4 Application context: pharmacogenomics

The work presented in this thesis has been conducted within the *PractiKPharma* project[5] [38] (2016–2020), funded by the French National Research Agency[6]. This project aims at studying approaches in Computer Science to extract and compare knowledge in the biomedical domain of pharmacogenomics (PGx).

Pharmacogenomics studies the influence of genetic factors in drug response phenotypes. This knowledge is beneficial in precision medicine, which aims at tailoring drug treatments to patients to reduce adverse effects and maximize drug efficacy [30]. Units of knowledge in PGx have typically the form of n-ary relationships, relating a set of drugs, a set of genetic factors, and a set of phenotypes (Figure 1.5). Such a relationship states that a patient being treated with the specified drugs, while having the specified genomic variations will be more likely to experience the given phenotypes, *e.g.*, adverse effects. To illustrate, Figure 1.6 depicts three well-studied PGx relationships stating that patients with different variants of the CYP2D6 gene may experience different reactions to the very same codeine treatment. PGx relationships can be seen as statements, and thus constitute knowledge in view of our adopted Definition 1. PGx knowledge originates from distinct sources:

State-of-the-art knowledge originates from expert databases and articles from the biomedical literature. It is noteworthy to mention PharmGKB [173] that is the reference database in the PGx domain. It is manually curated by biologists, *i.e.*, they add knowledge units to PharmGKB after a literature review.

Observational knowledge originates from the mining of observational data such as Electronic Health Records of hospitals (EHRs).

It is noteworthy that state-of-the-art knowledge in PGx may lack validation or clinical counterpart [81], for example, due to reduced cohorts of patients. Such knowledge remains to be further studied before being implemented in clinical practice. For example, Figure 1.7 shows that only 10% of the PGx relationships in PharmGKB have a level of evidence of "1" or "2", corresponding to relationships implemented in clinical guidelines, or with studies showing at least a moderate evidence of association. In contrast, the remaining 90% have a level of evidence of "3" or "4", corresponding to unreplicated studies, multiple studies showing lack of evidence, or non-significant studies.

The PractiKPharma project formulates the hypothesis that PGx can benefit from matching approaches developed in Computer Science. Indeed, matching the sources of PGx knowledge to reconcile them would provide a consolidated view on the knowledge of this domain, possibly enabling the validation of state-of-the-art relationships by EHRs knowledge. For this matching

[5] `http://practikpharma.loria.fr`
[6] Agence Nationale de la Recherche

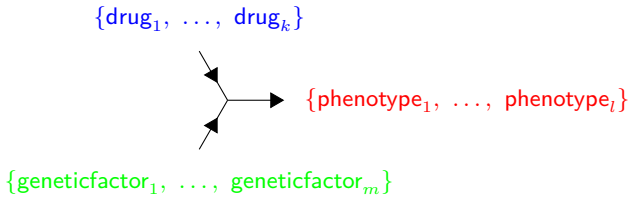


Figure 1.5: Graphical representation of an abstract pharmacogenomic relationship.

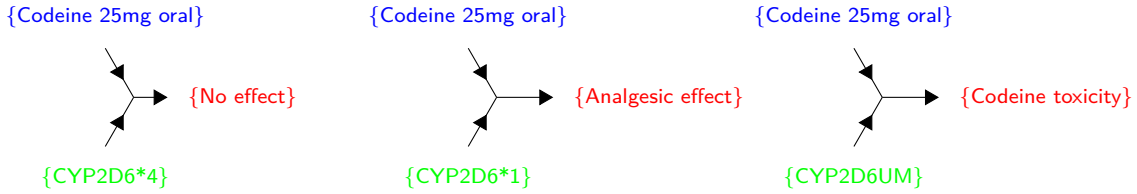


Figure 1.6: Three PGx relationships presenting the influence of the variants of the CYP2D6 gene in patient reactions to the very same codeine treatment. For example, patients having the CYP2D6*4 variant will not experience the expected effect of the codeine treatment.

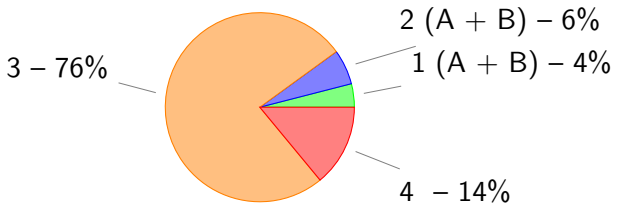


Figure 1.7: Distribution of levels of evidence of pharmacogenomic relationships in PharmGKB (2019-07-05 version). Levels 1 and 2 correspond to PGx relationships implemented in clinical guidelines, or with studies showing at least a moderate evidence of association. These two levels are both divided into two sublevels A and B that we aggregate here. Levels 3 and 4 correspond to PGx relationships with unreplicated studies, multiple studies showing lack of evidence, or non-significant studies.

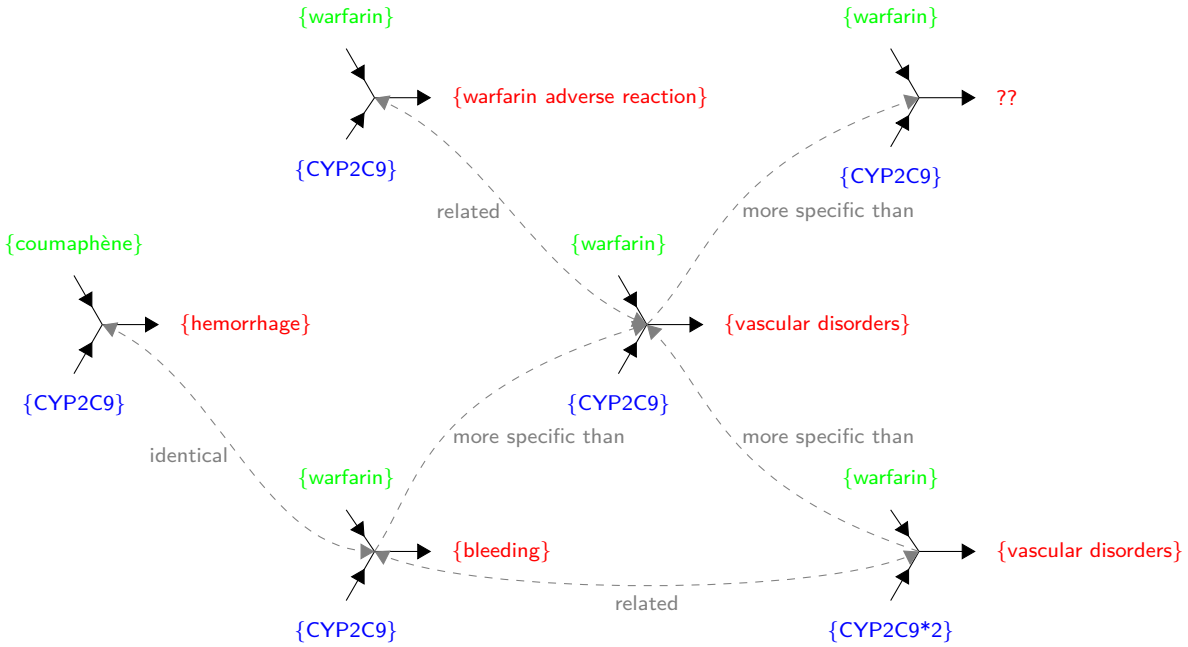


Figure 1.8: Example of heterogeneity issues and expected matching results (dashed gray arrows) between PGx relationships of various provenances. The phenotype is unknown for one relationship, coumaphène is the French word for warfarin, hemorrhage is a synonym of bleeding, CYP2C9*2 is a gene variant that is more specific than the gene CYP2C9 itself, bleeding is more specific than vascular disorder, and vascular disorders is related to warfarin adverse reaction.

procedure, the PractiKPharma project made the choice to rely on existing knowledge graphs and ontologies about Life Sciences, which constitute a huge part of available Linked Open Data [143] (in red in Figure 1.2). These existing works facilitate the representation of PGx relationships, by already defining knowledge about their components, *i.e.*, drugs, genetic factors, and phenotypes. PractiKPharma consequently proposed to leverage Semantic Web and Artificial Intelligence technologies and research works to enable the reconciliation of PGx knowledge sources.

Conversely, it is noteworthy that this real-world application also benefits Computer Science approaches by raising several important issues and motivating the development of new approaches to tackle them. Indeed, PGx relationships illustrates the need to develop matching approaches that target n-ary relationships. Due to their various sources, another challenge of the matching process resides in facing their inherent heterogeneity, in terms of languages, vocabularies, granularities, and completeness (*i.e.*, unknown/missing components). To illustrate, Figure 1.8 depicts some examples of heterogeneity issues in PGx relationships and the results we would expect to obtain from a matching process. This real-world application and its inherent challenges motivated the contributions detailed in this manuscript.

1.5 Outline of this thesis and contributions

This manuscript is organized as follows.

In **Chapter 2**, we review the ecosystem of knowledge graphs, some of their related tasks, and existing approaches to perform these tasks. Particularly, we review some of the existing

definitions of *knowledge graph* and related concepts, and explicit those we use in the following chapters. Then, we introduce the broad perspective of knowledge graph *refinement* before focusing on the task of *matching* within and across knowledge graphs. We also detail the task of *mining* knowledge graphs to discover additional knowledge units. We outline some existing works tackling these two tasks, which allows to illustrate the variety of approaches related to knowledge graph refinement and mining.

Chapter 3 details our work on representing PGx knowledge. In this chapter, we motivate the need for a knowledge graph focusing on the integration of such knowledge. We use Semantic Web technologies to represent, trace, and integrate PGx knowledge units coming from three sources: the PharmGKB reference database, the biomedical literature, and EHRs. To this aim, we define PGxO, a small ontology to represent PGx knowledge units. We instantiate PGxO with knowledge units from the three mentioned sources, resulting in the knowledge graph we named PGxLOD. Because of its size and the particularities related to PGx knowledge (*e.g.*, arity, heterogeneity), PGxLOD offers us an interesting and unified experimental setting to test the approaches presented in other chapters. This chapter is based on two contributions: a workshop article published in *NETTAB 2017* that introduces the preliminary version of PGxO [114], and a journal article published in *BMC Bioinformatics* (2019) that details PGxO and PGxLOD [115].

In **Chapter 4**, we present a knowledge-based approach to match n-ary relationships represented in a knowledge graph. We propose a general framework that models these relationships as n-ary tuples whose arguments are sets of individuals. We perform the matching between tuples by defining five rules that are mathematically well-founded. These rules compare the arguments of two tuples and conclude on their relatedness among a scale of five levels, such as being equivalent, more specific, or weakly related. The main originality of these rules resides in the use of preorders arising from domain knowledge to face heterogeneity issues when comparing tuples. This chapter is based on a conference article published in *ICCS 2020* [113].

Results of this rule-based approach highlighted the need for flexible comparison approaches to deal with heterogeneity issues. Hence, **Chapter 5** explores the use of graph embedding, and especially Graph Convolutional Networks (GCNs) [92, 146], to match entities in knowledge graphs. GCNs learns an embedding for each entity, *i.e.*, a vector representation [27], based on its neighborhood in a knowledge graph. Here, we aim at learning embeddings whose distance reflects the similarity between their entities. This method is more flexible than a rule-based process due to the continuous representation of embeddings. Additionally, relatedness conditions are learned from examples and not given as rules. Thus, embeddings may be more resilient to heterogeneous representations and may yield more results. Within this task, we investigate the interplay between domain knowledge and GCN models with the two following main focuses. First, we measure the improvement in matching results when considering various inference rules associated with domain knowledge, independently or combined. Second, while our GCN model is agnostic to the exact alignment relations (*e.g.*, equivalence, weak similarity), we observe different distance distributions in the embedding space for these relations, which somehow corresponds to their rediscovery by the model. This chapter is an extension of the preliminary results described in a workshop article published in *DL4KG 2019* [120].

In **Chapter 6**, we consider a particular set of nodes of interest, called *seed nodes*, and we mine features associated with them such as their neighboring nodes or the paths they can root in a knowledge graph. Entities involved in such paths may instantiate ontology classes. Thus, we can replace entities by their instantiated classes to generate path patterns that may characterize more seed nodes than paths. For example, based on our running example, the path $\xrightarrow{\texttt{is-friend-of}}$ `Bob` may only be associated with `Alice`, while the path pattern $\xrightarrow{\texttt{is-friend-of}}$ `Person`

may be associated with more entities. Mining such patterns immediately entails scalability issues due to the combinatorial nature of knowledge graphs. We address these issues by proposing a pattern mining approach that relies on a set of constraints (*e.g.*, support or degree thresholds), a pruning of redundant patterns based on the hierarchy of ontology classes, and the *monotonicity* of the support of paths and path patterns. This chapter is based on a conference paper published in *ALGOS 2020* [112].

In **Chapter 7**, we propose to refine and mine knowledge graphs using Concept Annotation, an extension to Formal Concept Analysis (FCA) that we propose. FCA [70] is a mathematical framework that groups objects in formal concepts based on their common attributes. These concepts are organized in a hierarchical structure called a concept lattice. The lattice enables to identify equivalent objects (in the same concept) or more precise ones (in subsumed concepts). We use such a hierarchical organization to *(i)* organize classes of an ontology in a hierarchy, *(ii)* suggest alignments between classes of different ontologies, and *(iii)* discover frequent domain $\rightarrow$ range associations for predicates, *i.e.*, associations $C_1 \rightarrow C_2$ where C_1 and C_2 are two ontology classes frequently appearing as domain and range of a predicate in its assertions. This chapter is based on a conference article published in *ISMIS 2017* [116], a poster published in *BDA 2017* [117], and a workshop article published in *FCA4AI 2018* [118].

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

Chapter 2

Ecosystem, refinement, and mining of knowledge graphs

Contents

2.1 Ecosystem of knowledge graphs 23
2.1.1 Ontology, knowledge base, knowledge-based system, and knowledge graph 23
2.1.2 The Web of data and knowledge graphs 26
2.1.3 The Open World Assumption and its consequences 29
2.1.4 Examples of ontologies and knowledge graphs 31
2.2 Refinement and mining of knowledge graphs 32
2.2.1 Refinement of knowledge graphs 32
2.2.2 Mining of knowledge graphs 37
2.3 Two frameworks to refine and mine knowledge graphs 40
2.3.1 Formal Concept Analysis and its extensions 40
2.3.2 Knowledge graph embedding 45

In this chapter, we further present related work about knowledge graphs. First, we review the ecosystem of associated concepts and technologies, and we make explicit the definitions used in the remainder of this manuscript (Section 2.1). Second, we detail the two research problems of *refining* and *mining* knowledge graphs and present some categorizations and examples of approaches (Section 2.2). Lastly, we present the frameworks of Formal Concept Analysis and graph embedding, and illustrate their use for the tasks of refining and mining knowledge graphs (Section 2.3).

2.1 Ecosystem of knowledge graphs

2.1.1 Ontology, knowledge base, knowledge-based system, and knowledge graph

The concept of *knowledge graph* is tightly related to those of *ontology*, *knowledge base*, and *knowledge-based system*. Even if they are sometimes interchangeably used [57], they present different characteristics that we outline in the following paragraphs.

Ontology

An *ontology* is defined by Gruber [74] as "*an explicit specification of a conceptualization*". A conceptualization is a particular model of a domain of interest. In this thesis, we consider that such a conceptualization consists of classes and relations, which we define as follows:

Definition 3 (Classes [58])**.** Classes are interpreted as sets of individuals of the domain which *instantiate* the classes. They are also named concepts or types. For example, in a model representing friends, the class `Person` can be instantiated by the individuals `alice` and `bob`.

Definition 4 (Relations [58])**.** Relations are interpreted as subsets of the Cartesian product of their domain and range. For example, a relation `is-a-friend-of` can link friends together. They are also named properties or predicates. It is possible to differentiate *object properties* that link individuals together, such as `is-a-friend-of`, from *data properties* that link individuals to data values, such as `has-age`.

In an ontology, classes (respectively predicates) are organized in a hierarchy by the subsumption relation denoted by $\sqsubseteq$. This relation is a partial order. Given two classes `A` and `B`, if $\texttt{A} \sqsubseteq \texttt{B}$, then all instances of `A` are also instances of `B`. The class `A` can then be seen as a subclass or "more specific" than the class `B`.

The *explicit* character of the conceptualization underlines that its constituents are associated with a description that allows them to be manipulated and interpreted both by human and software agents. Gruber [74] list two kinds of descriptions: textual definitions and formal axioms. First, definitions associate constituents of an ontology with a human-readable text describing what these constituents actually denote. Second, formal axioms constrain the interpretation and the use of such constituents. These formal axioms must be represented in a formalism that allows their manipulation by both human and software agents. To this aim, Description Logics [9] have been developed as a family of knowledge representation languages. Such languages are based on decidable fragments of first-order logic. Hence, they are well-founded and allow to unambiguously represent knowledge. They are also equipped with logical deduction which enables to reason about the represented knowledge by inferring additional information from the facts explicitly stated. Different Description Logics provide different logic constructions and thus, different expressiveness levels and reasoning complexities. The choice of the expressiveness level then depends on the intended application. In this view, Feilmayr and Wöß [60] enrich the definition of *ontology* with the relation between expressiveness and complexity as follows: "*a formal, explicit specification of a shared conceptualization that is characterized by high semantic expressiveness required for increased complexity*".

Knowledge base

A conceptualization of a domain can also involve *individuals*, defined as follows:

Definition 5 (Individuals [58])**.** Individuals are interpreted as basic individuals of a domain. They are also named instances or entities. As aforementioned, in a model representing friends, `alice` and `bob` are two individuals.

In this thesis, we previously restrained the definition of an ontology to classes and predicates. Hence, we consider that an *instantiated ontology* constitutes a *knowledge base*. Our definition is in accordance with the architecture of a knowledge base proposed by Baader et al. [9] and depicted in Figure 2.1. In this architecture, the *description language* can correspond to Description Logics, which allow *reasoning*. The *terminological box* (TBox) contains axioms related to

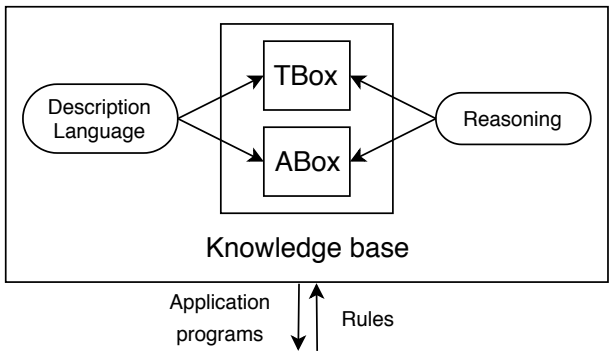


Figure 2.1: Architecture of a knowledge base, as proposed by Baader et al. [9]. The TBox is the *terminological box* and contains axioms related to classes and predicates, which corresponds to our definition of *ontology*. The ABox is the *assertional box* and contains assertions or facts about individuals. These facts use predicates defined in the TBox or indicate that an individual instantiate a class from the TBox.

classes and predicates. Thus, it corresponds to what we defined as an *ontology*. The *assertional box* (ABox) contains facts. Such facts represent the relationships holding between individuals by using predicates of the TBox (*e.g.*, $\texttt{is-a-friend-of}\,(\texttt{alice}, \texttt{bob})$) or indicate that an individual instantiates a class from the TBox (*e.g.*, $\texttt{Person}\,(\texttt{alice})$). In the RDF language that we introduced in Section 1.2 and further detail in Subsection 2.1.2, these facts take the form of triples, *e.g.*, $\langle \texttt{alice}, \texttt{is-a-friend-of}, \texttt{bob} \rangle$ and $\langle \texttt{alice}, \texttt{type}, \texttt{Person} \rangle$. A *relational box* (RBox) is sometimes added to the TBox and ABox of a knowledge base. It contains the axioms related to predicates (*e.g.*, subsumption, equivalence, or inverse axioms between predicates). In this case, the TBox only contains the axioms related to classes.

The previous definitions are widely adopted but are not unique. For example, Davies et al. [45] define a knowledge base as "*a data set with formal semantics that can contain different kinds of knowledge, for example, rules, facts, axioms, definitions, statements, and primitives*". Similarly to Baader et al. [9], it involves formal semantics which can be, for example, Description Logics. Ehrlinger and Wöß [57] consider that an ontology can hold instances and constitutes one type of knowledge base in view of the definition of Davies et al. [45].

Knowledge-based system

A *knowledge-based system* is defined by Akerkar and Sajja [4] as "*a system that uses artificial intelligence techniques in problem-solving processes to support human decision-making, learning, and action.*". In view of Figure 2.1, a knowledge base can support a knowledge-based system. Indeed, as indicated by Baader et al. [9], an application program can interact with a knowledge base. Hence, such a program can carry out problem-solving activities by relying on the represented knowledge and the reasoning capabilities provided by the knowledge base.

Knowledge graph

The definition of *knowledge graph* remains contentious since no definition has been widely adopted up to this date. The popularization of the expression has been associated with a blog post from Google in 2012 [152]. However, it has been used in research papers that can

be dated back to 1973. It can also be considered as a derivation from similar notions such as the *existential graph* of Charles S. Peirce. In the following, we will discuss some of the recent definitions but the interested reader can find detailed historical reviews in the works of Bergman [13] and Hogan et al. [79].

Among the recent definitions, different criteria are proposed to characterize a knowledge graph. For example, Ehrlinger and Wöß [57] suggest that a knowledge graph is somehow superior to a knowledge base, and thus require that a knowledge graph "*acquires and integrates information into an ontology and applies a reasoner to derive new knowledge*". Paulheim [132] also lists several criteria such as the coverage of several domains. However, he also suggests that, from a broader perspective, any graph-based representation of knowledge could constitute a knowledge graph. In this view, in a Dagstuhl Seminar [20] in 2019, the following definition was proposed: "*a graph of data with the intent to compose knowledge*". This definition does not specify the concrete graph model and considers that any graph model is sufficient (*e.g.*, directed edge-labeled graphs, property graphs). However, authors acknowledge that some specific models may be more convenient depending on the considered application. The *intent to compose knowledge* is viewed as a process that enriches and improves the interpretability of the graph by humans and machines. Such a process involves the extraction and representation of knowledge by, *e.g.*, describing the formal semantics of used terms, interlinking the graph with external data sets, or adding contextual information such as provenance. Similarly, Krötzsch [93] proposes a definition that does not constrain the type of graph. He also identifies the three main characteristics of *(i)* normalization, *(ii)* connectivity, and *(iii)* context. That is to say, *(i)* information is decomposed into small units seen as edges in a graph, *(ii)* knowledge emerges from such relationships between units, and *(iii)* units are enriched with contextual information such as temporal validity, provenance, or trustworthiness.

The two previous definitions are close to the definition of knowledge graph that we adopt in this thesis (Definition 2). Indeed, the intent to compose knowledge [20] corresponds to the accumulation and conveyance of knowledge of the real world, which can be seen as emerging from the relationships holding between units [93]. Such an intent includes the contextualization proposed by Krötzsch [93]. Although it is not explicitly stated in Definition 2, Hogan et al. [79] also consider the key role of schema, identity, and context to face the inherent diversity resulting from assembling knowledge of diverse sources. Such a contextualization is taken into account in our work when representing pharmacogenomic knowledge (see Chapter 3).

2.1.2 The Web of data and knowledge graphs

Technologies of the Web of data

We previously mentioned that various graph formalisms can be used for knowledge graphs. In this thesis, we focus on knowledge graphs represented using technologies developed within the Semantic Web paradigms [15], namely RDF, RDFS, and OWL. Such technologies are standardized by the Word Wide Web Consortium (W3C)[7]. They are at the core of Linked Open Data [17] which enable the Web of data, thus realizing the Semantic Web vision.

In the Web of data, entities, predicates, and classes are identified by a *Uniform Resource Identifier* (URI), *i.e.*, a string such as `http://pgxo.loria.fr/Drug`. In this URI, we can identify two parts: the *namespace* `http://pgxo.loria.fr/` and the *resource* `Drug`. The namespace can be abbreviated by a prefix, such as `pgxo`, leading to represent the URI in the shortened form `pgxo:Drug`. URIs have been extended into *Internationalized Resource Identifiers* (IRIs) to

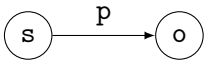
[7] `https://www.w3.org/`

enable the use of additional characters than those encoded in ASCII. However, to simplify, we will only speak about URIs. When a URI can be dereferenced, it is also a *Uniform Resource Locator* (URL). This dereferenciation allows an agent to obtain a description of the resource identified by the URI. Thanks to HTTP content negotiation, such a description can be a text when accessed by a human, or a set of formal axioms when accessed by a software agent or a knowledge engineer (*i.e.*, someone who knows how to read knowledge representation languages). It is a best practice to provide dereferenceable URIs to ease navigation and discovery in the Web of data [17, 174].

Statements or facts involving these URIs are expressed in the Resource Description Format language (RDF) [41] by *triples*:

$$\langle \texttt{subject}, \texttt{predicate}, \texttt{object} \rangle \in (U \cup B) \times U \times (U \cup L \cup B)$$

where U denotes the set of URIs, L denotes the set of literals (*e.g.*, strings, integers, dates), and B denotes the set of blank nodes. Blank nodes are anonymous entities without URIs that can be used, for example, to represent complex attributes. Hence, an address can be represented as a blank node that is linked to the street, the number, and the city. An $\langle \texttt{s}, \texttt{p}, \texttt{o} \rangle$ triple can be seen as the logical formula $\texttt{s}(\texttt{p}, \texttt{o})$ or as an edge labeled by `p` that links a node labeled by `s` to a node labeled by `o`:



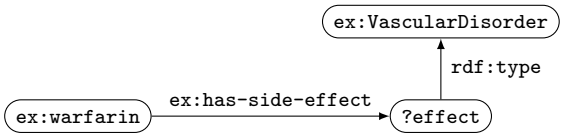
Consequently, "data sets" in the Web of data can be seen as knowledge graphs of the form of *direct labeled multigraph*. It should be noted that RDF provides a predicate `rdf:type` for instantiation axioms. Hence, the triple $\langle \texttt{alice}, \texttt{rdf:type}, \texttt{Person} \rangle$ states that the individual `alice` instantiates the class `Person`.

RDF is not equipped to define ontologies. To this aim, the two standards named RDFS and OWL have been proposed and recommended by the W3C. First, the RDF Schema (RDFS) [26] extends RDF to define light ontologies. Indeed, it provides the two predicates `rdfs:subClassOf` and `rdfs:subPropertyOf` that are used in subsumption axioms between classes or predicates. Additionally, RDFS enables to constrain the domain and the range of predicates. This light expressiveness is associated with reasoning mechanisms that, for example, conclude on the type of an individual based on the transitivity of subsumption axioms or the domain and range of predicates. However, some applications may require additional expressiveness, which is enabled by the Web Ontology Language (OWL) [166]. This language enables to encode Description Logics [9], and thus offers more complex possibilities. For example, it is possible to define classes as disjoint or equivalent. Predicates can be symmetric, reflexive, or defined as inverses. OWL 2 provides three different profiles corresponding to syntactic subsets of all the possible axioms in OWL 2. These restrictions in the set of usable axioms result in computational benefits, for example, in the complexity of reasoning tasks. Hence, it is possible to find a balance between the complexity of some reasoning tasks and the level of expressiveness required by an application.

Designing highly expressive ontologies is time-consuming and requires some expertise in knowledge representation languages. Not all applications can afford such costs or require the high level of expressiveness provided by ontologies. In such cases, it is possible to use the Simple Knowledge Organization System (SKOS) [105]. SKOS is a data model designed for the representation of thesauri, taxonomies, or controlled vocabularies. In SKOS, so called "concepts" can be organized hierarchically with predicates such as `skos:broadMatch` and `skos:narrowMatch`. Additional predicates are available to represent various relations between SKOS concepts, such

```
SELECT ?effect WHERE {
  ex:warfarin ex:has-side-effect ?effect .
  ?effect rdf:type ex:VascularDisorder .
}
```

(a) SPARQL query



(b) Corresponding graph pattern

Figure 2.2: Example of a SPARQL query to retrieve the side effects of `warfarin` that instantiate the class `VascularDisorder`. The prefix `ex` is an example prefix frequently used and represents the namespace `http://example.org/`.

as `skos:related` or `skos:member`. Unlike OWL, SKOS is not considered as a formal knowledge representation language. Hence, statements in SKOS should not be considered as formal axioms or facts that model the world but rather as informal descriptions of SKOS concepts. However, SKOS can be used conjointly with RDFS and OWL. For example, in Chapter 3, we represent pharmacogenomic knowledge using RDFS and OWL, but results of our matching approach described in Chapter 4 require a less formal representation and thus are encoded with SKOS.

To be used in applications, information can be retrieved from RDF-based knowledge graphs with queries expressed in the SPARQL Protocol and RDF Query Language (SPARQL) [160]. SPARQL relies on *basic graph patterns* that are represented by triples in the `WHERE` clause of the query (see Figure 2.2 for an example). Besides basic graph patterns, SPARQL provides several mechanisms such as filters for data values and grouping mechanisms.

Recall that some discussed definitions of knowledge graph involve the addition of contextual information such as provenance. RDF does not provide *per se* a mechanism for such contextualization. However, it is possible to add metadata in the form of RDF triples that use specific vocabularies developed for such purposes. For example, the PROV-O [96] ontology is a W3C recommendation that aims at representing provenance information. We further present this ontology and use it when representing pharmacogenomic knowledge of various provenances in Chapter 3.

Linked Open Data and FAIR principles

The Semantic Web vision promotes the publication of data sets on the Web and their interlinking with other published data sets. The aim is to facilitate a conjoint reuse of data sets that may go beyond their original purposes and coverage. In this view, principles are available and define best publishing practices. For example, Berners-Lee [14] recommends to follow the four following "rules" called the Linked Open Data principles:

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

To evaluate if a data set complies with the expectations of Linked Open Data, Berners-Lee [14] also proposes an incremental scale of "five stars". Hence, a data set must be

⋆ Published on the Web (with an open license, to be Open Data)

⋆ ⋆ Published in a machine-readable format

⋆ ⋆ ⋆ Published in a machine-readable format that is non-proprietary

⋆ ⋆ ⋆ ⋆ Published using open standards of the W3C (*e.g.*, RDF)

⋆ ⋆ ⋆ ⋆ ⋆ Linked to other RDF data sets

Interestingly, the last level is one of the considered task when *refining* knowledge graphs (see Subsection 2.2.1).

Alternatively, Wilkinson et al. [174] introduce the FAIR principles, which can be summarized as follows:

Findable Human and software agents can easily find data sets that may benefit them. This aim is achieved through, *e.g.*,the use of unique and persistent identifiers, and the indexing of data sets in searchable resources such as LOD cloud[8] or Google dataset search[9].

Accessible Agents can retrieve data by using their identifiers and standardized communication protocols. Hence, dereferenceable URIs and W3C standards allow such an accessibility.

Interoperable Data sets can be used conjointly with other data sets. For example, this criterion implies the use of a broadly applicable language for knowledge representation (*e.g.*, RDF) and vocabularies that respect the FAIR principles.

Reusable Data sets are reusable, which requires, *e.g.*, an accessible data license and detailed provenance metadata.

These principles initially targeted scholarly data. However, they are applicable and considered when publishing knowledge graphs in the Web of data. As illustrated, Linked Open Data principles enable to (partly) comply with FAIR principles.

2.1.3 The Open World Assumption and its consequences

Semantic Web paradigms and Description Logics make the *Open World Assumption* (OWA). This assumption differs from the *Closed World Assumption* (CWA) usually made in database systems. Under CWA, a statement that is not known to be true is false. On the contrary, under OWA, no inference or deduction can be made from a statement that is unknown. Example 1 illustrates the answer to a question under these two assumptions.

Example 1. Statement: warfarin causes vascular disorders
Question: does warfarin cause gastric disorders?
Answer under OWA: unknown
Answer under CWA: no

Content in knowledge graphs is always being improved in terms of completeness and correctness [132]. These incompleteness and ever-improving characteristics are coherent with the OWA. However, the OWA raises issues when using data mining or machine learning algorithms

[8] `https://lod-cloud.net/`
[9] `https://datasetsearch.research.google.com/`

on knowledge graphs. Indeed, such algorithms often make the CWA and require negative statements. True negative statements are seldom available in knowledge graphs. Thus, to train and evaluate algorithms or models, it is frequent to make the CWA and consider that absent statements are false. In this view, some approaches randomly generate negative statements. For example, it is possible to corrupt existing triples by changing their subject, object, or both. Advanced strategies for negative sampling have also been proposed such as the choice of an entity in the same domain as the entity to corrupt [83]. The main challenge in negative sampling resides in avoiding to generate false negative, *e.g.*, a corrupted triple that should have existed in the knowledge graph and thus should constitute a positive example.

The OWA also impacts the evaluation of data mining and machine learning algorithms. Indeed, metrics usually require true negative statements. For example, consider a rule $B \Rightarrow r(x, y)$ that predicts the triple $r(x, y)$ when a pair (x, y) makes the condition B true. Its confidence can be expressed as follows [66]:

$$\text{confidence}\,(B \Rightarrow r(x,y)) = \frac{|\,\{(x,y) \mid B \text{ is true and } \exists r(x,y)\}\,|}{|\,\{(x,y) \mid B \text{ is true}\}\,|}$$

Hence, pairs (x, y) such that B is true but $r(x, y)$ is absent are considered as negative statements, which corresponds to the CWA. To take into account the OWA made by knowledge graphs, some authors propose to *locally* close the world. In this view, Galárraga et al. [66] introduce the *Partial Completeness Assumption* (PCA). This assumption relies on the functionality of a predicate r, which is defined as follows:

$$\text{fun}(r) = \frac{|\{x \mid \exists y,\ r(x,y)\}|}{|\{(x,y) \mid \exists r(x,y)\}|}$$

A predicate r has a functionality equal to one when each of its subjects x is associated with one object y. The PCA considers that for predicates r having a high functionality, if a r-attribute of x is known, then all r-attributes of x are known. Hence, Galárraga et al. [66] define the *pca-confidence* as follows:

$$\text{pcaconf}\,(B \Rightarrow r(x,y)) = \frac{|\,\{(x,y) \mid B \text{ is true and } \exists r(x,y)\}\,|}{|\,\{(x,y) \mid B \text{ is true and } \exists r(x,y')\}\,|}$$

Here, only pairs (x, y) such that B is true and $r(x, y)$ does not exist but another triple $r(x, y')$ $(y' \neq y)$ exists are considered as negative statements. The PCA is similar to the *Local Closed World Assumption* (LCWA) [53]. Given a predicted triple $\langle s, p, o\rangle$, and $O(s, p)$ the set of objects associated with s through p, the triple is:

- considered correct if $O(s, p) \neq \emptyset$ and $o \in O(s, p)$;
- considered incorrect if $O(s, p) \neq \emptyset$ and $o \notin O(s, p)$;
- discarded if $O(s, p) = \emptyset$.

Hence, this assumption considers the knowledge graph as *locally complete* for s and p if $O(s, p) \neq \emptyset$. Similarly to the PCA, this local completeness is true for function or highly functional predicates. However, Galárraga et al. [66] suggest that the PCA is reasonable for knowledge graphs that integrate triples from one source since they usually contain all r-values or none. Dong et al. [53] empirically validate the LCWA by comparing performances on test sets where true and false statements are either obtained by making the LCWA or by human annotation.

Besides performance metrics, the OWA can also be directly considered into approaches. For example, d'Amato et al. [42] tackle the task of query answering, *i.e.*, given a query concept Q, they aim at determining the membership of an instance x. To comply with the OWA, they propose three values for this membership: $+1$ when x belongs to Q, -1 when x belongs to the negation of Q, and 0 when the membership cannot be determined. Interestingly, they also introduce the following performance metrics to compare a deductive reasoner and an inductive model on this task:

- *match rate*: counts the individuals that got the same classification from the deductive reasoner and the inductive model.
- *omission error rate*: counts the individuals whose membership was determined deductively (± 1) but not inductively (0).
- *commission error rate*: counts the individuals whose membership is determined both deductively and inductively but negated ($+1$ *vs.* -1 or -1 *vs.* $+1$).
- *induction rate*: counts the individuals whose membership was determined inductively (± 1) but not deductively (0).

These metrics implicitly take into account the OWA and provide a further comparison in addition to usual metrics such as *precision*, *recall*, and F_1*-score*.

In our contributions, we particularly consider the OWA when matching n-ary tuples with a rule-based approach (see Chapter 4).

2.1.4 Examples of ontologies and knowledge graphs

Numerous knowledge graphs and ontologies are available and can be discovered by consulting indexing resources. For example, as of August 2020, the LOD cloud[10] lists 1,449 knowledge graphs and ontologies in the form of Linked Open Data, and 883 ontologies are registered in the NCBO Bioportal[11] [126]. Here, we list some knowledge graphs and ontologies that are prominent or related to our application in pharmacogenomics. More detailed lists can be found in existing reviews and tutorials [79, 124, 132].

Among knowledge graphs, some are generic and cover multiple domains. For example, DBpedia [99] and YAGO [159] are both automatically built by extracting the semi-structured content from Wikipedia, such as infoboxes. Conversely, Wikidata [165] is collaboratively and manually built to supply Wikipedia (and others). Indeed, Vrandecic and Krötzsch [165] state that several articles of Wikipedia can involve the same fact. For example, Italian and English articles about Rome mention its population. This number is also used in the article "Cities in Italy" in the English Wikipedia. Thus, all these articles should be updated when the population changes. Such an update process is tedious with risks of inconsistencies. By connecting such articles to Wikidata, a single update in this knowledge graph could automatically trigger an update in all of them. Because of their generic coverage and their interlinking with other knowledge graphs, DBpedia, YAGO, and Wikidata have become major nodes of the LOD Cloud. In particular, DBpedia can be seen as a central hub. The construction of knowledge graphs can also consider other contents than semi-structured ones. For example, Never-Ending Language Learning (NELL) [29, 108] learns facts by extracting them from web pages. As for Knowledge Vault [53], it is automatically built from both structured and unstructured sources. Indeed, Dong

[10] `https://www.lod-cloud.net/`
[11] `http://bioportal.bioontology.org/`

et al. [53] extract triples with machine learning algorithms from text documents, HTML DOM trees, HTML tables, and human annotated pages. Knowledge from these sources is then fused with triples output by two algorithms performing a prediction task on the FreeBase knowledge graph [19].

Alternatively, some knowledge graphs are specialized in a domain. For instance, WordNet [106] is a linguistic knowledge graph that gathers semantic relations between words (*e.g.*, hypernyms, hyponyms). Regarding our application in pharmacogenomics, several knowledge graphs about Life Sciences exist, such as the ones output by the Bio2RDF project [54]. These knowledge graphs result from a transformation in Linked Open Data of databases in other formats (*e.g.*, relational databases). Additionally, specific ontologies model (part of) the pharmacogenomic domain. For example, Genomic CDS [145] aims at modeling and reasoning about genomic variations of patients to match these patients with appropriate guidelines and clinical decision support messages. We present and discuss additional examples in Section 3.1.

2.2 Refinement and mining of knowledge graphs

We previously introduced the diversity of research problems and fields associated with or about knowledge graphs (Section 1.3). Here, we propose to further detail the two problems of *refinement* (which encompasses *matching*) and *mining* of knowledge graphs.

2.2.1 Refinement of knowledge graphs

Knowledge graphs only represent a model of the real world. Thus, they are never perfect and can always be improved or *refined* [22, 132]. This refinement process can aim at improving the *completeness* of a knowledge graph (*i.e.*, does the knowledge graph contain all the knowledge of the represented domain?) or its *correctness* (*i.e.*, is the knowledge contained in the knowledge graph correct?) [176].

Categorizations of knowledge graph refinement methods and their evaluations

Numerous works exist in this area of research, as underlined by recent surveys and tutorials about knowledge graphs and their research problems [27, 79, 124, 132, 168]. Interestingly, Paulheim [132] proposes several criteria to categorize refinement approaches:

- the goal of the refinement;
- the type of data used by the approach;
- the targeted kind of information.

First, the goal of the refinement can be, as aforementioned, the completion of the knowledge graph or its correction (also called error detection). Second, approaches can use different sources of data to perform the refinement. Indeed, *internal approaches* only consider data in the knowledge graph whereas *external approaches* consider additional data such as text corpora or images [27, 83]. Third, approaches can differ in the kind of information to refine. For example, some works focus on triples $\langle s, p, o \rangle$. For such a refinement, Cai et al. [27] and Wang et al. [168] distinguish *link prediction* from *triple classification*. Link prediction aims at predicting a subject s, given a predicate p, and an object o, *i.e.*, finding the subject s of a triple $\langle ?, p, o \rangle$, or predicting an object o, given a subject s and a predicate p, *i.e.*, finding the object o of a

triple $\langle s, p, ? \rangle$. As for triple classification, it aims at classifying a given triple $\langle s, p, o \rangle$ as true or false. It is noteworthy that link prediction and triple classification are closely related. Indeed, classifying a triple $\langle s, p, o \rangle$ as true can be viewed as predicting a link. Approaches can consider any kind of predicates p or specific predicates. Instead of triples, works can also focus on entity type information [132], also called entity classification [168]. Given an entity, the objective is to predict its type among all classes available in an ontology. In knowledge graphs, an entity can instantiate several classes. Recall that classes are hierarchically organized in the ontology by the subsumption relation. That is why this task can be treated as a multiclass multilabel classification or a hierarchical classification [132]. Alternatively, it can be seen as a `is_a` link prediction task [132, 168]. Refinement methods can also focus on the schema of the knowledge graph, *i.e.*, the associated ontology, and propose its enrichment or correction. To illustrate, such an enrichment could consist in discovering definitions for ontology classes [5], or subsumption axioms between them. Here again, the discovery of subsumption axioms can be seen as a `subClassOf` link prediction task, which illustrates that categories of approaches are not necessarily disjoint.

There are numerous refinement approaches in the literature. To illustrate, we present here some works that particularly inspired our contributions. Galárraga et al. [66] developed AMIE [66], a rule mining system that makes the *Partial Completeness Assumption*. This system mines Horn rules, *i.e.*, rules of the form $B_1 \wedge B_2 \wedge \cdots \wedge B_n \Rightarrow r(x, y)$. Such a rule predicts the triple $r(x, y)$ for a pair (x, y) that validates conditions B_i. These conditions represent triples that must exist in the knowledge graph. For example, Galárraga et al. [66] mined the following rules from DBpedia [99] and YAGO [159]:

$$hasAdvisor(x, y) \wedge graduatedFrom(x, z) \Rightarrow worksAt(y, z)$$

$$hasWonPrize(x, G.W.Leibniz) \Rightarrow livesIn(x, Germany)$$

Interestingly, AMIE is able to mine rules that involve variables (*e.g.*, x) and constants (*e.g.*, *Germany*). Similarly, Stadelmaier and Padó [156] propose to predict a triple $\langle s, p, t \rangle$ based on the *paths* existing between s and t. In their Context Path Model [156], paths are seen as sequences of predicates between the source and target entities, *i.e.*, $s \xrightarrow{p_1} \xrightarrow{p_2} \cdots \xrightarrow{p_{k-1}} \xrightarrow{p_k} t$. Alternatively, Vandewiele et al. [164] focus on entity classification. They propose to learn a decision tree to classify entities based on paths of a knowledge graph. The authors suggest that the predictions of their model are explainable as they are obtained by a "white-box" model (*i.e.*, the decision tree) that combines interpretable features (*i.e.*, paths from a knowledge graph). Interestingly, this system considers paths with their intermediate predicates and entities, *i.e.*, $\texttt{root} \xrightarrow{p_1} e_1 \cdots \xrightarrow{p_k} e_k$. They allow a generalization of both predicates and entities by the use of a wildcard (`*`). In the context of Description Logics, this generalization corresponds to generalizing predicates with the universal predicate U and entities with the top level class $\top$. In their study, they focus on paths of the form $\texttt{root} \xrightarrow{*} * \cdots \xrightarrow{*} e$, which is somewhat equivalent to extracting neighbors and their distance from entities to classify. These works involving paths and path patterns (or generalized paths) motivated the contributions presented in Chapter 6.

Besides refinement approaches, Paulheim [132] also categorizes the evaluation methodologies of approaches. He identifies the three following paradigms: "knowledge graph as silver standard", "partial gold standard", and "ex post evaluation". First, the "knowledge graph as silver standard" paradigm is well adapted to evaluate completion methods. Indeed, this paradigm considers the knowledge graph as correct and methods are evaluated with regard to how well triples in the knowledge graph can be replicated. This paradigm cannot be applied on error detection methods since the knowledge graph is considered correct. However, this evaluation methodology raises the issue of evaluating a predicted triple that is absent from the original knowledge graph.

Indeed, as detailed in Subsection 2.1.3, such a triple may not be a false positive under the Open World Assumption. Additionally, the assumption that the knowledge graph is correct may not always hold, which can lead to the replication of incorrect triples to complete the knowledge graph. Second, a "partial gold standard" can be available, *i.e.*, a subset of entities and triples that have been selected and labeled manually. Triples are labeled as "correct" or "incorrect" for error detection tasks or "should be in the knowledge graph" for completeness tasks. Such a manual labeling process leads to high quality data but is costly. Consequently, gold standards are usually of small size. Third, in "ex post evaluation", a human expert manually evaluates the output of an approach. Such an evaluation cannot be reused, contrary to gold standards, since only a one time output is evaluated.

Interestingly, Paulheim [132] also cites computational performance as an evaluation criterion of refinement approaches. Indeed, as knowledge graphs become larger, the scalability of approaches is of importance. This concern is shared by other authors such as Ji et al. [83]. To evaluate these performances, computational complexities and runtime measurement metrics (*e.g.*, memory consumption) are usually used. We consider such scalability issues when mining paths and path patterns from knowledge graphs in Chapter 6.

A focus on matching methods in ontologies and knowledge graphs

Knowledge graphs may contain similar or identical units (*e.g.*, entities, facts, classes, predicates) due to the concurrent nature of their publication and edition. Hence, an important task lies in *matching* them, *i.e.*, identifying similar units within and across knowledge graphs. This process is also called *interlinking* between knowledge graphs. It results in *alignments*, *i.e.*, correspondences between units. Different alignment relations can hold between two units. For example, a relation can identify identical units (*e.g.*, `owl:sameAs`), weakly similar units (*e.g.*, `skos:related`) or units that are more specific than others (*e.g.*, `skos:broadMatch`). The matching process can be seen as a particular refinement task. Indeed, resulting alignments can be represented as triples added to the knowledge graphs, thus improving their completeness. Alignments that involve units within the same knowledge graph can be used for duplicate removal, which improves the quality of the knowledge graph.

The matching problem is extensively studied in the research field of *ontology matching*. A formalization of the matching task and a detailed presentation of the main methods can be found in the book of Euzenat and Shvaiko [58][12]. We recall here some basics about this research field. Matching units is challenging since knowledge graphs can be heterogeneous. Euzenat and Shvaiko [58] identify the four most important types of heterogeneity:

Syntactic hereogeneity occurs when two knowledge graphs are not expressed with the same knowledge representation formalism. For example, a knowledge graph can be represented with Semantic Web technologies whereas another can be a property graph.

Terminological heterogeneity occurs when constituents of two knowledge graphs have different names while representing the same entities, classes, or predicates. Such a terminological issue arises when using different languages (*e.g.*, warfarin in English *vs.* coumaphène in French) or synonyms (*e.g.*, Article *vs.* Paper).

Conceptual heterogeneity occurs when knowledge graphs model differently the same domain of interest. Indeed, knowledge graphs can differ in:

[12]The field is called *ontology matching* but Euzenat and Shvaiko [58] consider that an *ontology* can contain instances, which corresponds to our definition of knowledge graph.

Coverage when knowledge graphs describe different parts of the same domain of interest, possibly overlapping. For instance, in our application, the different sources of pharmacogenomic knowledge are expected to have differences in coverage.

Granularity when knowledge graphs describe the same domain of interest but with different levels of detail. In our pharmacogenomic application, relationships described at the gene level or at the gene variant level differ in granularity.

Perspective when knowledge graphs describe the same domain of interest but from different perspectives. For example, the scope of the Genomic CDS ontology lies in clinical decision support based on pharmacogenomic knowledge. In Chapter 3, we design PGxO, an ontology whose scope is to represent n-ary pharmacogenomic relationships to integrate them from different sources. Hence, Genomic CDS and PGxO both model the pharmacogenomic domain but from different perspectives.

Semiotic heterogeneity occurs when humans differently interpret constituents of a knowledge graphs, for example depending on the context in which they are used.

In their book, Euzenat and Shvaiko [58] focus on terminological and conceptual heterogeneities. A wide variety of approaches have been developed to match units while coping with such heterogeneities. To this aim, such approaches use different techniques. That is why Euzenat and Shvaiko [58] propose the two classifications of matching approaches depicted in Figure 2.3. The upper classification is based on the granularity and the input interpretation of matching approaches. Indeed, approaches can work at the element-level, *i.e.*, consider constituents of a knowledge graph in isolation, or at the structure-level, *i.e.*, consider constituents and their relationships with others. Additionally, approaches can interpret their input information syntactically (*i.e.*, with regard to its sole structure), with the help of external resources, or by considering formal semantics possibly associated with the input. Alternatively, the lower classification is based on the kind of input considered by matching approaches. Indeed, terminological approaches use strings or linguistic objects. Structural approaches rely on the structure of units. This structure is internal when only considering data values associated with units (*e.g.*, strings such as names, integers such as ages), or relational when considering the relationships holding between units. Extensional approaches rely on the ABox, that is to say instances of classes, or assertions of predicates. Semantic approaches consider the semantic interpretation of the knowledge graph, for example by using a reasoner.

Matching approaches can target different constituents of knowledge graphs, such as classes, predicates, or instances. In our work, we aim at matching pharmacogenomic relationships represented within a knowledge graph. Due to the adopted representation for such relationships (see Chapter 3), we are particularly interested in instance matching, which has several other names: record linkage, object identification, deduplication, entity alignment [124], entity resolution [168], or identity link prediction [79]. Usually, approaches in instance matching result in equivalence axioms between individuals. As previously mentioned, methods and tasks are not disjoint. Hence, such a matching can be seen as a link prediction task targeting `owl:sameAs` links. Alternatively, it can also be seen as a node clustering task [27]. In our contributions, we propose other alignment relations in addition to equivalences between individuals (see Chapter 4). We consider this task both in a link prediction perspective (see Chapter 4) and in a node clustering perspective (see Chapter 5). Due to the representation of pharmacogenomic relationships (see Chapter 3), our approaches are inherently relational. We rely on existing alignments, predicate assertions, graph structure, taxonomies, and other semantics to tackle heterogeneity issues.

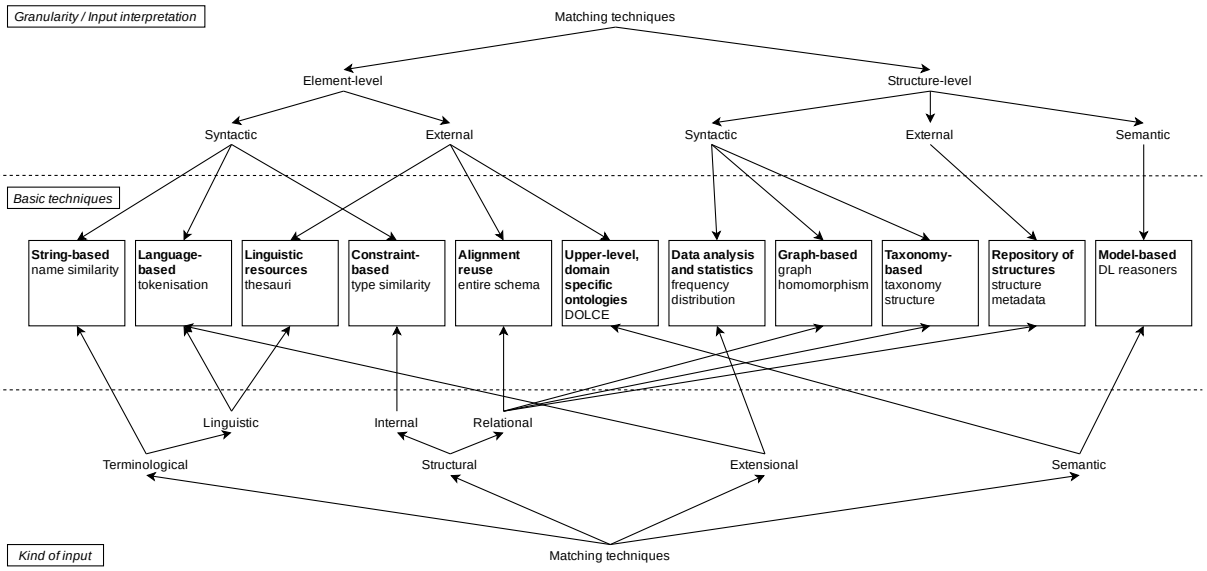


Figure 2.3: Classifications of elementary matching techniques proposed by Euzenat and Shvaiko [58]. Classes of basic techniques are depicted in the middle layer. A matching technique can be classified based on its granularity and how it interprets the input information (at the top), or based on the kind of input it uses (at the bottom).

Among existing matching approaches, some works are of particular interest and inspired our contributions. For example, Suchanek et al. [158] propose PARIS, a holistic method to align individuals, classes, and predicates. In this framework, alignments for each type of unit fertilize the others. PARIS is based on probabilistic rules that rely on the internal and relational structures of units and the functionality of predicates. Instead of being defined by authors, matching rules can also be mined from the knowledge graph. For example, Galárraga et al. [65] use their system AMIE [66] to mine specific rules to align knowledge graphs. Accordingly, their alignment process is performed under the *Partial Completeness Assumption*, which also considers the functionality of predicates. Alternatively, Atencia et al. [8] define *linkkeys*, a structure-based method to align individuals. A linkkey consists of a pair of classes (C_1, C_2) and a set of pairs of properties $\{\langle p_1, q_1\rangle, \ldots, \langle p_n, q_n\rangle\}$ from two knowledge graphs. An instance i_1 of C_1 is regarded as identical to an instance i_2 of C_2 whenever for each pair $\langle p_k, q_k\rangle$ there exist values or entities v such that $p_k(i_1, v)$ and $q_k(i_1, v)$, *i.e.*, there exist shared values or entities.

Other tasks can be considered similar to matching, such as reconciling knowledge sources, *i.e.*, harmonizing their content to merge them or make them independent [58]. Indeed, the reconciliation process inherently requires to identify similar entities. For example, in the domain of Natural Language Processing, Mongiovì et al. [110] extract knowledge from multiple natural language sources and merge them into a unique representation. Such a reconciliation can be performed by relying on the taxonomical structure, or by using machine learning techniques that learn vector representations of units such that similar units have close vectors, for example in terms of distances [6]. Some of these machine learning techniques are further described in Subsection 2.3.2.

2.2.2 Mining of knowledge graphs

We previously introduced the process of Knowledge Discovery in Databases (KDD) that aims at discovering new and useful knowledge from data [62]. Recall that, to simplify, we use the term *mining* to denote the whole KDD process. This process is inherently interactive and iterative. Indeed, it is guided by the knowledge and objectives of an analyst, and newly discovered knowledge can be interpreted by the analyst to guide the next iteration of the process.

Knowledge representation formalisms allow software agents to consume and interpret represented knowledge. Hence, some research works propose to use such formalized knowledge in the KDD process besides the knowledge of the analyst, leading to Knowledge Discovery guided by Domain Knowledge (KDDK) [100]. In this view, knowledge graphs could guide the KDDK process since they contain such formalized knowledge. Similarly, the survey of Ristoski and Paulheim [143] highlights the importance of knowledge graphs in the KDD process. Particularly, the authors identify the three following broad categories for mining approaches that involve Semantic Web knowledge graphs:

- Using Semantic Web knowledge graphs to support the KDD process;
- Using data mining techniques to mine Semantic Web knowledge graphs, which is also called *Semantic Web mining*;
- Using data mining and machine learning techniques to create and improve Semantic Web knowledge graphs.

It should be noted that these categories are not necessarily disjoint. Indeed, it is possible to use knowledge graphs to support a KDD process (first category) whose goal is to improve a given knowledge graph (third category). Since we already detailed the third category in Subsection 2.2.1, we focus here on the first and second categories.

Using Semantic Web knowledge graphs to support the KDD process

Ristoski and Paulheim [143] show that all steps of the KDD process have been associated with knowledge graphs in various approaches. Hence, they propose the KDD process enriched with knowledge graphs, as depicted in Figure 2.4. In this view, given a data set to mine, a first major step resides in its *linking* with one or several knowledge graphs. These knowledge graphs are chosen by the analyst, *e.g.*, because their knowledge is related to the data set and the considered mining task. This linking can be performed at any step of the KDD process but is usually achieved in the beginning. Linked knowledge graphs then support all steps of the KDD process. When *selecting* data, knowledge graphs combined with data visualization and summarization techniques can help an analyst understand the data set and select the appropriate target data. Ontologies and knowledge graphs allow to *consolidate* and *cleanse* the data in the *preprocessing* step to increase data quality. Indeed, constraints expressed in ontologies enable the detection of outliers and false values. Missing values can be filled by being retrieved from the linked knowledge graphs. They can also be inferred with reasoning mechanisms. Synonyms represented in knowledge graphs enable to normalize the target data.

Knowledge graphs can provide additional features to enrich the mined data set. However, mining algorithms may not be suited for graph data. Thus, knowledge graphs should be *transformed* to create features, a process sometimes called *propositionalization*. For example, de Vries and de Rooij [48] propose a graph kernel that counts common substructures in RDF graphs (*e.g.*, walks, subtrees) [47, 48]. FeGeLOD [134] transforms graph data into various features such as

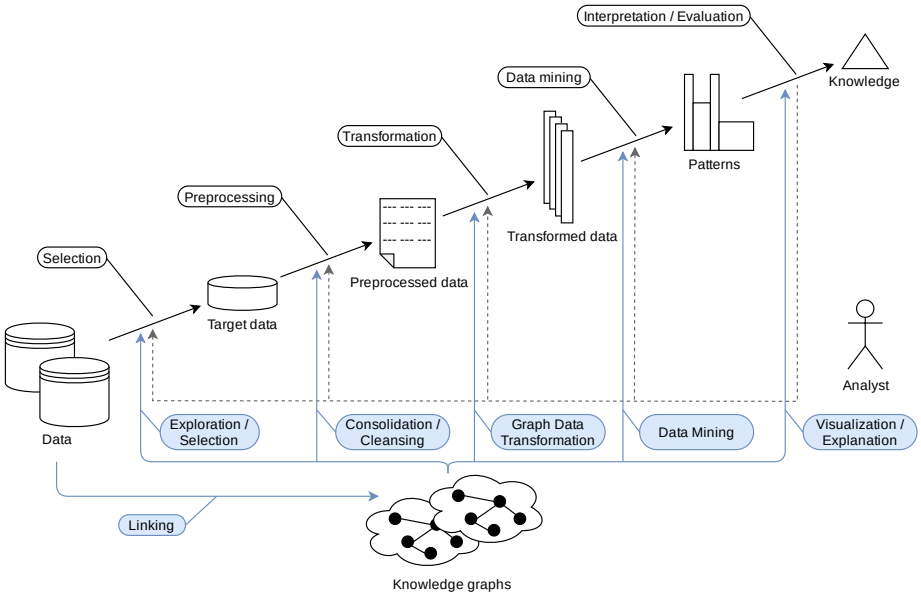


Figure 2.4: Enrichment of the KDD process (see Figure 1.4) with knowledge graphs, as proposed by Ristoski and Paulheim [143].

relations $\xrightarrow{p} e$ and *qualified relations* $\xrightarrow{p} t$. A relation $\xrightarrow{p} e$ is associated with all individuals of a knowledge graph that are linked to the individual e through the predicate p. Similarly, a qualified relation $\xrightarrow{p} t$ is associated with all individuals of a knowledge graph that are linked through the predicate p to an individual that instantiates the class t. These relations and qualified relations are similar to the paths and path patterns that we mine in Chapter 6. In the *mining* step, ontologies can be used to define mining workflows (*e.g.*, the succession of mining algorithms) and check their consistency (*e.g.*, the output of an algorithm is accepted as input of the next algorithm). Alternatively, they can be used to focus on some patterns of interest. For example, in Chapter 6, we use ontologies to retain only the most specific path patterns with regard to the hierarchy of classes. Indeed, we believe these patterns are the most descriptive, and thus the most useful to an analyst.

Knowledge graphs are interpretable both by human and software agents. That is why features mined from knowledge graphs are particularly interesting in the *interpretation* step, in which they can be used to *explain* results. For example, Explain-a-LOD [131] enriches statistical data sets with features from DBpedia. When correlations can be established between statistics and DBpedia features, these features can be used as explanations for the original statistics. To illustrate, the quality of living in cities has been correlated with whether these cities are European capitals. Interestingly, Explain-a-LOD leverages the different outputs of FeGeLOD [134]. Ontology classes can also be used to increase the interpretability of *association rules* [3]. Consider a set of objects G and a set of attributes M. Each object $o \in G$ can be associated with some attributes $m \in M$. Associations rules have the form $A \rightarrow B$, where $A, B \subseteq M$, and can be read as "objects having all attributes in A will also have all attributes in B". A rule may only be valid for some objects in G, which is measured by its *support* and *confidence*:

$$\text{support}(A \rightarrow B) = \frac{|\{o \mid o \in G \text{ and o is associated with all attributes in } A \cup B\}|}{|G|}$$

$$\text{confidence}(A \rightarrow B) = \frac{|\{o \mid o \in G \text{ and o is associated with all attributes in } A \cup B\}|}{|\{o \mid o \in G \text{ and o is associated with all attributes in } A\}|}$$

If attributes in M instantiate classes of an ontology, it is possible to generate *generalized association rules* [155], which is similar to the concept of *raising* [177]. This process replaces attributes with their instantiated ontology classes, up to a certain level in the ontology hierarchy. Resulting generalized rules may be more interpretable than the original sets of attributes. Additionally, raising can be performed to increase the support while preserving a high confidence. Other approaches also investigate an iterative generalization of rules and prune redundant ones at each iteration [52]. Generalized rules can be considered somewhat similar to the path patterns we mine in Chapter 6, and our focus on the most specific patterns also aims at reducing their redundancy.

Using data mining techniques to mine Semantic Web knowledge graphs

We saw that knowledge graphs can support all steps of the KDD process and, particularly, that they can be linked to mined data sets to enrich them. However, knowledge graphs themselves are also interesting targets to mine. Hence, numerous approaches have been proposed for such a *Semantic Web mining*. These approaches have various applications. For example, recommender systems benefit from the many useful features about products or services that can be mined from knowledge graphs [143, 168]. Alternatively, fact-checking approaches can use knowledge graphs to check the validity of a fact. In this view, Shi and Weninger [150] model a fact as a triple $\langle s, p, t \rangle$. They check whether this triple is true by trying to predict it from a set of learned *discriminative paths* $\mathbf{o}_s \xrightarrow{p_1} \xrightarrow{p_2} \cdots \xrightarrow{p_{k-1}} \xrightarrow{p_k} \mathbf{o}_t$, where $\mathbf{o}_s$ and $\mathbf{o}_t$ are respectively the set of classes instantiated by s and t.

Mining knowledge graphs can also be useful in the biomedical domain. Hence, Odgers and Dumontier [128] propose to transform Electronic Health Records of the STRIDE Clinical Data Warehouse into a knowledge graph. STRIDE is a database that contains EHR data from the Lucile Packard Children's Hospital and Stanford Hospital and Clinics. The resulting knowledge graph is then linked to those from the Bio2RDF project [54]. This interlinking enables federated queries across knowledge graphs that can be used to answer various medical questions of interest. For example, it is possible to retrieve the adverse events experienced by patients suffering from a specified disease and treated by a specified drug. Because of the interlinking, the specification of the disease and the drug can rely on external terminologies and knowledge graphs. Dalleau et al. [44] assemble various Linked Open Data sets related to pharmacogenomics and propose to identify pharmacogenes, *i.e.*, genes that may be involved in variability in drug responses. They view this task as classifying genes either as pharmacogenes or not. Interestingly, they use the graph kernel proposed by de Vries and de Rooij [48] to transform their knowledge graph into features. Such features are given to a Support Vector Machine that performs the classification. Similarly, Kamdar and Musen [85] use federated SPARQL queries across several knowledge graphs but they use them to build a hidden conditional random field (HCRF), *i.e.*, a discriminative undirected probabilistic graphical model. Because this model is a graph that results from querying several knowledge graphs, it could be seen as a "simplified" knowledge graph. Such a model predicts the probabilities of *outcomes* given *inputs*. Here, the aim is to predict adverse events that may occur given some drug treatments. Interestingly, all the previous works consider several knowledge graphs together. For example, all the links of interest to build the HCRF may appear in different knowledge graphs [85]. We also consider such a combined use in our work, particularly in Chapter 5 and Chapter 6.

	apple pie	waffle	ice cream	pastry	dessert
alice	×			×	×
bob		×		×	×
carl			×		×
denise			×		×
eric	×	×		×	×

Table 2.1: Example of a formal context that associates objects (here, persons) and their attributes (here, the food they like).

It should be noted that the knowledge discovered by mining knowledge graphs can, in turn, be represented within the knowledge graphs that were mined. Recall that the mining process is iterative. Hence, in this view, discovered knowledge can then support the next iteration of the KDDK process.

2.3 Two frameworks to refine and mine knowledge graphs

We saw that many different approaches have been proposed to refine and mine knowledge graphs, illustrating the variety of possible frameworks used for these tasks. In our work, we focus on the two specific frameworks of Formal Concept Analysis (FCA) and knowledge graph embedding, which we further detail below.

2.3.1 Formal Concept Analysis and its extensions

Basics about Formal Concept Analysis

Formal Concept Analysis (FCA) [70] is a mathematical framework that groups objects with regard to their common attributes. Such a grouping highlights regularities in data sets and is well adapted to various tasks such as data analysis, classification, knowledge discovery, and knowledge engineering [12, 33, 151]. In the following, we present the basics of FCA, following the notations used by Ganter and Wille [70].

FCA takes as input a *formal context* which is a binary table defined as follows:

Definition 6 (Formal context)**.** A *formal context* is a triple (G, M, I), where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is the incidence relation. Given an object $g \in G$ and an attribute $m \in M$, gIm indicates that the object g has the attribute m.

Such a formal context can be represented as a binary table, as illustrated in Table 2.1 where persons are associated with the food they like.

Remark 1. It should be noted that "objects" in FCA are not restricted to the "objects" of RDF triples previously presented. When needed to avoid ambiguity, we use "formal objects" to differentiate objects used in FCA from objects of RDF triples.

Given a set of objects, it is possible to obtain the set of their shared attributes by applying the derivation operator denoted by $(\cdot)' : 2^G \to 2^M$. Conversely, given a set of attributes, it is possible to obtain the set of their shared objects by applying the derivation operator $(\cdot)' : 2^M \to 2^G$. In

both cases, 2^G and 2^M respectively denote the powerset of G and the powerset of M. Formally, given a set of objects $A \subseteq G$ and a set of attributes $B \subseteq M$,

$$A' = \{m \in M \mid gIm \ \forall g \in A\}$$

$$B' = \{g \in G \mid gIm \ \forall m \in M\}$$

A' is the set of attributes shared by all the objects in A and B' is the set of objects having all the attributes in B. These two derivation operators constitute a *Galois connection* between 2^G and 2^M. Their compositions (*i.e.*, $(\cdot)'' : 2^G \to 2^G$ and $(\cdot)'' : 2^M \to 2^M$) are *closure operators*, *i.e.*, monotonous ($X \subseteq Y \Rightarrow X'' \subseteq Y''$), extensive ($X \subseteq X''$), and idempotent ($X'' = X''''$) operators.

FCA computes *formal concepts* from the formal context and these derivation operators. These concepts are defined as follows:

Definition 7 (Formal concept)**.** A pair (A, B), where $A \subseteq G$ and $B \subseteq M$, is a *formal concept* if and only if $A' = B$ and $B' = A$. A is called the extent of the concept and B is called the intent of the concept.

Given a formal concept (A, B), A is the maximal set of objects associated with the maximal set of attributes B. That is to say, no other attributes than those in B are shared by all objects in A. Conversely, no other objects than those in A have all attributes in B.

Considering two formal concepts (A_1, B_1) and (A_2, B_2), they are partially ordered as follows:

$$(A_1, B_1) \leqslant (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 \text{ (or dually } B_2 \subseteq B_1)$$

(A_1, B_1) is a *subconcept* of (A_2, B_2) and (A_2, B_2) is a *superconcept* of (A_1, B_1). The set of all formal concepts with the partial order $\leqslant$ form a *concept lattice* that can be represented using a Hasse Diagram. For example, the concept lattice resulting from the formal context in Table 2.1 is represented in Figure 2.5. Usually, concepts lattice are depicted using the *reduced notation* Indeed, considering two formal concepts $(A_1, B_1) \leqslant (A_2, B_2)$, because $A_1 \subseteq A_2$, A_2 can be depicted showing only the new objects that are not already in A_1. Similarly, because $B_2 \subseteq B_1$, B_1 can be depicted showing only the new attributes that are not already in B_2. The concept lattice in reduced notation resulting from the formal context in Table 2.1 is represented in Figure 2.6.

We mentioned that FCA and lattices can be used in various knowledge discovery tasks. To illustrate, it is possible to directly read rules from the lattice in Figure 2.6. Consider the orange edge. It can be read *bottom-up* as the implication waffle $\Rightarrow$ pastry, *i.e.*, all persons that like waffle also like (some) pastry. An implication is an association rule with a confidence of 1. When read *top-down*, it is possible to extract the association rule pastry $\rightarrow$ waffle, which states that persons liking pastry *may* also like waffle. This association rule has a support of $\frac{2}{5}$ and a confidence of $\frac{2}{3}$.

Pattern structures and other extensions

FCA is well suited for binary contexts. However, data mining tasks frequently involve other types of data, such as numerical values, qualitative attributes, or complex descriptions. That is why several extensions of FCA were introduced. For example, it is possible to handle numerical values with *conceptual scaling*. Let us consider the formal context in Table 2.1 but with preference ratings between 0 and 5 instead of binary preferences. Such a context could be scaled into a

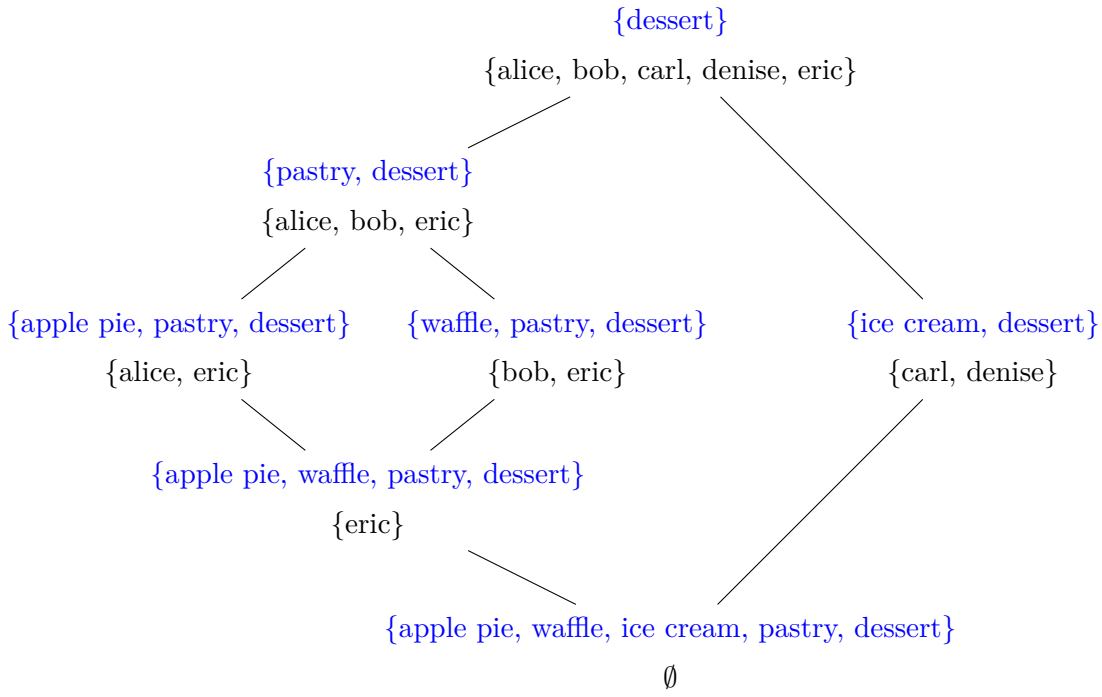


Figure 2.5: Hasse diagram representing the concept lattice built from the formal context in Table 2.1. Objects are depicted in black and attributes are depicted in blue.

binary context by transforming values into binary attributes such as applie pie ≤ 1, applie pie ≥ 1, applie pie ≤ 2, applie pie ≥ 2, etc. A similar scaling could be adopted for qualitative attributes by considering each possible value as a binary feature.

When objects are associated with complex descriptions that are ordered, it is possible to use a *pattern structure*.

Definition 8 (Pattern structure)**.** A pattern structure is a triple $(G, (D, \sqcap), \delta)$ where:

- G is a set of objects.
- $(D, \sqcap)$ is a *meet semi-lattice* of descriptions, *i.e.*, D is a set of complex descriptions (or *patterns*) that are organized hierarchically by a subsumption relation $\sqsubseteq$ such that it is always possible to compute the meet of two descriptions by applying the operator $\sqcap$. Given two descriptions $d_1, d_2 \in D$, $d_1 \sqcap d_2$ always exists and can be seen as the "similarity" between d_1 and d_2.
- $\delta : G \to D$ is a mapping that associates an object $g \in G$ with its description $d \in D$.

Given a set of objects $A \subseteq G$ and a description $d \in D$, the derivation operators $(\cdot)^{\square} : 2^G \to D$ and $(\cdot)^{\square} : D \to 2^G$ are defined as follows:

$$A^{\square} = \bigsqcap_{g \in A} \delta(g)$$

$$d^{\square} = \{g \in G \mid d \sqsubseteq \delta(g)\}$$

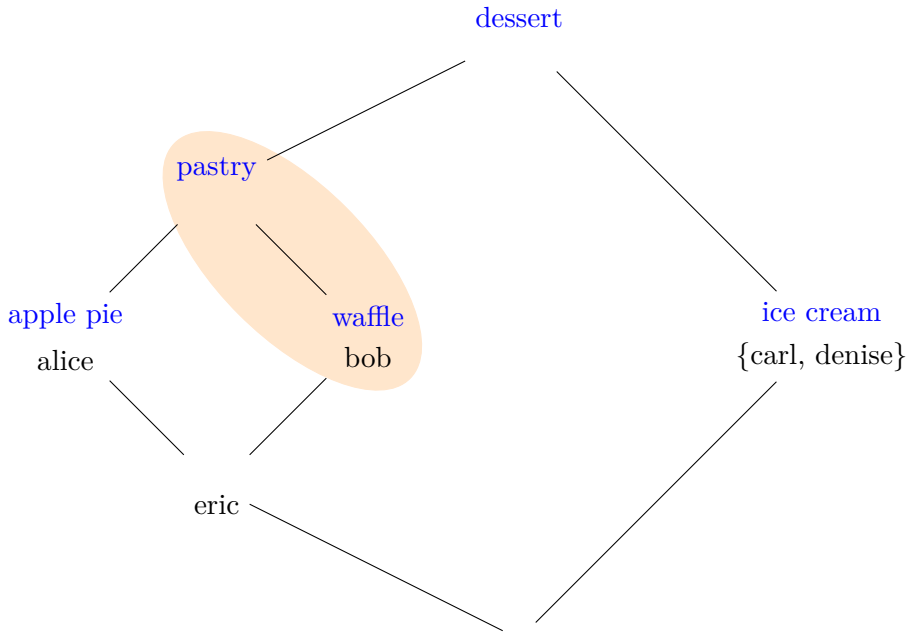


Figure 2.6: Hasse diagram representing the concept lattice built from the formal context in Table 2.1. The lattice is displayed using the reduced notation. Following this notation, objects, depicted in black, are associated with one concept and, implicitly, with all its superconcepts. Attributes, in blue, are associated with one concept and, implicitly, with all its subconcepts. Rules can be directly read from such a lattice. For example, the orange edge can be read *bottom-up* as the implication waffle $\Rightarrow$ pastry, or *top-down* as the association rule pastry $\rightarrow$ waffle whose support is $\frac{2}{5}$ and whose confidence is $\frac{2}{3}$.

$A^\square$ is the minimal description shared by all objects in A, and $d^\square$ is the set of all objects whose description is larger than d. These derivation operators enable to compute *pattern concepts*, defined as follows:

Definition 9 (Pattern concept)**.** A pair (A, d), where $A \subseteq G$ and $d \in D$, is a *pattern concept* if and only if $A^\square = d$ and $d^\square = A$.

Considering two pattern concepts (A_1, d_1) and (A_2, d_2), there are partially ordered as follows:

$$(A_1, d_1) \leqslant (A_2, d_2) \Leftrightarrow A_1 \subseteq A_2 \text{ (or dually } d_2 \sqsubseteq d_1)$$

This partial ordering allows to organize pattern concepts in a lattice, similarly to standard binary FCA.

Pattern structures can be used on various types of data sets. For example, Kaytoue et al. [89] propose a pattern structure to deal with numerical values. Indeed, a numerical value can be seen as an interval. For example, 4 can be seen as the interval $[4, 4]$. Given two descriptions $\delta(g_1) = [l_1, r_1]$ and $\delta(g_2) = [l_2, r_2]$, their meet description can be computed as follows:

$$\delta(g_1) \sqcap \delta(g_2) = [\min(l_1, l_2), \max(r_1, r_2)]$$

Interestingly, standard binary FCA can be seen as a pattern structure where the set of descriptions D is the powerset of M (2^M), which is hierarchically ordered by set inclusion $(\subseteq)$. An intuitive analogy can also be made between the hierarchy of descriptions and the hierarchy of ontology classes, which can be seen as descriptions of their instances. Hence, in Chapter 7, we use pattern structures with this class hierarchy to refine ontologies.

Remark 2. In the formalism of pattern structures, $d_1 \sqsubseteq d_2$ means that d_1 is a description that is associated with more objects than d_2. On the contrary, in Description Logics, $C_1 \sqsubseteq C_2$ means that C_2 is a class that is instantiated by more individuals than C_1.

Other extensions of FCA have also been introduced. For example, Relational Concept Analysis (RCA) [77] considers multiple contexts whose object sets are in relation. Contexts are then iteratively enriched with *relational attributes*. These attributes indicate the objects of a context that are in relation with concepts that result from another context. Various quantifiers can be used to build these relational attributes. Indeed, attributes can indicate that objects are in relation with at least one object in the extent of the other concept ($\exists$ quantifier) or all objects ($\forall$ quantifier). Interestingly, Codocedo and Napoli [35] propose to conjointly use heterogeneous contexts involving attributes that can be relational, binary, or multi-valued. For such a conjoint use, they propose a framework called "heterogeneous pattern structure". Alternatively, it is also possible to consider contexts with more dimensions than the two considered by FCA. Hence, Triadic Concept Analysis [98] adds a third dimension called "conditions".

Some works using FCA to refine and mine knowledge graphs

Numerous works use FCA or its extensions to refine and mine knowledge graphs. For example, Alam et al. [5] propose to refine DBpedia by mining definitions for the DBpedia categories using an heterogeneous pattern structure. Objects of the context are pages belonging to these categories. Attributes can be binary and indicate the category of the page or the presence of a triple (*e.g.*, pages that are subject of a triple $\langle ?, \texttt{dbp:manufacturer}, \texttt{dbp:Lamborghini} \rangle$). They can also be numerical values (*e.g.*, the production year of a car) that are handled as intervals. Recall that rules and implications can be derived from a lattice. Hence, in this work,

implications $X \Rightarrow Y$ are derived and proposed as definitions $X \Leftrightarrow Y$ when the rule $Y \rightarrow X$ has a high confidence. Shi et al. [151] also consider the task of refining semantic wikis but use RCA instead. They have various refinement objectives. To illustrate, from the resulting lattices, they identify equivalent classes when they appear in the same intent, and add subsumption axioms between classes in the intent of a concept and classes in the intent of its superconcepts.

Some papers use FCA to refine and mine knowledge graphs with applications in the biomedical domain. For example, Coulet et al. [36] consider drug-related documents from the DrugBank database that are annotated by (or associated with) ontology classes. However, such annotations may not be complete. Hence, Coulet et al. [36] propose to complete them based on a pattern lattice built from a pattern structure. The context is formed by documents as objects and the classes annotating them as attributes. The descriptions consist of the ontology classes that are ordered by subsumption. Since DrugBank is also a knowledge graph, this work can be seen as a completion approach. We mentioned in Chapter 1 that knowledge graphs can benefit various applications such as query answering or search result enhancement. To illustrate, Curé et al. [40] propose to define health outcomes of interest (HOIs) with a semantical expansion of seminal terms given by an expert in plain text (*e.g.*, "myocardial infarction"). HOIs can be used to select corresponding patients, for example for drug safety surveillance. To this aim, the seminal terms are matched with different classes of different ontologies, which in turn are found in the annotations of Electronic Health Records of hospitals (EHRs). However, the ontology classes matching the given terms may not allow to retrieve all relevant EHRs because of heterogeneity in annotated texts of EHRs. Hence, seminal ontology classes are expanded with their superclasses, which increases coverage but may reduce relevance. To tackle this issue, Curé et al. [40] build a binary context with the seminal classes as objects and all their superclasses as attributes. The resulting lattice is a compact representation of the ontology hierarchy that enables an efficient pruning of irrelevant classes. Indeed, consider a concept $(\{g_1, \ldots, g_m\}, \{a_1, \ldots, a_n\})$. Each a_j is a superclass of all seminal classes g_i. From such a concept and the ontology hierarchy, it is possible to quickly compute the ratio of common subclasses between a_j and all g_i. If this ratio is below a threshold, *i.e.*, if the consideration of a_j will decrease relevance, then a_j is pruned from the semantic expansion.

2.3.2 Knowledge graph embedding

Basics about graph embedding and some models

The objective of graph embedding is to convert a graph into a low dimensional space in which graph structural information and graph properties are preserved as much as possible [27]. To illustrate, in Figure 2.7, a 2-dimensional embedding space is represented. Each node of the original graph is represented as a pair of coordinates (x, y). In this space, the translation vector from a country to its capital is preserved across countries. Indeed, the translation from `Sweden` to `Stockholm` is the same as from `France` to `Paris`. Such low dimensional spaces enable to efficiently carry out various tasks without suffering from the high computational and space costs associated with big graphs [27]. For example, it is possible to predict the triple $\langle$`France`, `capital`, `Paris`$\rangle$ by applying the translation between `Sweden` and `Stockholm` to `France`. Other properties can be preserved in an embedding space, such as a geometrical grouping of similar entities. In our example, this would correspond to having all countries grouped and all capitals (or cities) grouped elsewhere in the embedding space. Numerous graph embedding models have been proposed and successfully applied on various applications such as node classification, link prediction, or node clustering [27, 168]. We do not intend to provide

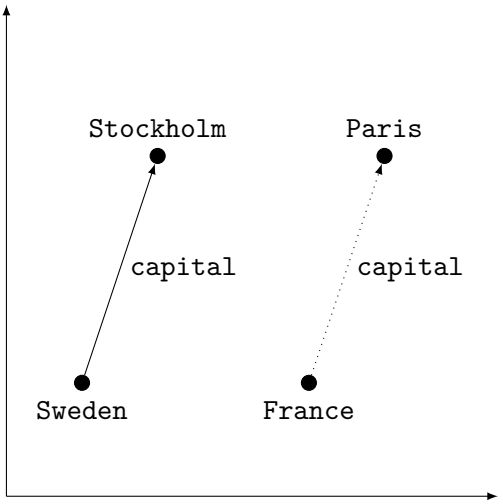


Figure 2.7: Example of an embedding space in which the translation representing the `capital` relation between a country and its capital is preserved, *i.e.*, the translation is the same for all countries. Hence, it is possible to complete the knowledge graph: if the capital of `France` is unknown, it can be retrieved by applying on `France` the translation from `Sweden` to `Stockholm`. Such an embedding space could result from TransE [23].

a full review but rather focus on some key categorizations and models. For further details, the interested reader could refer to the surveys of Cai et al. [27], Chami et al. [31], Nickel et al. [124], and Wang et al. [168].

Graph embedding approaches may differ in several criteria. Cai et al. [27] propose a taxonomy of problems depicted in Figure 2.8 that categorizes approaches based on their input and output. Hence, it is possible to consider different types of graphs, *e.g.*, homogeneous graphs, or heterogeneous graphs such as knowledge graphs. Additionally, embeddings can be learned for different substructures of the original graph such as nodes or edges. Hybrid embeddings involve nodes and edges, or nodes and communities. The whole graph can also be represented as a vector, for example to investigate the similarity between two or several graphs as wholes. Interestingly, auxiliary information such as texts or images can be attached to constituents of the graph. Such auxiliary information could be seen as encompassing literal values associated with knowledge graphs. Hence, one challenge of learning graph embeddings with knowledge graphs lies in considering such literals [71]. This information can be taken into account at the level of the input graph, as in the taxonomy of Cai et al. [27], or at the level of the embedding technique itself, as in the categorization of Ji et al. [83]. This latter categorization is depicted in Figure 2.9 and considers four main criteria: the type of representation space, the scoring function, the encoding model used and the consideration of auxiliary information. We introduce some of the main models below in view of this categorization.

Embeddings of entities and relations are frequently vectors, matrices, or tensors whose values are in $\mathbb{R}$, *i.e.*, the representation space is a point-wise Euclidean space. For example, Bordes et al. [24] propose structured embeddings. In this framework, the embedding e_i of an entity i is a real vector, *i.e.*, $e_i \in \mathbb{R}^k$, where k is the number of dimensions of the embedding space. A relation r is embedded into two matrices $R_r^s, R_r^t \in \mathbb{R}^{d \times d}$ to differentiate the "influence" of the source and the tail of a triple in the scoring function. Indeed, given a triple $\langle i, r, j \rangle$, structured

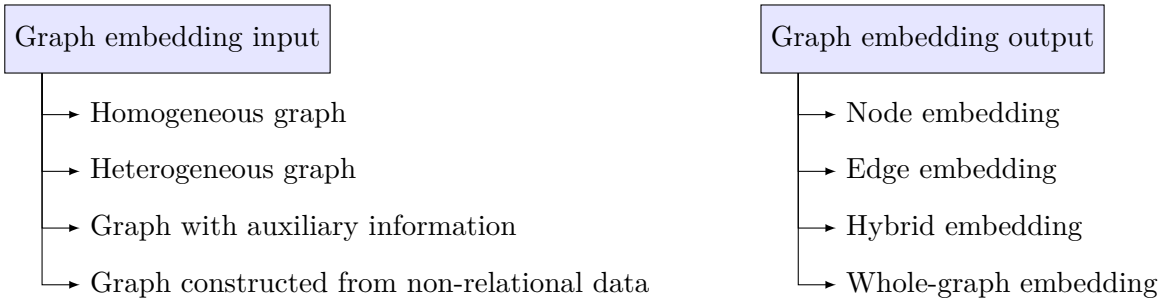


Figure 2.8: Taxonomy of graph embedding problem settings proposed by Cai et al. [27].

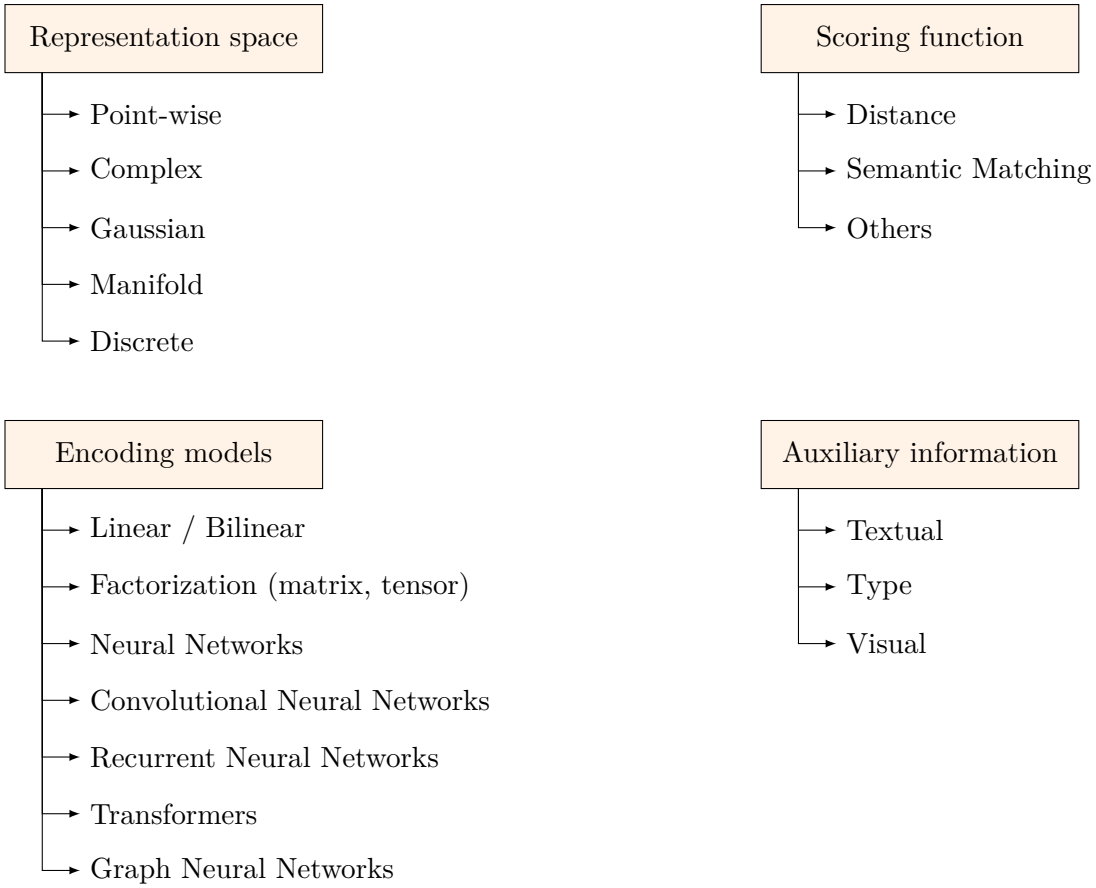


Figure 2.9: The four criteria and their categories proposed by Ji et al. [83] to classify knowledge representation learning approaches.

embeddings use the following linear and distance-based scoring function:

$$f(i,r,j) = ||R_r^s e_i - R_r^t e_j||_1$$

The objective lies in learning embeddings such that the score of a valid triple $\langle i,r,j\rangle$ is lower than the score of a invalid triple $\langle i',r,j'\rangle$[13]. Other approaches use similar linear and distance-based functions. Hence, TransE [23] is a translational approach that computes for each triple $\langle i,r,j\rangle$ of a knowledge graph, embeddings $e_i, e_r, e_j \in \mathbb{R}^k$ such that $e_i + e_r \approx e_j$, *i.e.*, the translation vector from the subject to the object of a triple corresponds to the embedding of the relation as depicted in Figure 2.7. Accordingly, the scoring function of a triple $\langle i,r,j\rangle$ is as follows:

$$f(i,r,j) = ||e_i + e_r - e_j||$$

This approach is adapted for link prediction but, according to the authors, it is unclear if it can adequately model relations of distinct arities, such as *1-to-Many*, or *Many-to-Many*. TransE has been used as a basis of several other translational models. For example, in TransH [169], the translation e_r associated with a relation is computed between the projection of e_i and e_j on a hyperplane specific to r represented by a normal vector w_r. In TransR [101], e_i and e_j are learned in an entity space (of dimension k) and the translation e_r (of dimension d) is applied in a relation space specific to r thanks to a mapping matrix $M_r \in \mathbb{R}^{k\times d}$.

All the aforementioned approaches are linear, use distance-based scoring functions, and learn real-valued embeddings. However, as depicted in Figure 2.9, other possibilities can be considered. Hence, ComplEx [161] uses complex-valued vectors for entities and relations, *i.e.*, $e_i, e_j, e_r \in \mathbb{C}^k$. Accordingly, its scoring function for a triple $\langle i,r,j\rangle$ can use operations such as the Hermitian dot product and complex conjugation:

$$f(i,r,j) = \mathrm{Re}\left(< e_r, e_i, \overline{e_j} >\right) = \mathrm{Re}\left(\sum_{l=1}^{k} e_{rl} e_{il} \overline{e_{jl}}\right)$$

Alternatively, DistMult learns real-valued vectors for entities and diagonal matrices M_r for relations. It adopts a semantic matching scoring function, *i.e.*, the plausibility of a fact is evaluated by a matrix multiplication transforming the head entity into the tail entity in the representation space [83]:

$$f(i,r,j) = e_i^T M_r e_j$$

Other approaches use random walks in the knowledge graph. For example, RDF2Vec [142] first extracts, for each node, a set of sequences of graph substructures starting from this node. Elements in these sequences can be edges, nodes, or subtrees. Then, sequences feed the word2vec model that compute embeddings for each element in a sequence by either maximizing the probability of an element given the other elements of the sequence (Continuous Bag of World architecture) or maximizing the probability of the other elements given the considered element (Skip-gram architecture). Interestingly, such random walks can be biased by weighting edges to traverse based on predicate frequency, object frequency, or pairs (predicate, object) frequency [34].

Embeddings can be learned using neural networks. For example, in Neural Tensor Networks [154], the scoring function computes a bilinear tensor product $e_i^T W_r^{[1:k]} e_j$[14] additionally

[13] More details about the learning process, invalid triples, and loss functions are given below.

[14] The bilinear tensor product results in a vector $h \in \mathbb{R}^k$ where each value h_i is computed on one slice of the tensor, *i.e.*, $h_i = e_i^T W_r^{[i]} e_j$.

to the standard form of a neural network (*i.e.*, the multiplication by the matrix V_r and the addition of the bias b_r). Then, a non-linear function is applied, here tanh:

$$f(i,r,j) = u_r^T \tanh\left(e_i^T W_r^{[1:k]} e_j + V_r \begin{bmatrix} e_i \\ e_j \end{bmatrix} + b_r\right)$$

Specific graph neural networks have also been developed. Hence, contrasting TransE and RDF2Vec that work at the triple and sequence levels, Graph Convolutional Networks (GCNs) compute the embedding of a node by considering its neighborhood in the graph. GCNs can be seen as a message-passing framework of multiple layers, in which the embedding $h_i^{(l+1)}$ of a node i at layer $(l+1)$ depends on the embeddings of its neighbors at level (l), as stated in the following convolution equation:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}} \sum_{j\in\mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)}\right)$$

where $\mathcal{N}_i^r$ is the set of neighboring nodes j accessible from i with the relation r, σ is a non-linear function such as ReLU or tanh, and $W^{(l)}$ are weight matrices. GCNs have been introduced for semi-supervised classification over graphs [92] and extended for entity classification and link prediction in knowledge graphs [146]. Intuitively, in GCNs, the embedding of a node i depends on the embeddings of its neighbors j, which makes them well-adapted to structure-based matching approaches, *i.e.*, approaches that match nodes by considering their relations with other nodes that can be individuals, classes, or data values. Indeed, in such approaches, nodes having similar neighborhoods are considered as similar. The convolution equation of GCNs entails that such nodes will have close embeddings, and thus can be efficiently identified in the embedding space. From this intuition shared with other papers (see below), we adopt GCNs to match individuals in Chapter 5 and further detail their framework there.

To train models and learn embeddings, various loss functions can be used [109]. The following loss function, sometimes called the margin-based ranking loss or Pairwise Hinge loss, is frequently adopted:

$$\mathcal{L}_{\text{PH}} = \sum_{\langle i,r,j\rangle\in S} \sum_{\langle i',r,j'\rangle\in S'_{\langle i,r,j\rangle}} \max\left(\gamma + f(i,r,j) - f(i',r,j'), 0\right)$$

For example, this loss is used to train TransE, TransH, and TransR. The objective is to minimize this loss by minimizing the score of valid triples $\langle i,r,j\rangle$ with regard to scores of invalid triples $\langle i',r,j'\rangle$ that should be maximized. The hyperparameter γ is a margin to separate valid triples from invalid ones. The set S of valid triples consists of triples that exist in the knowledge graph. For each valid triple $\langle i,r,j\rangle$, a set of invalid triples $S'_{\langle i,r,j\rangle}$ is formed by either corrupting the head or tail entity, *i.e.*, generating triples $\langle i',r,j\rangle$ or $\langle i,r,j'\rangle$ that do not exist in the knowledge graph. However, as discussed in Subsection 2.1.3, a challenge lies in generating corrupted triples that are not false negative, *i.e.*, triples that are actually valid and should be present in the knowledge graph. Additionally, another interrogation resides in the suitable number of corrupted triples to generate. It should be noted that DistMult and Neural Tensor Networks adopt a modified version of $\mathcal{L}_{\text{PH}}$, in which the signs of $f(i,r,j)$ and $f(i',r,j')$ are inverted, *i.e.*, the objective is to maximize the score of valid triples and minimize the score of invalid ones. In Chapter 5, we present the Soft Nearest Neighbor loss [63]. We adopt this loss with GCNs for a matching task that we view as a clustering task in the embedding space.

Towards semantic graph embeddings?

Graph embedding approaches have proven very powerful in various tasks related to knowledge graph refinement and mining. These performances can be partly explained since the continuous representation of embeddings is less strict than symbolic approaches, which may allow to better cope with heterogeneity [75]. However, recall that knowledge graphs can be represented within languages associated with formal semantics that provide reasoning mechanisms based on deduction. Hence, symbolic approaches and graph embedding have complementary strengths, and thus could be combined [75]. Additionally, the Semantic Web vision encompasses the interpretation of knowledge by both human and software agents. Embeddings do not provide such interpretability. Hence, Paulheim [133] advocates for *semantic embeddings*, *i.e.*, embeddings whose values can be interpreted.

In view of these observations, several works investigate the consideration of semantics, domain knowledge, and reasoning mechanisms in graph embedding approaches. For example, Guha [75] propose a model theory on vectors of reals, *i.e.*, embeddings. Logic Tensor Networks [149] learn groundings of logical terms and logical clauses. The grounding of a logical term consists in a vector of real numbers (*i.e.*, an embedding) and the grounding of a logical clause is a real number in the interval $[0, 1]$ (*i.e.*, the confidence in the truth of the clause). The learning process aims at minimizing the satisfiability error of a set of clauses, while ensuring the logical reasoning. This work can interestingly be compared to graph embedding if knowledge graphs are considered in their logical form, *i.e.*, considering nodes as logical terms and edges linking two nodes as logical formulae. Alternatively, Wang et al. [167] propose a hybrid attention mechanism, named "Logic Attention Network" (LAN), to use in embedding approaches for link prediction. LAN combines a mechanism based on logical rules and a neural network mechanism. When computing the embeddings of two nodes i and j to predict a triple $\langle i, r, j\rangle$, the rule-based mechanism weights their neighbors by promoting those linked through a predicate that has been found to strongly imply the predicate r of the link to predict. Such implications could be discovered from the hierarchy of predicates. Besides implications between predicates, more complex logical rules can be associated with knowledge graphs through ontologies. That is why Gutiérrez-Basulto and Schockaert [76] investigate how to ensure logical consistency through geometrical constraints on embedding spaces and if classical embedding techniques respect such constraints.

Some additional works using graph embedding to refine and mine knowledge graphs

Some of the presented models were successfully used for link prediction, entity classification, or node recommendation on knowledge graphs [27]. Interestingly, among the numerous available papers using graph embedding to refine and mine knowledge graphs, some already use GCNs for a matching task since they assume similar nodes have similar neighborhoods. For example, Wang et al. [170] propose to align cross-lingual knowledge graphs by using GCNs to learn node embeddings such that nodes representing the same entity in different languages have close embeddings. Pang et al. [130] use the same approach to align two KGs, but introduce an iterative aspect. Some newly-aligned entities are selected and used when learning embeddings in the next iteration. To avoid introducing false positive alignments, the newly-aligned entities are selected with a distance-based criteria proposed by the authors. Interestingly, the two previous works take into account literals during the embedding process and use the Pairwise Hinge loss, also used by TransE.

Other applications than link prediction or entity classification benefit from graph embedding frameworks. For example, Jurisch and Igler [84] consider two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ that are

already aligned. Given new versions of these ontologies $\mathcal{O}'_1$ and $\mathcal{O}'_2$, they use and evaluate various models (*i.e.*, TransE, TransH, ComplEx, DistMult, RDF2Vec, and GCNs) to automatically update existing alignments for these new versions. Hamilton et al. [78] focus on embedding conjunctive logical queries on knowledge graphs. Such a querying framework presents the advantage of answering queries even with missing intermediate entities. For example, consider two entities d_1 and d_2, and the following query where C is the query variable [78]:

$$C : \exists P,\ \texttt{assoc}(d_1, P) \wedge \texttt{assoc}(d_2, P) \wedge \texttt{target}(P, C)$$

Suppose the entity P is missing from the knowledge graph. In the embedding space, it is possible to carry out the geometric transformation associated with `assoc` on the embeddings of d_1 and d_2 to find an embedding that should be the one of the missing entity P. On this embedding, the transformation associated with `target` is applied to obtain the expected embedding of C. Then, the query result consists of entities whose embeddings are nearby (for example, by applying a nearest neighbor search). Their distance with the expected embedding of C can be interpreted as their likelihood to be the appropriate answer to the query. Besides coping with incompleteness in knowledge graphs, such an embedding for queries achieves a time complexity that is linear in the number of query variables, which is better than the exponential complexity obtained with a naive enumeration.

Inspired by these approaches using graph embedding to refine and mine knowledge graphs, in Chapter 5, we propose to use Graph Convolutional Networks to match individuals. Additionally, influenced by the papers considering the semantics associated with knowledge graphs in embedding approaches, we particularly investigate how *(i)* reasoning mechanisms can improve the performances in node matching with GCNs, and *(ii)* the distributions of distances in the embedding space can correspond to a "rediscovery" of the alignment relations.

Chapter 3

Representing and integrating pharmacogenomic knowledge

Contents

3.1 The need for a knowledge graph focused on integration 54
3.2 PGxO: a sufficient ontology for pharmacogenomics 54
3.2.1 Specification 55
3.2.2 Conception 55
3.2.3 Diffusion 57
3.2.4 Evaluation 57
3.3 PGxLOD: a knowledge graph for pharmacogenomics 58
3.3.1 Population with preexisting LOD 58
3.3.2 Population with PharmGKB data 59
3.3.3 Population with the biomedical literature 63
3.3.4 Population with Electronic Health Records and linked biobanks studies 67
3.3.5 Generation of mappings 68
3.4 Discussion and perspectives 69

In this chapter, we consider our application of knowledge management in pharmacogenomics (PGx)[15]. First, we motivate the need for a new knowledge graph to represent, integrate, trace, and reconcile pharmacogenomic relationships from various sources (Section 3.1). Accordingly, we developed a small ontology, PGxO (Section 3.2), and a knowledge graph, PGxLOD (Section 3.3), that instantiates PGxO with relationships from three sources: PharmGKB, the biomedical literature and results of studies that analyzed Electronic Health Records (EHRs) and linked DNA biobanks. By importing knowledge from these three sources, PGxLOD illustrates the structuring role of PGxO as well as constitutes a community resource for both pharmacogenomic and knowledge graph research. This chapter is based on an article published in *BMC Bioinformatics* in 2019 [115] that presented version 0.4 of PGxO and version 2 of PGxLOD. Results reported here are updated with regard to version 0.5 of PGxO and version 4 of PGxLOD. The first version of PGxO was described in an article published in the international workshop NETTAB 2017 [114].

[15] See Section 1.4 for more details.

3.1 The need for a knowledge graph focused on integration

In the application domain of this thesis, we want to reconcile pharmacogenomic knowledge of various origins: extracted from specialized databases (*e.g.*, PharmGKB), from the biomedical literature (*e.g.*, PubMed) or discovered by mining Electronic Health Records of hospitals. Recall that typical knowledge units in PGx are n-ary relationships between one (or more) *genetic factor(s)*, one (or more) *drug treatment(s)* and one (or more) *phenotype(s)*. Such a relationship states that the genetic factors impact the response to the drugs by causing the phenotypes. Such phenotypes can be the expected outcomes of drug treatments or some adverse effects. The abstract formalization of a pharmacogenomic relationship and examples are depicted in Figure 1.5 on page 18 and Figure 1.6 on page 18. The reconciliation of PGx relationships first requires their matching, which is challenging because of their n-ary aspect and their heterogeneity in terms of vocabularies, granularities, or languages as illustrated in Figure 1.8 on page 19. Therefore, there is a strong need for developing a common schema that would enable comparing in a knowledge graph the PGx knowledge extracted from databases and the literature or discovered from EHRs.

Several ontologies have already been developed for pharmacogenomics, but with different purposes, making them inadequate to the present need. In particular, SO-Pharm (Suggested Ontology for Pharmacogenomics) [39] and PO (Pharmacogenomic Ontology) [55] have been developed for knowledge discovery purposes rather than data integration or knowledge reconciliation. The PHARE ontology (PHArmacogenomic RElationships) [37] has been built for normalizing binary *gene – drug* and *gene – disease* relationships extracted from text. Consequently, PHARE is not suitable for representing n-ary pharmacogenomic relationships. More recently, Samwald et al. [145] introduced the Genomic CDS ontology (Pharmacogenomic Clinical Decision Support). It aims at proposing consistent information about pharmacogenomic patient testing to guide physician decisions in clinical practice.

Similarly to ontologies, several knowledge graphs focusing on Life Sciences already exist such as knowledge graphs from the Bio2RDF project [54] or DisGeNET-RDF [136]. Parts of PharmGKB data are already available in the form of LOD as an output of the Bio2RDF project [54]. Nonetheless, this version is outdated and provides only a small portion of PharmGKB. Even though these knowledge graphs don't focus on pharmacogenomics, they provide knowledge about components of pharmacogenomic relationships, *i.e.*, drugs, genetic factors, and phenotypes. Particularly, they provide cross-references between such components, indicating for example that a drug in a knowledge graph is also present with a different identifier in another knowledge graph. This is of particular interest in our purpose of reconciling relationships. Indeed, such cross-references may allow to match heterogeneously-represented relationships by identifying identical components from different sources.

We built PGxO and PGxLOD by learning and adapting from these previous experiments. Hence, for consistency reasons and good practices, PGxO classes are mapped with classes of aforementioned ontologies (*i.e.*, SO-Pharm, PO, PHARE, and Genomic CDS). Additionally, some existing knowledge graphs about Life Sciences are integrated inside PGxLOD. Hence, we connect the drugs, genetic factors, and phenotypes provided by these other knowledge graphs to represent PGx relationships in PGxLOD.

3.2 PGxO: a sufficient ontology for pharmacogenomics

PGxO is manually, collaboratively, and iteratively developed by 3 persons (Pierre Monnin, Clément Jonquet, and Adrien Coulet). We follow classical ontology constructions methods and

life cycle [51, 125], including steps of specification, conception, diffusion and evaluation.

3.2.1 Specification

Our aim is to represent and reconcile what we defined as pharmacogenomic knowledge units, *i.e.*, n-ary relationships between one (or more) *genetic factor(s)*, one (or more) *drug treatment(s)* and one (or more) *phenotype(s)*. In order to keep our ontology simple and leverage existing works representing pharmacogenomic components, we restrain the *scope* of PGxO only to representing pharmacogenomic knowledge units and not all facets of pharmacogenomics. The *objective* of PGxO is twofold: reconciling and tracing these pharmacogenomic knowledge units. To enable this reconciliation, we need to encode metadata and provenance information about a pharmacogenomic relationship.

3.2.2 Conception

Because PGxO is of small size, conception step was performed simultaneously with conceptualization, formalization and implementation steps. The ontology has been implemented in OWL using the Protégé ontology editor [122]. The expressive Description Logic associated with PGxO is $\mathcal{ALHI}(D)$ [9].

Representation of PGx knowledge

PGxO class hierarchy and object property hierarchy are depicted in Figure 3.1 and Figure 3.2 respectively. PGxO is composed of 11 classes, organized around the central class `PharmacogenomicRelationship`. Instances of the latter class represent atomic units of pharmacogenomic knowledge. Indeed, in the Semantic Web standards, only binary predicates exist. Thus, the proper representation of such n-ary relationships requires reifying them: relationships are individualized and linked to their components by predicates (see examples in Figures 3.3, 3.4, 3.5, and 3.6) [127].

Key components of pharmacogenomic relationships are instances of `Drug`, `GeneticFactor` and `Phenotype`. They are linked with the relationships they are involved in thanks to object properties `isAssociatedWith`, `isNotAssociatedWith`, and their descendants. These object properties specify the involvement of components in pharmacogenomic relationships. Apart from their association with pharmacogenomic relationships, drugs, genetic factors, and phenotypes can be directly associated together by two other object properties: `partOf` and `dependsOn`. The `partOf` property (or `ro:BFO_0000050`), from the Relation Ontology (RO) [153], is used to express that a genomic variation is located within the sequence of a specific gene. The `dependsOn` property (or `ro:RO_0002502`), also from RO, is used to express complex phenotypes involving other entities. Such complex phenotypes can be, for example, gene expressions (*e.g.*, *the expression of VKORC1*, depending on *VKORC1*) or drug response phenotypes (*e.g.*, *carbamazepine hypersensitivity*, depending on *carbamazepine*).

Mappings

For consistency reasons and good practices, we manually mapped classes of PGxO to aforementioned ontologies related to pharmacogenomics: SO-Pharm, PO, PHARE and Genomic CDS. This set of mappings is available on GitHub[16]. PGxO is published on the NCBO BioPortal (see

[16]Mappings from PGxO to SO-Pharm, PO, PHARE and Genomic CDS: `https://github.com/practikpharma/PGxO/blob/master/mappings/mapp1.owl`

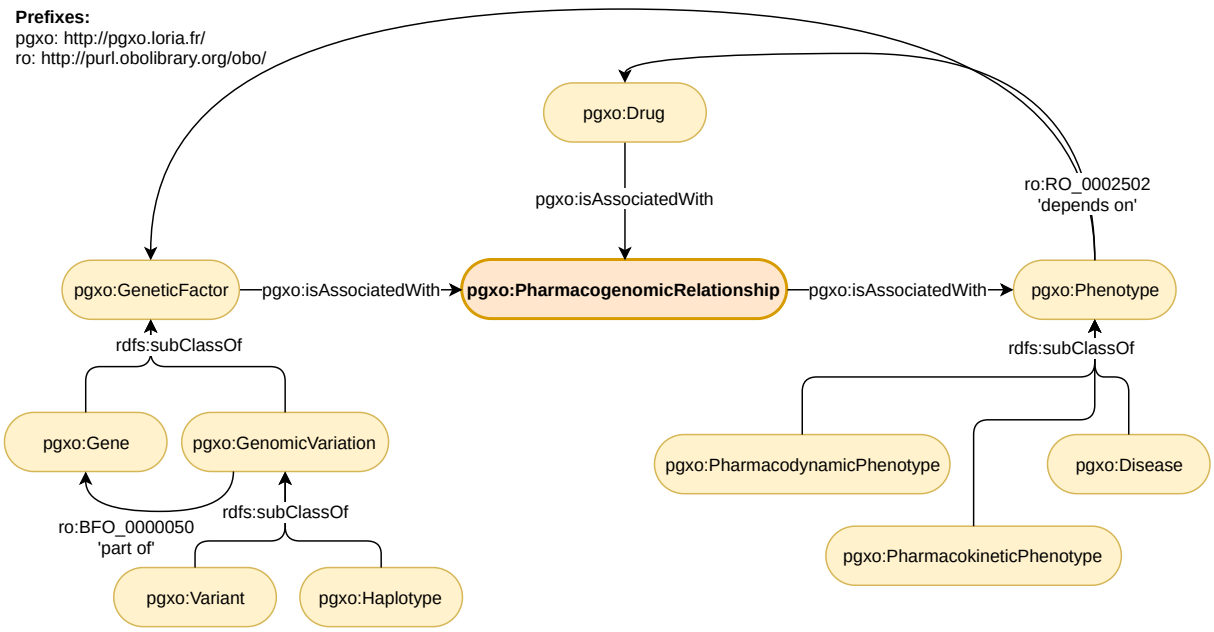


Figure 3.1: Main classes and object properties of PGxO. The central classes of the ontology is `PharmacogenomicRelationship`.

- `isAssociatedWith`
  - `influences`
    - `causes`
    - `decreases`
    - `decreasesChanceOf`
    - `increases`
    - `increasesChanceOf`
    - `metabolizes`
    - `transports`
  - `isInfluencedBy`
    - `hasChanceDecreasedBy`
    - `hasChanceIncreasedBy`
    - `isCausedBy`
    - `isDecreasedBy`
    - `isIncreasedBy`
    - `isMetabolizedBy`
    - `isTransportedBy`
  - `isTreatedBy`
  - `treats`

- `isNotAssociatedWith`
  - `doesNotInfluence`
    - `doesNotCause`
    - `doesNotDecrease`
    - `doesNotDecreaseChanceOf`
    - `doesNotIncrease`
    - `doesNotIncreaseChanceOf`
    - `doesNotMetabolize`
    - `doesNotTransport`
  - `doesNotTreat`
  - `isNotInfluencedBy`
    - `hasNotChanceDecreasedBy`
    - `hasNotChanceIncreasedBy`
    - `isNotCausedBy`
    - `isNotDecreasedBy`
    - `isNotIncreasedBy`
    - `isNotMetabolizedBy`
    - `isNotTransportedBy`
  - `isNotTreatedBy`

Figure 3.2: Hierarchy of object properties defined in PGxO to link drugs, genetic factors and phenotypes to instances of `PharmacogenomicRelationship`. `isAssociatedWith` and `isNotAssociatedWith` are symetric, *i.e.*, $\texttt{isAssociatedWith} \equiv \texttt{isAssociatedWith}^{-1}$ and $\texttt{isNotAssociatedWith} \equiv \texttt{isNotAssociatedWith}^{-1}$. Some relations are defined as inverses of others, *e.g.*, $\texttt{causes} \equiv \texttt{isCausedBy}^{-1}$. Negation is not explicitly defined in PGxO.

Subsection 3.2.3), which generates lexical-based mappings between hosted ontologies. Hence, it provides an initial set of mappings from PGxO to many standard ontologies. We manually completed BioPortal mappings between PGxO and three standard and broad spectrum ontologies: MeSH, NCIt and SNOMED CT. These mappings are also available on GitHub[17]. We choose to provide mappings as two independent OWL files to allow a flexible loading of the ontology, with or without mappings.

Provenance encoding

Data provenance (sometimes called lineage) refers to metadata stating where data came from, how they were derived, manipulated, and combined, and how they may have been updated [25]. With PGxO, we do not only want to represent units of knowledge of different origins, but also to trace their origins. For this purpose, we leverage an existing ontology for provenance, named PROV-O [96], which is a W3C Recommendation since 2013. In addition, for some particular provenance metadata, PGxO reuses object properties of the high-level ontology DUL (DOLCE+DnS Ultralite) [68].

PROV-O is built around three main classes: `Entity`, `Activity` and `Agent`. Entities represent things that can be generated, modified, etc. by activities. Activities are realized by agents that can be either human or software agents. Entities can also be directly attributed to agents.

In terms of PROV-O classes, authorities publishing sources from which we extract knowledge units are considered to be agents. Such authorities are, for instance, the PharmGKB team, the National Library of Medicine (NLM) in charge of PubMed, or an hospital in charge of a repository of EHRs. Data sources (*e.g.*, a version of PharmGKB, of PubMed, a repository of EHRs) are attributed to agents. They may then be used to derive data. These data, in turn, are used during the execution of an activity (*e.g.*, a mining algorithm). Such an execution generates entities that, in our case, are pharmacogenomic knowledge units. Quantitative and qualitative metadata may be associated to an activity and to the entities it generates. For instance, one can specify the version of an algorithm, the date of its execution, the quality of generated entities (*e.g.*, their levels of confidence, their p-value, etc.). Thereby, a further reconciliation of two knowledge units may take into account these various elements. Next section present examples of provenance metadata encoding.

3.2.3 Diffusion

All versions of PGxO are published online and shared with collaborators through both the NCBO BioPortal[18] [126] and GitHub[19]. PGxO is also described following the Minimum Information for the Reporting of an Ontology (MIRO) guidelines [103][20].

3.2.4 Evaluation

To evaluate our ontology, we used *competency questions* as proposed by Gangemi [67]. The questions we defined are the following:

[17]Mappings from PGxO to MeSH, NCIt and SNOMED CT: `https://github.com/practikpharma/PGxO/blob/master/mappings/mapp2.owl`

[18]PGxO summary page on the NCBO BioPortal: `https://bioportal.bioontology.org/ontologies/PGXO`

[19]PGxO repository on GitHub: `https://github.com/practikpharma/pgxo`

[20]MIRO description of PGxO: `https://github.com/practikpharma/PGxO/blob/master/doc/MIRO.md`

1. Does PGxO enable to represent a pharmacogenomic knowledge unit from the state of the art (*i.e.*, from a reference database or extracted from the biomedical literature), along with its provenance?

2. Does PGxO enable to represent a pharmacogenomic knowledge unit discovered from clinical data (such as EHRs), along with its provenance?

3. Does PGxO, coupled with matching mechanisms (see Chapters 4 and 5), enable to decide if two knowledge units may refer to the same thing?

We answered these questions twice, once early and once late in the iterations of PGxO development. For the former iteration [114], we manually instantiated PGxO with examples of knowledge units, associated with their provenance, from *(i)* PharmGKB, *(ii)* the literature (extracted by Semantic MEDLINE [139] or FACTA+ [162]) and *(iii)* hand designed facts corresponding to what we thought may be discovered in EHRs. For the latter iteration, we answered these questions by instantiating PGxO with knowledge units extracted programmatically from PharmGKB and the biomedical literature, and manually from results reported by studies analyzing EHR data and linked biobanks. Details on methods used to populate PGxO and create PGxLOD from these various sources are provided in the next section.

3.3 PGxLOD: a knowledge graph for pharmacogenomics

We instantiated PGxO with pharmacogenomic knowledge units from various sources, at first to answer the competency questions defined for this ontology, and then to constitute a knowledge graph integrating PGx knowledge of such various sources. The resulting knowledge graph, called PGxLOD (*PGx Linked Open Data*), constitutes a valuable community resource for pharmacogenomic and knowledge graph research. Therefore, we provide an open access to the large portion of PGxLOD data that has no license restriction[21]. Full access to PGxLOD including a small set of proprietary data is granted upon request. It should be noted that PGxLOD data respect the FAIR principles [174]. Population processes of PGxLOD are detailed in the next subsections and their results are summarized in Table 3.1.

3.3.1 Population with preexisting LOD

We initiated PGxLOD from a preexisting set of Linked Open Data made of interconnected genes, drugs, and diseases according to 6 standard databases [44]. This preexisting data set is an aggregation of data from ClinVar [95], DisGeNET [72, 136], DrugBank [175], SIDER [94] and MediSpan (a proprietary database). We completed it with data from CTD [46] and KEGG [86]. For our purpose, we try to maintain a difference between individuals and classes. However, DisGeNET and SIDER use UMLS CUIs as individuals while they are used as classes in other data sets. Hence, we modified these two data sets as follows: all triples in the ABox of DisGeNET and SIDER that involve a UMLS CUI are modified to involve a newly created individual that instantiate the UMLS CUI.

The resulting aggregation includes and relates data about drugs, diseases and phenotypes, but no pharmacogenomic relationships. Nevertheless, it groups together data related to entities involved in pharmacogenomic relationships, and cross-references between entities that may be present in different data sources, *e.g.*, a drug referenced both in DrugBank and SIDER. As

[21] `https://pgxlod.loria.fr`

Table 3.1: Main statistics of PGxLOD. Both direct instantiations and instantiations inferred from class hierarchy are counted. Equivalent individuals (via `owl:sameAs` links, see Subsection 3.3.5) are counted distinctly.

PGxO class	Number of instances
`Drug`	63,485
`GeneticFactor`	494,982
↳ `Gene`	173,426
↳ `GenomicVariation`	321,484
↳ `Haplotype`	433
↳ `Variant`	320,589
`Phenotype`	65,133
↳ `Disease`	34,528
↳ `PharmacodynamicPhenotype`	6,368
↳ `PharmacokineticPhenotype`	1,309
`PharmacogenomicRelationship`	50,435
↳ *From PharmGKB (structured data)*	3,650
↳ *From PharmGKB (text mining)*	10,240
↳ *From the literature*	36,535
↳ *From EHR studies*	10

previously explained (Section 3.1), these cross-references are of particular interest for our purpose of comparing pharmacogenomic relationships. The instantiation of PGxO with preexisting LOD sets is straightforward: we add instantiate corresponding PGxO classes with entities representing genes, drugs and diseases, using the RDF predicate `rdf:type`.

Table 3.2 summarizes results of PGxO instantiation with preexisting LOD sets. At this stage PGxLOD does not contain any pharmacogenomic relationship, but provides entities appearing as components of such relationships, as well as cross-references between these entities.

3.3.2 Population with PharmGKB data

Next, PGxO was instantiated with data from PharmGKB [173], a reference database for pharmacogenomics. PharmGKB's clinical annotations describe pharmacogenomic relationships between genes (potentially their variants), drugs, and phenotypes. They are produced by PharmGKB curators after reviewing the biomedical literature and recommendations from health agencies such as the U.S. Food and Drug Administration. In addition, PharmGKB contains cross-references, *i.e.*, identifiers of genes, variants, drugs, and phenotypes within other databases (such as NCBI Gene for genes) or ontologies (such as the Anatomical Therapeutic Chemical Classification System for drugs).

Recall that a small and outdated portion of PharmGKB is already available in the form of LOD as an output of the Bio2RDF project [54]. Therefore, inspired by Bio2RDF precursor work and following their guidelines, we developed new scripts producing a more complete RDF version of PharmGKB. These scripts transform the latest downloadable text files of PharmGKB, first, into a SQL database (with a script named *pharmgkb2sql*) and then into RDF triples (with a script named *pharmgkbsql2triples*). Table 3.3 summarizes results of PGxO instantiation with PharmGKB data (2019-07-05 release).

In clinical annotations, drug response phenotypes are provided in two manners:

Table 3.2: Statistics of the instantiation of PGxO classes with data from preexisting LOD sets. Only direct instantiations are counted.

Source	PGxO class				
	`Drug`	`Gene`	`Variant`	`Phenotype`	`Disease`
ClinVar	0	21,487	103,219	6,837	0
CTD	5,030	42,653	0	0	6,228
DisGeNET	0	20,585	200,877	7,328	15,969
DrugBank	7,740	4,300	0	0	0
KEGG	10,082	30,680	0	0	1,322
MediSpan	5,820	0	0	0	2,481
SIDER	25,479	0	0	0	6,291
UniProt	0	24,736	0	0	0
Total	54,151	144,441	304,096	14,165	32,291

Table 3.3: Statistics of the instantiation of PGxO classes with data from PharmGKB (2019-07-05 release). Only direct instantiations are counted. Only drugs, genes, variants and phenotypes having a PharmGKB ID in structured data are counted. Additional components may be detected when representing PharmGKB relationships. They are given a local URI in PGxLOD namespace and are not counted in this table.

PGxO class	Number of instances
`Drug`	4,010
`Gene`	27,232
`Variant`	5,389
`Phenotype`	3,546
`PharmacogenomicRelationship`	13,890
↳ *From structured data*	3,650
↳ Level of evidence 1A	126
↳ Level of evidence 1B	13
↳ Level of evidence 2A	102
↳ Level of evidence 2B	117
↳ Level of evidence 3	2,833
↳ Level of evidence 4	459
↳ *From text mining*	10,240
↳ Level of evidence 1A	395
↳ Level of evidence 1B	41
↳ Level of evidence 2A	325
↳ Level of evidence 2B	322
↳ Level of evidence 3	7,752
↳ Level of evidence 4	1,405

- One or several broad types are provided in a structured manner. These broad types are chosen among a short list of possibilities (Efficacy, Toxicity/ADR, Metabolism/PK, Dosage, and Other);

- Within plain-text sentences.

Hence, *pharmgkbsql2triples* performs two extractions.

First, for simplicity, each clinical annotation is transformed into a pharmacogenomic relationship whose associated phenotypes consist in provided structured broad types. Such phenotypes are represented by instances of following MeSH classes: `D016896` (*Treatment Outcome*) for Efficacy, `D064420` (*Drug-Related Side Effects and Adverse Reactions*) for Toxicity/ADR, `D010599` (*Pharmacokinetics*) for Metabolism/PK, and `D004305` (*Dose-Response Relationship, Drug*) for Dosage. We chose to discard clinical annotations only associated with the "Other" broad type. Figure 3.3 provides an example of a PharmGKB relationship represented with PGxO using broad types of drug response.

Second, the plain-text sentences contain a more detailed description of the PGx relationship, but inconveniently require additional text mining to be transformed into the structured formats of knowledge graphs such as RDF. For this reason we developed a text mining process on PharmGKB plain-text sentences. A clinical annotation can be associated with several sentences, each sentence describing the drug treatment outcome for a specific genotype also provided as plain text (*e.g.*, *GG*, **1/*1*, etc.). Hence, a pharmacogenomic relationship is created for each pair (clinical annotation, sentence). It is noteworthy that most of PharmGKB sentences describing drug treatment outcomes are written in a reduced number of similar ways. For example, multiple sentences are written as "*Patients with … genotype who are treated with … may have an increased risk of … as compared to …*". Therefore, we wrote regular expressions to capture specific phenotypes in sentences. For example, consider the following sentence:

"*Patients with CYP2D6 extensive (e.g. *1/*1), intermediate (e.g. *4/*10) or poor (e.g. *4/*4) metabolizer genotypes are more likely to have an increased response to ondansetron as compared to those with CYP2D6 ultrarapid metabolizer genotypes (e.g. *1/*1XN).*"[22]

Our aim is to detect the *increase* in the *response to ondansetron*. Such a phenotype will be represented as an entity `response to ondansetron`, linked with the pharmacogenomic relationship by the predicate `pgxo:increases` (see Figure 3.4). Recall that the clinical annotation associated with this sentence will also be transformed into a pharmacogenomic relationship whose phenotypes consist of the broad types of the annotation. Here, the only broad type is *Efficacy* transformed into an instance of the MeSH class `D016896` (see Figure 3.3). In a later reconciliation process, we will need to be able to compare the phenotype captured from text with the structured one to reconcile the two pharmacogenomic relationships. Intuitively, we want to ensure that a relationship resulting from the text mining of a clinical annotation can be considered as related to the relationship resulting from the structured data of this annotation. To this aim, captured phenotypes from the plain-text sentences of a clinical annotation instantiate the same MeSH classes as the structured phenotypes of the clinical annotation. Here, `response to ondansetron` instantiates the MeSH class `D016896`, as depicted in Figure 3.4.

Captured phenotypes can still be complex, combine different entities, and require to be normalized with existing identifiers in databases or ontologies. That is why they are then processed

[22]The clinical annotation is available at `https://www.pharmgkb.org/gene/PA128/clinicalAnnotation/982014094`

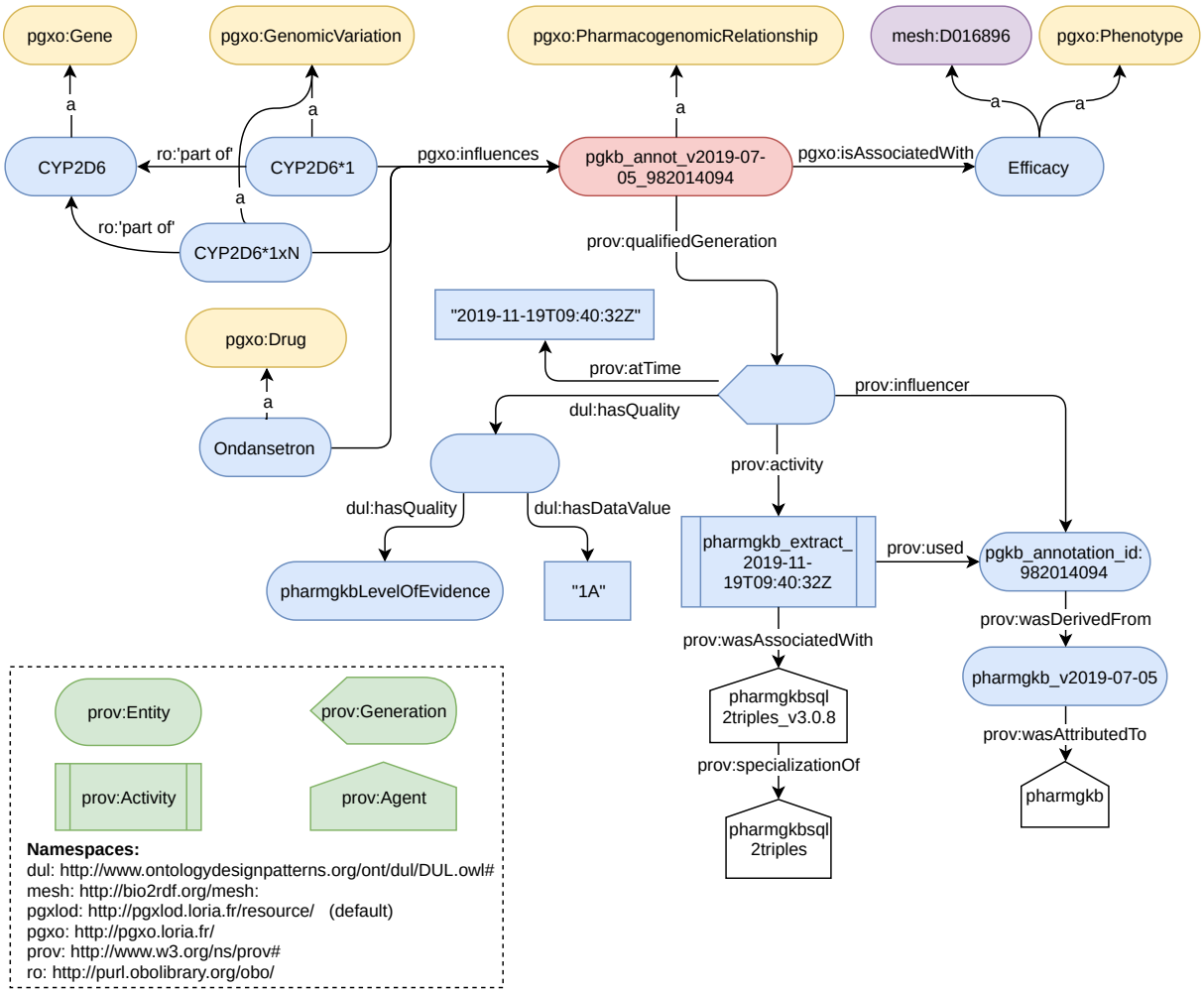


Figure 3.3: A pharmacogenomic relationship extracted from structured data of PharmGKB on November 19th, 2019 and represented with PGxO. For readability purposes, labels are used instead of URIs for components of the relationship. Only some parts of involved triples are depicted. The clinical annotation is available at `https://www.pharmgkb.org/gene/PA128/clinicalAnnotation/982014094`.

by a script called *PGxLOD_reconcile*, developed by Andon Tchechmedjiev[23]. It annotates captured phenotypes with ontology classes or NCBI gene identifiers matching all or part of their text. For example, in Figure 3.4 the ATC class `A04AA01` ("ondansetron") annotates the entity `response to ondansetron`. It also detects noisy elements, *e.g.*, a gene captured instead of a phenotype. When importing PharmGKB data into PGxLOD, we discard such noisy elements and their associated pharmacogenomic relationships. We also add `dependsOn` links between captured phenotypes and their annotations resulting from *PGxLOD_reconcile* execution. When an annotation consists in a NCBI gene URI, a `dependsOn` link is directly added between the annotated phenotype and the NCBI gene URI. When an annotation consists in an ontology class, a `dependsOn` link is added between the annotated phenotype and a created local individual instantiating the ontology class. Hence, in Figure 3.4, `response to ondansetron` is linked with `dependsOn` to a newly created individual that instantiates the ATC class `A04AA01`.

Additionally to pharmacogenomic relationships, their components in PharmGKB (*i.e.*, drugs, genes and variants) are represented in PGxLOD. To do so, they are given URIs using both PharmGKB identifiers and Bio2RDF naming conventions. In addition, cross-references to external databases and ontologies available in PharmGKB are used to map PharmGKB URIs either to URIs already defined in our LOD, or to new ones. Each cross-reference is represented by a `bio2rdf:x-ref` link. We rely on these cross-references to define mappings (*i.e.*, with `owl:sameAs` or `rdf:type` relations) in Subsection 3.3.5.

Among metadata associated with PharmGKB clinical annotations, we particularly keep the *level of evidence*. Levels of evidence are defined by PharmGKB as a six-level scale (1A, 1B, 2A, 2B, 3, 4) and given to clinical annotations by PharmGKB curators after a literature review. Higher levels (1A, 1B, 2A, 2B) indicate that a relationship has been significantly studied or is medically implemented. Lower levels (3, 4) indicate that a relationship has only been reported in a single study or lacks clear evidence[24]. Levels of evidence are of particular importance. Indeed, they may help us identify poorly-validated pharmacogenomic knowledge in the state of the art that may benefit from being reconciled with observational data to be further validated. Figure 3.3 and Figure 3.4 depict how the level of evidence of a relationship is represented with PROV-O and DUL concepts. In the case of relationships extracted from the plain-text sentences of clinical annotations, specific genotypes are also stored in metadata (see Figure 3.4). Finally, we specify in metadata our algorithm *pharmgkbsql2triples*, its version, and the version of PharmGKB. This allows coexisting extractions resulting from concurrent versions of PharmGKB or the algorithm.

3.3.3 Population with the biomedical literature

Third, we instantiated PGxO with elements automatically extracted from the biomedical literature, here abstracts of articles available in PubMed. Specifically, we extracted relationships *(i)* from a manually annotated corpus, named PGxCorpus, which was developped in the PractiKPharma project, and *(ii)* with a machine learning-based extraction of PubMed abstracts, trained on PGxCorpus. This corpus as well as the machine learning approach are described in the article of Legrand et al. [97]. Legrand et al. [97] assembled a set of 657,538 sentences from 86,520 PubMed abstracts related to pharmacogenomics. Malformed sentences were removed, based on tokenization errors. Additionally, sentences that did not contain at least one drug and one genetic factor, based on named entities recognized by PubTator [171], were also discarded.

[23] Andon Tchechmedjiev was working on the PractiKPharma as a postdoctoral researcher at LIRMM, Montpellier.

[24] The full description of each level is available at `https://www.pharmgkb.org/page/clinAnnLevels`

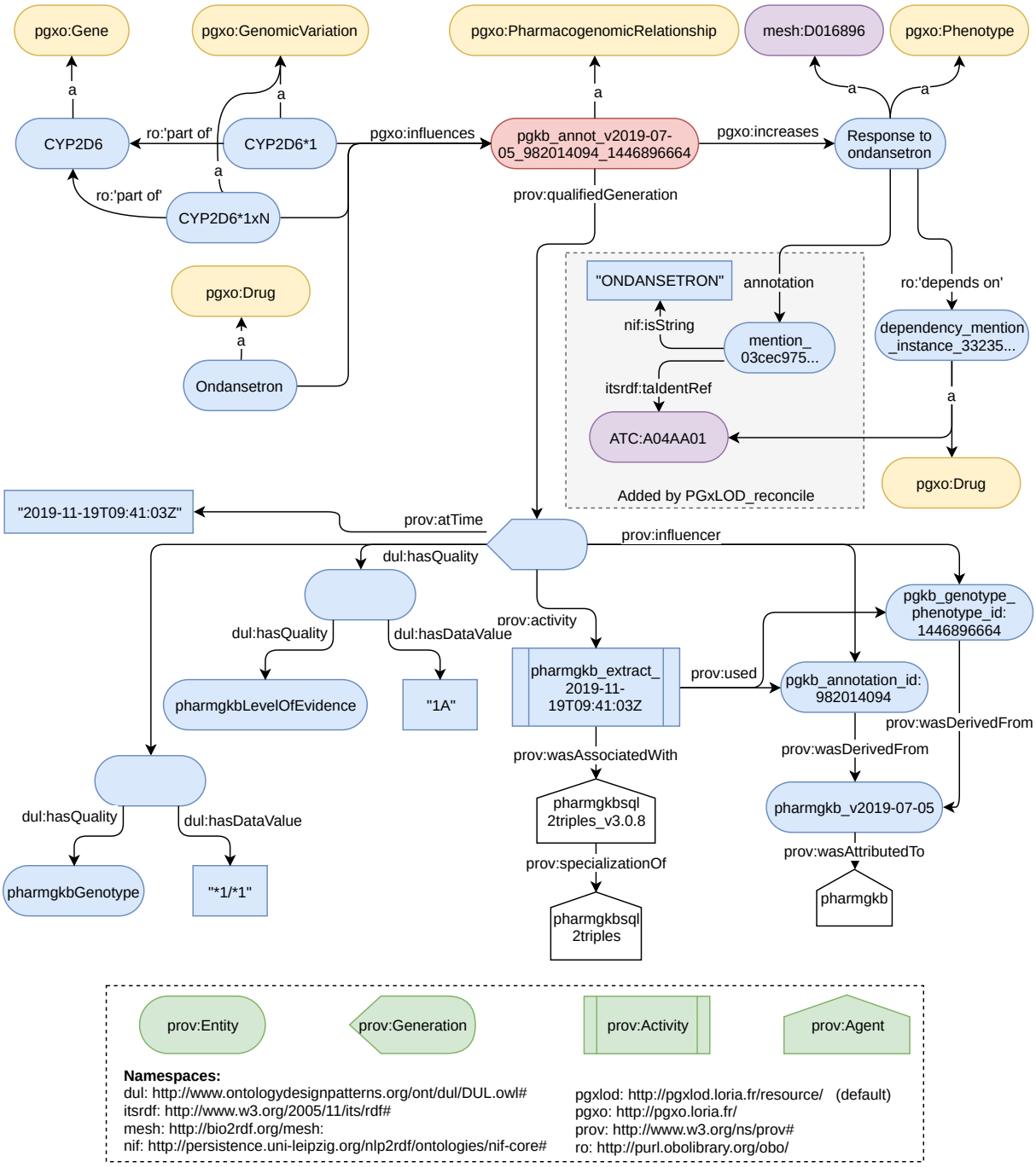


Figure 3.4: A pharmacogenomic relationship extracted from text mining of PharmGKB sentences on November 19th, 2019 and represented with PGxO. For readability purposes, labels are used instead of URIs for components of the relationship. Only some parts of involved triples are depicted. The clinical annotation is available at `https://www.pharmgkb.org/gene/PA128/clinicalAnnotation/982014094`.

Table 3.4: Reference databases and ontologies used to normalize components of pharmacogenomic relationships extracted from the biomedical literature. PGxLOD means that a local URI is created.

Order	PGxO class			
	`Drug`	`Gene`	`GenomicVariation`	`Phenotype`
1st	MeSH	NCBI Gene	dbSNP	MeSH
2nd	ChEBI	PGxLOD	PharmGKB	MEDDRA
3rd	ATC		PGxLOD	PGxLOD
4th	PGxLOD			

The final corpus is formed by 176,704 sentences. Out of those, 945 sentences were selected and manually annotated with the BRAT software [157], by 3 distinct annotators from a group of 11 pharmacists, biologists and bioinformaticians. The machine learning model trained on PGxCorpus extracts relations from all sentences.

After filtering out relationships that relate two genetic factors, two phenotypes or two drugs, both manually annotated relations and automatically extracted ones are transformed into RDF. Each extracted component is associated with a URI that is constructed, depending on its type, either from an identifier of a reference database (such as NCBI Gene for genes) or from an identifier of an ontology (such as ATC for drugs). Distinct reference databases or ontologies may be used for each type of components. Accordingly, we defined an arbitrary order of sources to search for references, presented in Table 3.4. This set of sources was motivated in part by the output of PubTator, used to recognize and normalize entities in PGxCorpus. For each type, the procedure is the following: given an entity and its type, the first reference database or ontology is searched for the entity using string matching; if no entry matches, the next reference database or ontology is searched. Lastly, if no entry is found, we create a local URI within the PGxLOD namespace. Considering an extracted component, when an entry is found in a reference database, its identifier is used to construct the corresponding URI. When an entry is found in an ontology (*i.e.*, a class of an ontology), the extracted component is given a local URI, and instantiates the ontology class.

Similarly to relationships extracted from PharmGKB, those extracted from the biomedical literature are subsequently processed by the *PGxLOD_reconcile* script. In this case, as all types of captured components (*i.e.*, genetic factors, drugs, and phenotypes) result from a text mining process, they are all considered for annotation with ontology classes or NCBI gene identifiers. As previously, noisy elements and associated pharmacogenomic relationships are detected by *PGxLOD_reconcile* and discarded during import. Annotations are also completed with `dependsOn` links as described above for phenotypes captured from PharmGKB sentences.

As a result, from the biomedical literature, we integrated 36,535 pharmacogenomic relationships (2,324 from PGxCorpus and 34,214 from the machine learning model). Table 3.5 shows statistics for the normalization of these entities to identifiers of reference databases or ontologies listed in Table 3.4. Figure 3.5 depicts the RDF encoding of a pharmacogenomic relationship extracted from the literature. It also illustrates the use of reference databases or ontologies when adding `dependsOn` links. The variant depends directly on the ITPA gene, reusing the URI from NCBI Gene, while the phenotype depends on an instance of the ChEBI class representing the mercaptopurine (an anti-cancer drug).

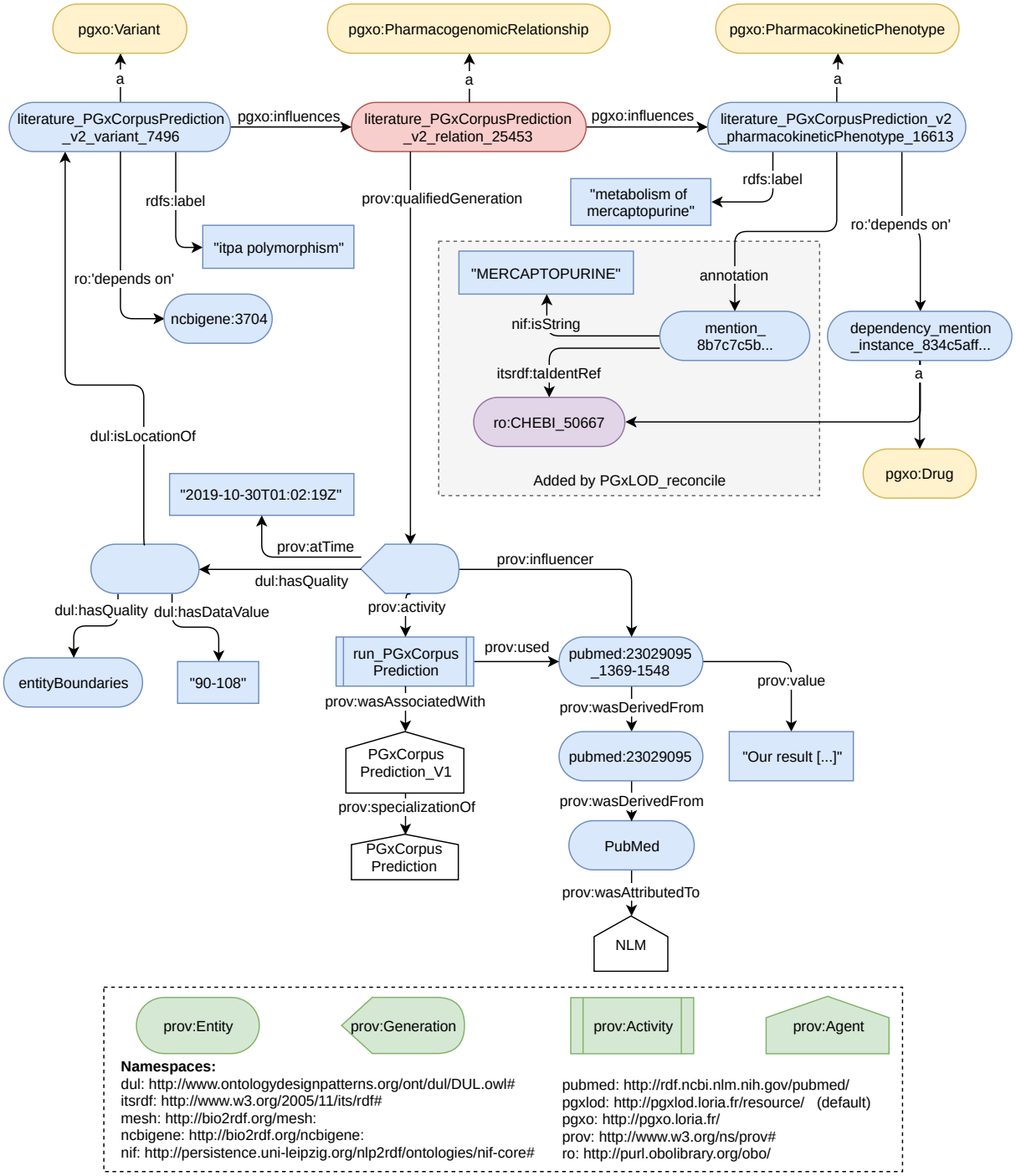


Figure 3.5: A pharmacogenomic relationship extracted from the literature on October 30th, 2019 and represented with PGxO. For readability purposes, labels are used instead of URIs for components of the relationship. Only some parts of involved triples are depicted. For example, all components have entity boundaries in metadata. The abstract is available at `https://www.ncbi.nlm.nih.gov/pubmed/23029095/`.

Table 3.5: Numbers of unique entities recognized in pharmacogenomic relationships from expert annotations (PGxCorpus) and from the machine learning model execution (Prediction), and successfully mapped with reference databases or ontologies. Reference databases and ontologies are listed in Table 3.4. PGxLOD means that a local URI is created.

(a) `Drug`

	PGxCorpus	Prediction
MeSH	244	1,330
ChEBI	27	217
ATC	14	25
PGxLOD	191	1,216
Total	476	2,788

(b) `Gene`

	PGxCorpus	Prediction
NCBI Gene	463	2,823
PGxLOD	36	50
Total	499	2,873

(c) `GenomicVariation`

	PGxCorpus	Prediction
dbSNP	77	780
PharmGKB	21	86
PGxLOD	708	10,013
Total	806	10,879

(d) `Phenotype`

	PGxCorpus	Prediction
MeSH	165	2,050
MEDDRA	0	0
PGxLOD	989	9,755
Total	1,154	11,805

3.3.4 Population with Electronic Health Records and linked biobanks studies

Fourth, we instantiated PGxO with pharmacogenomic knowledge Adrien Coulet manually extracted from the reading of ten studies on patient Electronic Health Records (EHRs) and linked biobanks [16, 49, 61, 87, 88, 121, 123, 137, 163, 172]. Here, the aim consists in assessing the adequacy of PGxO classes and predicates to represent results of such studies. This corresponds to one of our use cases, *i.e.*, considering researchers who want to compare results they obtained on their local biobanks+EHRs, to results elsewhere reported. The ten studies were selected from mentions in CPIC (Clinical Pharmacogenetics Implementation Consortium) guidelines [30] and in the literature review of Denny et al. [50]. Interestingly, out of ten, eight were preformed on the BioVU biobank and its linked EHRs [144], one on clinical data of the eMERGE Network [73] and one on data of the HEGP, a French University Hospital [82]. Six studies used a statistical analysis using linear regression and four used logistic regression. Regarding genetic factors, studies involve either a single nucleotide polymorphism (7/10), a haplotype (2/10) or an enzyme activity (1/10). For instance, Kawai et al. [88] report a statistical association between the haplotype *CYP2C9*3*, and *severe bleeding*, in patients treated with *warfarin*. Their study was performed on the BioVU biobank, linked to patient EHRs of the Vanderbuilt University Hospital [144].

Each of the ten studies listed previously results in one instance of a pharmacogenomic relationship, along with its provenance. Entities involved in such relationships were manually associated with URIs already defined in PGxLOD. For example, Figure 3.6 represents the instantiation of PGxO, achieved from the results of Neuraz et al. [123] and the thiopurine CPIC guidelines. In this particular case, no genetic data were provided in the study, but an enzyme

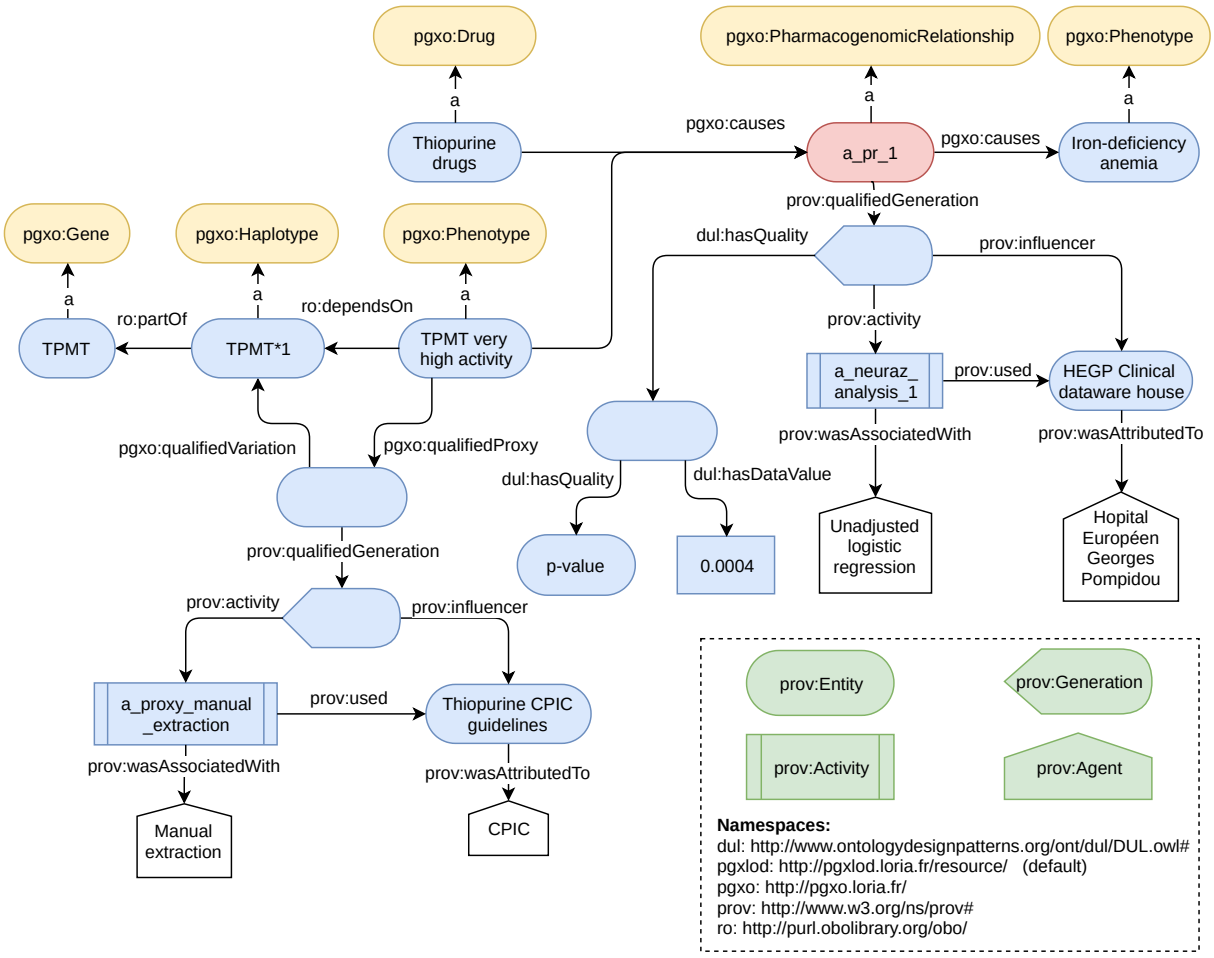


Figure 3.6: A pharmacogenomic relationship discovered from EHRs [123] and represented with PGxO. The initial association discovered from EHRs is standing between a drug response and the TPMT activity, *i.e.*, a phenotype. The latter is considered a *proxy* to the genotype of the TPMT gene, as stated by the CPIC guidelines. For readability purposes, in some cases labels are used instead of URIs.

activity. However, the TPMT enzyme activity may be considered as a proxy for the genotype of the TPMT gene, as stated in the thiopurine-related CPIC guidelines [138]. Hence, we added a RDF triple, with the `ro:dependsOn` relation type, stating that the TPMT activity depends on the TPMT haplotype. We also documented the provenance of this assertion with the `pgxo:qualifiedProxy` and `pgxo:qualifiedVariation` relation types and PROV-O concepts and relation types.

3.3.5 Generation of mappings

Finally, for our purpose of reconciling pharmacogenomic relationships involving components (*i.e.*, drugs, genetic factors, and phenotypes) from different sources, we need to be able to identify components from these sources that represent the same individual. To add such mappings to PGxLOD, our naive approach relies on cross-references from preexisting LOD sets and PharmGKB. Each cross-reference link is completed with either a `owl:sameAs` link when the

cross-reference points to an identical entity in an external database, or with a `rdf:type` link when it points to an ontology class.

However, not all cross-references may represent equivalence relationships as encoded by `owl:sameAs` links. Hence, we need to assess the correctness of resulting equivalences. To this aim, we propose the following coarse-grained validation approach. We compute connected components on PGxLOD over undirected `owl:sameAs` links. Considering undirected links corresponds to `owl:sameAs` symmetry. Computing connected components corresponds to `owl:sameAs` transitivity. Thus, each resulting connected component groups individuals that are equivalent. PGxLOD integrates 8 data sources of drugs, genetic factors, and phenotypes. Each individual from a source is represented with two URIs: one with the `http://bio2rdf.org/` prefix and another with the `http://identifiers.org/` prefix. Consequently,

- An individual represented in only one source will belong to a connected component of size 2. Thus, the minimum size to expect for connected components is 2.

- An individual represented in all sources will belong to a connected component of size $2 \times 8 = 16$. Thus, the maximum size to expect for connected components is 16.

We see in Figure 3.7 that most connected components in PGxLOD have a size between 2 and 16, which corresponds to the expected sizes. However, five connected components have a size greater than 16. From our previous explanations, we can deduce that they inevitably result from equivalences between individuals across all the integrated sources but also within sources, which violates the *Unique Name Assumption* that we suppose these sources make. Hence, such connected components clearly result from invalid `owl:sameAs` mappings. We manually investigated some of them: they seem to result from cross-references between, for example, chemical components and drugs, causing all drugs having a common chemical component to be equivalent. As the number of invalid connected components (*i.e.*, those with a size greater than 16) is reduced, we consider our naive mapping approach to be valid for our current purpose of reconciling PGx relationships. However, recall that this validation approach is only coarse-grained. Indeed, connected components whose size is in the expected range could still result from equivalences incorrectly connecting different individuals. To identify such erroneous equivalences, additional fine-grained approaches are needed.

3.4 Discussion and perspectives

In this chapter, we presented PGxO, a sufficient ontology to represent pharmacogenomic knowledge of various sources and its provenance with the combined use of PROV-O and DUL. We instantiated PGxO with knowledge extracted from PharmGKB, the biomedical literature, and clinical guidelines or EHR+biobank studies. Instantiating PGxO with knowledge from such various sources allowed us to partly answer the defined *competency questions*. Indeed, question 3 about reconciliation mechanisms will be later tackled by Chapter 4 and Chapter 5. The instantiation of PGxO resulted in the PGxLOD knowledge graph, which constitutes by itself a valuable resource for pharmacogenomic and knowledge graph research that is made available to the community. This knowledge graph is frequently improved, as highlighted by its different versions summarized in Table 3.6. To ensure reproductibility and deployability, we implemented the construction pipeline of PGxLOD with Docker. Additionally, PGxLOD respects the FAIR principles [174]. Particularly, all PGxO and PGxLOD URIs are dereferenceable and PGxLOD

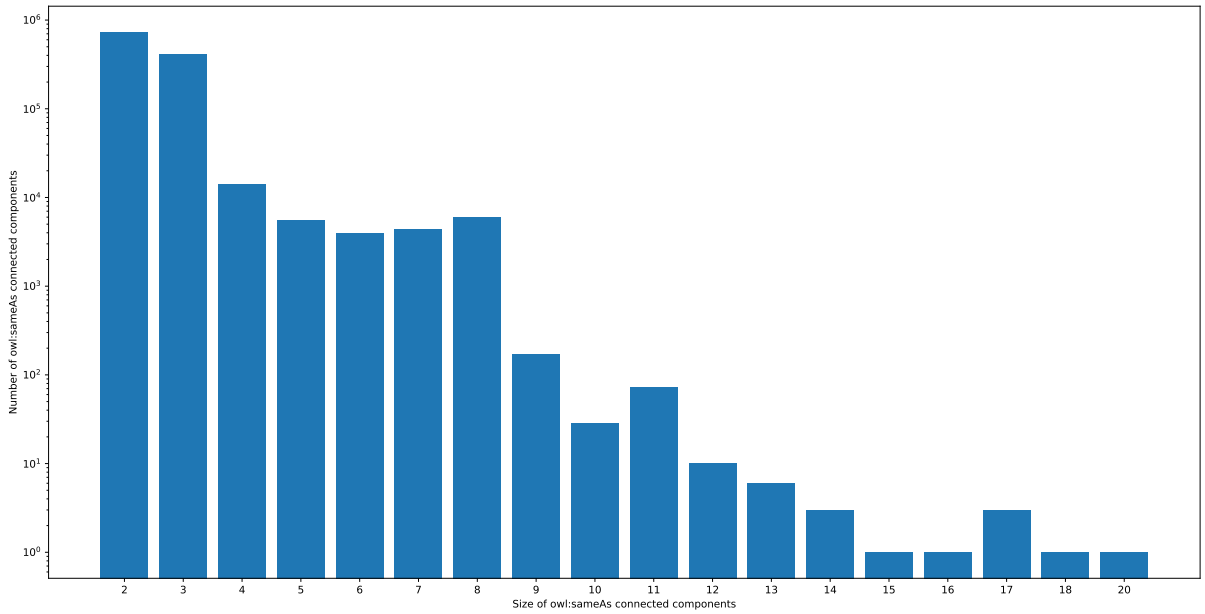


Figure 3.7: Number of `owl:sameAs` connected components by size of connected components. Expected range for sizes is $[2, 16]$.

is referenced in both LOD cloud[25] and Google dataset search[26].

By using Semantic Web technologies, our proposed encoding for PGx knowledge and its provenance metadata can easily evolve depending on one's needs. Additionally, our global framework for knowledge reconciliation in pharmacogenomics can conveniently leverage knowledge elsewhere defined (*e.g.*, ontologies or other available LOD sets). This is of particular importance as reconciliation mechanisms may rely on existing mappings and subsumption relations. Therefore, a future task resides in completing mappings between individuals from various data sources. Using both concept hierarchies and ontology-to-ontology mappings defined in the UMLS [18] or the NCBO Bioportal [80] may be of interest for this task. Matching approaches relying on structure or semantics (such as those proposed in Chapter 4 and Chapter 5) and standardization efforts for entity naming [104] (such as `http://identifiers.org`) may ease this task of mapping completion. Regarding mapping correctness, it is noteworthy that our coarse-grained validation approach does not allow to assess the quality of connected components whose size is in the expected range (*i.e.*, between 2 and 16). Indeed, they still could link together individuals that should not be considered equivalent. We leave the design and test of a fine-grained validation approach for future work. Also, PGxLOD has been built in a data integration perspective, which requires high data maintenance to follow the evolution of associated databases, LOD sets, and ontologies. Therefore, one challenge is to keep PGxLOD up-to-date with respect to the associated data sources.

The automatic instantiation of PGxO with knowledge mined from PharmGKB sentences allows us to add fine-grained phenotypes additionally to coarse-grained phenotypes extracted from structured broad types. Such phenotypes are more likely to be comparable with those found in the biomedical literature. However, it is noteworthy that this extraction should be further evaluated. Additionally, if regular expressions are not sufficient, then a text mining

[25] `https://lod-cloud.net/dataset/PGxLOD`
[26] `https://datasetsearch.research.google.com/search?query=PGxLOD`

Version of PGxLOD	Summary
v1	Initial version of PGxLOD that consisted in an aggregation of LOD sets without PGx relationships (Subsection 3.3.1 and journal article from Dalleau et al. [44])
v2	Integration of PGx relationships from PharmGKB (structured data only), the biomedical literature (*i.e.*, PGxCorpus), and clinical guidelines or EHR+biobank studies. This version corresponds to the journal article published in *BMC Bioinformatics* in 2019 [115].
v3	Integration of CTD (in the public and private versions of PGxLOD) and KEGG (in the private version of PGxLOD).
v4	Update of PharmGKB and DisGeNET data; integration of PGx relationships from plain-text sentences of PharmGKB; normalization of PGx relationships extracted from both plain-text sentences of PharmGKB and the literature with *PGxLOD_reconcile* from Andon Tchechmedjiev; new version of matching rules (see Chapter 4).

Table 3.6: Summary of the different versions of PGxLOD

approach similar to the machine learning model used for the biomedical literature could be considered. Also, diploid genotypes specified in PharmGKB sentences (*e.g.*, **1/*1*) are only stored in metadata, and thus are not directly usable by software agents. As they are provided as plain text, one future challenge resides in extracting, and then representing such genotypes for instance by individuals involved in relationships. This would allow an even more fine-grained representation of PGx knowledge that would require additional matching mechanisms than those presented in the rest of this thesis.

EHRs are difficult to access because of their sensitivity and are complex to analyze because of, *e.g.*, their heterogeneity, the important amount of information available as text, and their temporal dimension. Because this analysis was out of the scope of this thesis, we limited ourselves to a proof of concept that consisted in the manual instantiation of PGxO with elements of knowledge identified in clinical guidelines or EHR+biobank studies. One notable drawback lies in gene variants and precise drug response phenotypes being unavailable in EHRs in most cases. Indeed, genetic data are particularly sensitive, and for this reason are not shared as easily as diagnostic codes or drug prescriptions. Thus, the knowledge discovery process needs to rely on proxies such as a phenotype being a marker of the patient genotype or an unstable dose requirement being a marker of the patient sensitivity to the considered drug. Therefore, a pharmacogenomic relationship discovery from EHRs would benefit from the availability of a larger list of proxies. To the best of our knowledge, no such list is available, but would constitute a key resource for biomedical research. In addition, more contextual information about knowledge discovered from patient data would be of interest. For example, indications for which patients are treated [49] may be necessary to properly document some pharmacogenomic relationships. Representing pharmacogenomic relationships mined from EHRs with their provenance metadata would also require to consider data privacy and anonymity issues related to patient medical data. Similarly to Odgers and Dumontier [128], one possible way to overcome such issues could consist in building a restricted PGxLOD-Observational in an hospital containing locally

mined relationships. This PGxLOD-Observational could be locally reconciled and linked with the global PGxLOD. Considering these challenges, one major perspective lies in automatically instantiating PGxO with knowledge extracted by mining EHRs.

Chapter 4

Knowledge-based matching of n-ary tuples with preorders and rules

Contents

4.1 Motivation and matching task definition 74
4.2 Ontology-based preorders 75
4.2.1 Preorder $\preccurlyeq^{\mathrm{P}}$ based on links between individuals 75
4.2.2 Preorder $\preccurlyeq^{\mathcal{O}}$ based on instantiation and subsumption 76
4.3 Using preorders to define matching rules 78
4.4 Application to pharmacogenomic knowledge 81
4.5 Discussion and perspectives 83

Pharmacogenomic (PGx) relationships have been represented within the PGxLOD knowledge graph in Chapter 3. In this chapter, we propose to view them as n-ary tuples (see Figure 4.1) and, accordingly, we design a general and mathematically well-founded methodology to match such n-ary tuples within and across sources. This task is challenging since tuples can be heterogeneously represented in sources (*e.g.*, in terms of vocabularies or granularity). Precisely, given two n-ary tuples, we aim at deciding on their relatedness among five levels such as being equivalent or more specific. In our approach, we assume that the tuples to match have the same arity, the same indices for their arguments, and that they are reified using the same predicates and classes in a knowledge graph $\mathcal{K}$. Arguments are formed by sets of individuals (no literal values) and may be unknown. The matching process thus reduces to comparing each argument of the tuples and aggregating these comparisons to establish their level of relatedness. We achieve this process by defining five general rules, designed to satisfy some desired properties such as transitivity and symmetry. To tackle the heterogeneity in the representation of tuples, we enrich this structure-based comparison with domain knowledge, *e.g.*, the hierarchy of ontology classes and links between individuals. As aforementioned, our work is motivated by our application in pharmacogenomics since PGx relationships can be seen as n-ary tuples relating sets of drugs, sets of genomic variations, and sets of phenotypes. Therefore, we experimented the five proposed rules on PGx knowledge. We found insightful results that highlight noteworthy agreements and particularities within and across the sources of this knowledge. This chapter is based on an article published in the international conference ICCS 2020 [113].

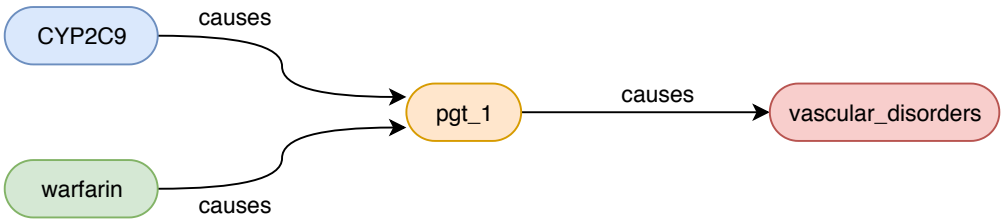


Figure 4.1: Representation of a PGx relationship between gene `CYP2C9`, drug `warfarin` and phenotype `vascular_disorders`. It can be seen as an n-ary tuple `pgt_1` = ({`warfarin`}, {`CYP2C9`}, {`vascular_disorders`}). This tuple is reified through the individual `pgt_1`, connecting its components through the `causes` predicate.

4.1 Motivation and matching task definition

We aim at matching n-ary tuples represented within a knowledge graph $\mathcal{K}$, *i.e.*, we aim at determining the relatedness level of two tuples t_1 and t_2 (*e.g.*, whether they are equivalent, more specific, or similar). We illustrate the interest of such a matching process to reconcile knowledge within the biomedical domain of pharmacogenomics. For example, Figure 4.1 depicts the PGx tuple `pgt_1`, which states that patients treated with warfarin may experience vascular disorders because of variations in the CYP2C9 gene. If a source contained the same tuple but with the genetic factor unknown, then it should be identified as less specific than `pgt_1`. Conversely, if a source contained the same tuple but with myocardial infarction as phenotype, then it should be identified as more specific than `pgt_1`.

The process of matching n-ary tuples appears naturally in the scope of ontology matching [58], *i.e.*, finding equivalences or subsumptions between classes, predicates, or instances of two ontologies. Here, we match individuals representing reified n-ary tuples, which is somewhat related to instance matching and the extraction of *linkkeys* [8]. However, we allow ourselves to state that a tuple is more specific than another, which is unusual in instance matching but common when matching classes or predicates with systems such as PARIS [158] and AMIE [65]. Besides, to the best of our knowledge, works available in the literature do not deal with the complex task of matching n-ary tuples with potentially unknown arguments formed by sets of individuals.

Precisely, the knowledge graph $\mathcal{K}$ is represented in the formalism of Description Logics [9] and thus consists of a TBox and an ABox. We consider a set $\mathcal{T}$ of n-ary tuples to match. This set is formed by tuples whose matching makes sense in a given application. For example, in our use case, $\mathcal{T}$ consists of all PGx tuples from the considered sources. All tuples in $\mathcal{T}$ have the same arity n, and their arguments are sets of individuals of $\mathcal{K}$. Such a tuple t can be formally represented as $t = (\pi_1(t), \ldots, \pi_n(t))$, where $\pi_i : \mathcal{T} \rightarrow 2^{\Delta}$ is a mapping that associates each tuple t to its i-th argument $\pi_i(t)$, which is a set of individuals included in the domain of interpretation Δ. The index set is the same for all tuples in $\mathcal{T}$. Tuples come from potentially noisy sources and some arguments may be missing. As $\mathcal{K}$ verifies the Open World Assumption, such arguments that are not explicitly specified as empty, can only be considered unknown and they are set to Δ to express the fact that all individuals may apply. To illustrate, `pgt_1` in Figure 4.1 could be seen as a ternary tuple `pgt_1` = ({`warfarin`}, {`CYP2C9`}, {`vascular_disorders`}), where arguments respectively represent the sets of involved drugs, genetic factors, and phenotypes. In Section 4.4, we propose a finer definition of components to consider predicates (*e.g.*, `causes`, `influences`, etc.) in addition to the type of involved individuals (*e.g.*, drugs, genetic factors,

etc.).

In view of our formalism, matching two n-ary tuples t_1 and t_2 comes down to comparing their arguments $\pi_i(t_1)$ and $\pi_i(t_2)$ for each $i \in \{1, \ldots, n\}$. For instance, if $\pi_i(t_1) = \pi_i(t_2)$ for all i, then t_1 and t_2 are representing the same knowledge unit, highlighting an agreement between their sources. However, such an equality test may fail often due to the heterogeneity in the representation of tuples. That is why, in the next section, we propose other tests between arguments that are based on domain knowledge.

4.2 Ontology-based preorders

As previously illustrated, the matching of two n-ary tuples t_1 and t_2 relies on the comparison of each of their arguments $\pi_i(t_1)$ and $\pi_i(t_2)$, which are sets of individuals. Such a comparison can be achieved by testing their inclusion or equality. Thus, if $\pi_i(t_1) \subseteq \pi_i(t_2)$, then $\pi_i(t_1)$ can be considered as more specific than $\pi_i(t_2)$. It is noteworthy that testing inclusion or equality implicitly considers `owl:sameAs` links that indicate identical individuals. For example, the comparison of $\{\mathtt{e}_1\}$ with $\{\mathtt{e}_2\}$ while knowing that $\mathtt{owl{:}sameAs}(\mathtt{e}_1, \mathtt{e}_2)$ results in an equality. However, additional domain knowledge can be considered to help tackle the heterogeneous representation of tuples. For instance, some individuals can be *part of* others. Individuals may also instantiate different ontological classes, which are themselves comparable through subsumption. To consider this domain knowledge in the matching process, we propose two preorders, *i.e.*, reflexive and transitive binary relations.

4.2.1 Preorder $\preccurlyeq^{\mathtt{p}}$ based on links between individuals

Several links may associate individuals in $\pi_i(t_j)$ with other individuals in $\mathcal{K}$. Some links involve a transitive and reflexive predicate (*i.e.*, a preorder). Then, for each such predicate $\mathtt{p}$, we define a preorder $\preccurlyeq^{\mathtt{p}}$ parameterized by $\mathtt{p}$ as follows:

$$\pi_i(t_1) \preccurlyeq^{\mathtt{p}} \pi_i(t_2) \Leftrightarrow \forall e_1 \in \pi_i(t_1),\ \exists e_2 \in \pi_i(t_2),\ \mathcal{K} \models \mathtt{p}(e_1, e_2) \tag{4.1}$$

Note that, from the reflexivity of $\mathtt{p}$ and the use of quantifiers $\forall$ and $\exists$, $\pi_i(t_1) \subseteq \pi_i(t_2)$ implies $\pi_i(t_1) \preccurlyeq^{\mathtt{p}} \pi_i(t_2)$. The equivalence relation $\sim^{\mathtt{p}}$ associated with $\preccurlyeq^{\mathtt{p}}$ is defined as usual by:

$$\pi_i(t_1) \sim^{\mathtt{p}} \pi_i(t_2) \Leftrightarrow \pi_i(t_1) \preccurlyeq^{\mathtt{p}} \pi_i(t_2) \text{ and } \pi_i(t_2) \preccurlyeq^{\mathtt{p}} \pi_i(t_1) \tag{4.2}$$

Proof that $\preccurlyeq^{\mathtt{p}}$ is a preorder. From the fact that $\mathtt{p}$ is reflexive, it immediately follows that $\preccurlyeq^{\mathtt{p}}$ is reflexive. Indeed, for every $E \subseteq \Delta$, $E \preccurlyeq^{\mathtt{p}} E$ since for every $e \in E$, $\mathtt{p}(e, e)$.

To prove that $\preccurlyeq^{\mathtt{p}}$ is a preorder, it remains to show that $\preccurlyeq^{\mathtt{p}}$ is transitive. Consider $E_1, E_2, E_3 \subseteq \Delta$ such that:

$$E_1 \preccurlyeq^{\mathtt{p}} E_2 \text{ and } E_2 \preccurlyeq^{\mathtt{p}} E_3.$$

In other words, $\forall e_1 \in E_1,\ \exists e_2 \in E_2,\ \mathcal{K} \models \mathtt{p}(e_1, e_2)$ and $\forall e_2 \in E_2,\ \exists e_3 \in E_3,\ \mathcal{K} \models \mathtt{p}(e_2, e_3)$. By the transitivity of $\mathtt{p}$, we then have that

$$\forall e_1 \in E_1,\ \exists e_3 \in E_3,\ \mathcal{K} \models \mathtt{p}(e_1, e_3),$$

i.e., $E_1 \preccurlyeq^{\mathtt{p}} E_3$. This shows that $\preccurlyeq^{\mathtt{p}}$ is transitive, and the proof is complete. □

Example 2. `partOf` is transitive and reflexive. Additionally, it can make sense to consider a part as more specific than its whole. Thus, this predicate is a suitable candidate for the $\preccurlyeq^{\mathtt{p}}$ preorder.

Consider three individuals e_1, e_2, e_3 such that $\mathcal{K} \models \mathtt{partOf}(e_3, e_1)$. Then it follows that:

- $\{e_1\} \preccurlyeq^{\texttt{partOf}} \{e_1, e_2\}$, from set inclusion.
- $\{e_3, e_2\} \preccurlyeq^{\texttt{partOf}} \{e_1, e_2\}$.
- $\{e_3\} \preccurlyeq^{\texttt{partOf}} \{e_1, e_2\}$.
- $\{e_3, e_1\} \sim^{\texttt{partOf}} \{e_1\}$. As e_3 is a part of e_1, having both e_3 and e_1 in the same set can be seen as a redundancy. Such a case may arise in $\mathcal{K}$ due to source heterogeneity. This redundancy is adequately identified by this equivalence result.

4.2.2 Preorder $\preccurlyeq^{\mathcal{O}}$ based on instantiation and subsumption

The second preorder we propose takes into account classes of an ontology $\mathcal{O}$ ordered by subsumption and instantiated by individuals in $\pi_i(t_j)$. We denote by classes($\mathcal{O}$) the set of all classes of $\mathcal{O}$. As it is standard in DL, $\top$ denotes the largest class in $\mathcal{O}$. Given an individual e, we denote by $\text{ci}(\mathcal{O}, e)$ the set of classes of $\mathcal{O}$ instantiated by e and distinct from $\top$, *i.e.*,

$$\text{ci}(\mathcal{O}, e) = \{C \in \text{classes}(\mathcal{O}) \setminus \{\top\} \mid \mathcal{K} \models C(e)\}\,.$$

Note that $\text{ci}(\mathcal{O}, e)$ may be empty. We explicitly exclude $\top$ from $\text{ci}(\mathcal{O}, e)$ since $\mathcal{K}$ may be incomplete. Indeed, individuals may lack instantiations of specific classes but instantiate $\top$ by default. Thus, $\top$ is excluded to prevent $\preccurlyeq^{\mathcal{O}}$ from inadequately considering these individuals more general than individuals instantiating classes other than $\top$. This unwanted behavior is detailed in Example 3.

Example 3. Consider two PGx tuples pgt_1 and pgt_2 that involve the same drug and genetic factor. Regarding the phenotype, pgt_1 is linked with an individual representing *headache* that does not instantiate the class *Headache* in $\mathcal{O}$ (*e.g.*, MeSH) but instantiates $\top$ by default. pgt_2 is linked with an individual *pain* that instantiates *Pain*, with $Headache \sqsubseteq Pain$. Intuitively, the knowledge expressed by pgt_1 is more specific than pgt_2. However, by considering instantiated classes and knowing that $Pain \sqsubseteq \top$, $\preccurlyeq^{\mathcal{O}}$ would inadequately conclude that pgt_1 is more general than pgt_2. By excluding $\top$ from $\text{ci}(\mathcal{O}, e)$, the tuples are incomparable, which avoids this unwanted behavior.

Given $\mathcal{C} = \{C_1,\ C_2,\ \ldots,\ C_k\} \subseteq \text{classes}(\mathcal{O})$, we denote by $\text{msc}(\mathcal{C})$ the set of the most specific classes of $\mathcal{C}$, *i.e.*, $\text{msc}(\mathcal{C}) = \{C \in \mathcal{C} \mid \nexists D \in \mathcal{C},\ D \sqsubset C\}$ [27]. Similarly, we denote by $\text{msci}(\mathcal{O}, e)$ the set of the most specific classes of $\mathcal{O}$, except $\top$, instantiated by an individual e, *i.e.*, $\text{msci}(\mathcal{O}, e) = \text{msc}(\text{ci}(\mathcal{O}, e))$.

Given an ontology $\mathcal{O}$, we define the preorder $\preccurlyeq^{\mathcal{O}}$ based on set inclusion and subsumption as follows:

$$\pi_i(t_1) \preccurlyeq^{\mathcal{O}} \pi_i(t_2) \Leftrightarrow \forall e_1 \in \pi_i(t_1),\ \Big[\underbrace{e_1 \in \pi_i(t_2)}_{\text{(4.3a)}}\Big] \bigvee \Big[\text{msci}(\mathcal{O}, e_1) \neq \emptyset \wedge \underbrace{\forall C_1 \in \text{msci}(\mathcal{O}, e_1),\ \exists e_2 \in \pi_i(t_2),\ \exists C_2 \in \text{msci}(\mathcal{O}, e_2),\ C_1 \sqsubseteq C_2}_{\text{(4.3b)}}\Big] \quad (4.3)$$

Clearly, if $\pi_i(t_1)$ is more specific than $\pi_i(t_2)$ and $e_1 \in \pi_i(t_1)$, then (4.3a) $e_1 \in \pi_i(t_2)$, or (4.3b) all the most specific classes instantiated by e_1 are subsumed by at least one of the most specific

[27] $D \sqsubset C$ means that $D \sqsubseteq C$ and $D \not\equiv C$.

classes instantiated by individuals in $\pi_i(t_2)$. Thus individuals in $\pi_i(t_2)$ can be seen as "more general" than those in $\pi_i(t_1)$. As before, $\preccurlyeq^{\mathcal{O}}$ induces the equivalence relation $\sim^{\mathcal{O}}$ defined by:

$$\pi_i(t_1) \sim^{\mathcal{O}} \pi_i(t_2) \Leftrightarrow \pi_i(t_1) \preccurlyeq^{\mathcal{O}} \pi_i(t_2) \text{ and } \pi_i(t_2) \preccurlyeq^{\mathcal{O}} \pi_i(t_1) \tag{4.4}$$

The preorder $\preccurlyeq^{\mathcal{O}}$ can be seen as parameterized by the ontology $\mathcal{O}$, allowing to consider different parts of the TBox of $\mathcal{K}$ for each argument $\pi_i(t_j)$, if needed.

Proof that $\preccurlyeq^{\mathcal{O}}$ is a preorder. The reflexivity of $\preccurlyeq^{\mathcal{O}}$ follows immediately from (4.3a). To see that it is also transitive, consider distinct $E_1, E_2, E_3 \subseteq \Delta$ such that $E_1 \preccurlyeq^{\mathcal{O}} E_2$ and $E_2 \preccurlyeq^{\mathcal{O}} E_3$. We need to prove that $E_1 \preccurlyeq^{\mathcal{O}} E_3$, that is,

$$\forall e_1 \in E_1,\ \Big[e_1 \in E_3\Big] \bigvee \Big[\text{msci}(\mathcal{O}, e_1) \neq \emptyset \wedge \\ \forall C_1 \in \text{msci}(\mathcal{O}, e_1),\ \exists e_3 \in E_3,\ \exists C_3 \in \text{msci}(\mathcal{O}, e_3),\ C_1 \sqsubseteq C_3\Big]$$

So let $e_1 \in E_1$. If $e_1 \in E_2$, then it follows from $E_2 \preccurlyeq^{\mathcal{O}} E_3$ that

$$\Big[e_1 \in E_3\Big] \bigvee \Big[\text{msci}(\mathcal{O}, e_1) \neq \emptyset \wedge \\ \forall C_1 \in \text{msci}(\mathcal{O}, e_1),\ \exists e_3 \in E_3,\ \exists C_3 \in \text{msci}(\mathcal{O}, e_3),\ C_1 \sqsubseteq C_3\Big], \tag{4.5}$$

and we are done. Otherwise,

$$\text{msci}(\mathcal{O}, e_1) \neq \emptyset\ \wedge\ \forall C_1 \in \text{msci}(\mathcal{O}, e_1),\ \exists e_2 \in E_2,\ \exists C_2 \in \text{msci}(\mathcal{O}, e_2),\ C_1 \sqsubseteq C_2.$$

As $E_2 \preccurlyeq^{\mathcal{O}} E_3$, we have two possible cases for each $e_2 \in E_2$:

- $e_2 \in E_3$ and for each $C_1 \in \text{msci}(\mathcal{O}, e_1)$ we also have:

$$\exists e_3 \in E_3,\ \exists C_3 \in \text{msci}(\mathcal{O}, e_3),\ C_1 \sqsubseteq C_3, \text{ or}$$

- $\exists e_3 \in E_3,\ \exists C_3 \in \text{msci}(\mathcal{O}, e_3),\ C_2 \sqsubseteq C_3$. Since the subsumption relation is transitive, $C_1 \sqsubseteq C_3$, and

$$\exists e_3 \in E_3,\ \exists C_3 \in \text{msci}(\mathcal{O}, e_3),\ C_1 \sqsubseteq C_3$$

From these two cases, it follows that for each $e_1 \in E_1$ such that

$$\text{msci}(\mathcal{O}, e_1) \neq \emptyset\ \wedge\ \forall C_1 \in \text{msci}(\mathcal{O}, e_1),\ \exists e_2 \in E_2,\ \exists C_2 \in \text{msci}(\mathcal{O}, e_2),\ C_1 \sqsubseteq C_2,$$

we have that

$$\exists e_3 \in E_3,\ \exists C_3 \in \text{msci}(\mathcal{O}, e_3),\ C_1 \sqsubseteq C_3. \tag{4.6}$$

From Equations (4.5) and (4.6), it then follows that:

$$\forall e_1 \in E_1,\ \Big[e_1 \in E_3\Big] \bigvee \Big[\text{msci}(\mathcal{O}, e_1) \neq \emptyset \wedge \\ \forall C_1 \in \text{msci}(\mathcal{O}, e_1),\ \exists e_3 \in E_3,\ \exists C_3 \in \text{msci}(\mathcal{O}, e_3),\ C_1 \sqsubseteq C_3\Big],$$

thus showing that $E_1 \preccurlyeq^{\mathcal{O}} E_3$. As the latter holds for every $E_1, E_2, E_3 \subseteq \Delta$, $\preccurlyeq^{\mathcal{O}}$ is transitive. □

Example 4. Figure 4.2 depicts six examples for the application of $\preccurlyeq^{\mathcal{O}}$:

(a) $\{e_1\}$ is more specific than $\{e_2, e_3\}$ even if e_3 instantiates a more specific class than e_1, because of the more general individual e_2.

(b) $\{e_1\}$ is more specific than $\{e_2, e_3\}$ since classes in $\text{msci}(\mathcal{O}, e_1)$ are either the same than those in $\text{msci}(\mathcal{O}, e_2)$ or more specific than those in $\text{msci}(\mathcal{O}, e_3)$.

(c) $\{e_1\}$ is more specific than $\{e_2, e_3\}$ since the class in $\text{msci}(\mathcal{O}, e_1)$ is more specific than the one in $\text{msci}(\mathcal{O}, e_2)$. There is no need to compare it with the class in $\text{msci}(\mathcal{O}, e_3)$.

(d) This example, similar to (c), illustrates the occurrence of the same behavior regardless of classes being instantiated by a single or by several individuals.

(e) $\{e_1\}$ and $\{e_2\}$ cannot be compared. Unlike the two latter examples, here, e_1 instantiates a class that is more specific than the class instantiated by e_2, but also a class that is not comparable.

(f) $\{e_1\}$ and $\{e_2\}$ are equivalent by instantiating the same most specific class.

4.3 Using preorders to define matching rules

Let $t_1, t_2 \in \mathcal{T}$ be two n-ary tuples to match. We assume that each argument $i \in \{1, \ldots, n\}$ is endowed with a preorder $\preccurlyeq_i \in \left\{\subseteq, \preccurlyeq^{\mathtt{P}}, \preccurlyeq^{\mathcal{O}}\right\}$ that enables the comparison of $\pi_i(t_1)$ and $\pi_i(t_2)$. We can define rules that aggregate such comparisons for all $i \in \{1, \ldots, n\}$ and establish the relatedness level of t_1 and t_2. Hence, our matching approach comes down to applying these rules to every ordered pair (t_1, t_2) of n-ary tuples from $\mathcal{T}$.

Here, we propose the following five relatedness levels: $=$, $\sim$, $\preccurlyeq$, $\lesseqgtr$, and $\propto$, from the strongest to the weakest. Accordingly, we propose five matching rules of the form $B \Rightarrow H$, where B expresses the conditions of the rule, testing equalities, equivalences, or inequalities between arguments of t_1 and t_2. Classically, these conditions can be combined using conjunctions or disjunctions, respectively denoted by $\wedge$ and $\vee$. If B holds, H expresses the relatedness between t_1 and t_2 to add to $\mathcal{K}$. Rules are applied from Rule 1 to Rule 5. Once conditions in B hold for a rule, H is added to $\mathcal{K}$ and the following rules are discarded, meaning that at most one relatedness level is added to $\mathcal{K}$ for each pair of tuples. When no rule can be applied, t_1 and t_2 are considered incomparable and nothing is added to $\mathcal{K}$. The first four rules are the following:

Matching rule 1. $\forall i \in \{1, \ldots, n\}\,,\ \pi_i(t_1) = \pi_i(t_2) \Rightarrow t_1 = t_2$

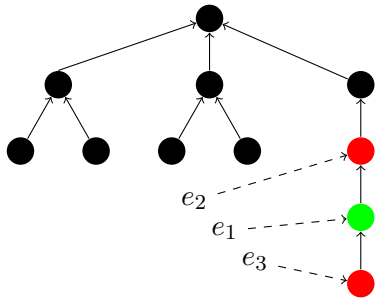
Matching rule 2. $\forall i \in \{1, \ldots, n\}\,,\ \pi_i(t_1) \sim_i \pi_i(t_2) \Rightarrow t_1 \sim t_2$

Matching rule 3. $\forall i \in \{1, \ldots, n\}\,,\ \pi_i(t_1) \preccurlyeq_i \pi_i(t_2) \Rightarrow t_1 \preccurlyeq t_2$

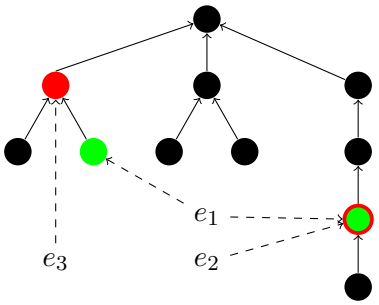
Matching rule 4.

$$\forall i \in \{1, \ldots, n\}\,,\ [\ (\pi_i(t_1) = \pi_i(t_2)) \vee (\pi_i(t_2) \neq \Delta \wedge \pi_i(t_1) \preccurlyeq_i \pi_i(t_2)) \ \vee \\ (\pi_i(t_1) \neq \Delta \wedge \pi_i(t_2) \preccurlyeq_i \pi_i(t_1))\ ] \Rightarrow t_1 \lesseqgtr t_2$$

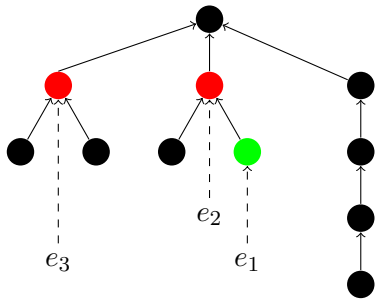
Rule 1 states that t_1 and t_2 are identical ($=$) whenever t_1 and t_2 coincide on each argument. Rule 2 states that t_1 and t_2 are equivalent ($\sim$) whenever each argument $i \in \{1, \ldots, n\}$ of t_1 is equivalent to the same argument of t_2. Rule 3 states that t_1 is more specific than t_2 ($\preccurlyeq$) whenever each argument $i \in \{1, \ldots, n\}$ of t_1 is more specific than the same argument of t_2 with regard to



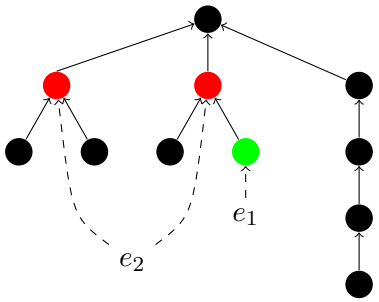
(a) $\{e_1\} \preccurlyeq^{\mathcal{O}} \{e_2, e_3\}$



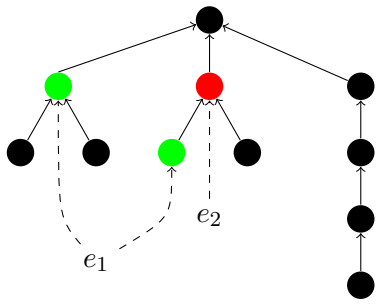
(b) $\{e_1\} \preccurlyeq^{\mathcal{O}} \{e_2, e_3\}$



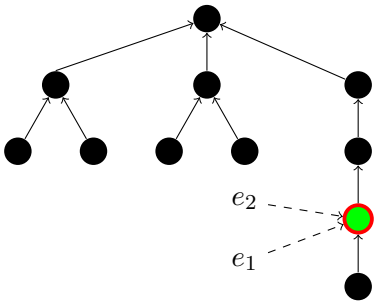
(c) $\{e_1\} \preccurlyeq^{\mathcal{O}} \{e_2, e_3\}$



(d) $\{e_1\} \preccurlyeq^{\mathcal{O}} \{e_2\}$



(e) $\{e_1\} \not\preccurlyeq^{\mathcal{O}} \{e_2\}$; $\{e_2\} \not\preccurlyeq^{\mathcal{O}} \{e_1\}$



(f) $\{e_1\} \sim^{\mathcal{O}} \{e_2\}$

Figure 4.2: Examples of use cases of the preorder $\preccurlyeq^{\mathcal{O}}$. Circles represent ontology classes. Solid arrows depict class subsumptions and dashed arrows depict class instantiations by individuals e_1, e_2, and e_3. The green color identifies classes in $\mathrm{msci}(\mathcal{O}, e_1)$. The red color identifies classes in $\mathrm{msci}(\mathcal{O}, e_2)$ and $\mathrm{msci}(\mathcal{O}, e_3)$.

$\preccurlyeq_i$. Rule 4 states that t_1 and t_2 have comparable arguments ($\lesseqgtr$) whenever they have the same specified arguments (*i.e.*, different from Δ), and these arguments are comparable with regard to $\preccurlyeq_i$. Rules 1 to 3 satisfy the transitivity property. Additionally, Rules 1, 2, and 4 satisfy the symmetry property. Results of these four rules are respectively encoded in $\mathcal{K}$ by `owl:sameAs`, `skos:closeMatch`, `skos:broadMatch`, and `skos:relatedMatch` links.

In Rules 1 to 4, comparisons are made argument-wise. However, other relatedness cases may require to aggregate over arguments. For example, we may want to compare all individuals involved in two tuples, regardless of their arguments. Alternatively, we may want to consider two tuples as weakly related if their arguments have a specified proportion of comparable individuals. To this aim, we propose Rule 5. Let $\mathbb{I} = \{I_1, \ldots, I_m\}$ be a partition of $\{1, \ldots, n\}$, defined by the user at the beginning of the matching process. We define the aggregated argument I_k of t_j as the union of all specified $\pi_i(t_j)$ (*i.e.*, different from Δ) for $i \in I_k$. Formally,

$$\pi_{I_k}(t_j) = \bigcup_{\substack{i \in I_k \\ \pi_i(t_j) \neq \Delta}} \pi_i(t_j).$$

If there is no $i \in I_k$ such that $\pi_i(t_j) \neq \Delta$, then $\pi_{I_k}(t_j)$ is unknown and set to Δ. We assume that each aggregated argument $I_k \in \mathbb{I}$ is endowed with a preorder $\preccurlyeq_{I_k} \in \left\{\subseteq, \preccurlyeq^{\mathtt{p}}, \preccurlyeq^{\mathcal{O}}\right\}$. We denote by $\mathrm{SSD}(\pi_{I_k}(t_1), \pi_{I_k}(t_2))$ the semantic set difference between $\pi_{I_k}(t_1)$ and $\pi_{I_k}(t_2)$, *i.e.*,

$$\mathrm{SSD}(\pi_{I_k}(t_1), \pi_{I_k}(t_2)) = \{e_1 \mid e_1 \in \pi_{I_k}(t_1) \text{ and } \{e_1\} \not\preccurlyeq_{I_k} \pi_{I_k}(t_2)\}.$$

Intuitively, it is the set of elements in $\pi_{I_k}(t_1)$ preventing it from being more specific than $\pi_{I_k}(t_2)$ with regard to $\preccurlyeq_{I_k}$. We define the operator $\propto_{I_k}$ as follows:

$$\pi_{I_k}(t_1) \propto_{I_k} \pi_{I_k}(t_2) = \begin{cases} 1 \text{ if } \pi_{I_k}(t_1) \preccurlyeq_{I_k} \pi_{I_k}(t_2) \text{ or } \pi_{I_k}(t_2) \preccurlyeq_{I_k} \pi_{I_k}(t_1) \\ 1 - \dfrac{|\mathrm{SSD}(\pi_{I_k}(t_1), \pi_{I_k}(t_2)) \ \cup \ \mathrm{SSD}(\pi_{I_k}(t_2), \pi_{I_k}(t_1))|}{|\pi_{I_k}(t_1) \ \cup \ \pi_{I_k}(t_2)|} \text{ otherwise} \end{cases}$$

This operator returns a number measuring the similarity between $\pi_{I_k}(t_1)$ and $\pi_{I_k}(t_2)$. This number is equal to 1 if the two aggregated arguments are comparable. Otherwise, it is equal to 1 minus the proportion of incomparable elements. We denote by $\mathbb{I}_{\neq\Delta}(t_1, t_2)$ the set of indices of aggregated arguments that are specified for both t_1 and t_2 (*i.e.*, different from Δ). Formally,

$$\mathbb{I}_{\neq\Delta}(t_1, t_2) = \{I_k \mid I_k \in \mathbb{I} \text{ and } \pi_{I_k}(t_1) \neq \Delta \text{ and } \pi_{I_k}(t_2) \neq \Delta\}.$$

Then, Rule 5 is defined as follows:

Matching rule 5. Let $\mathbb{I} = \{I_1, \ldots, I_m\}$ be a partition of $\{1, \ldots, n\}$, and let $\gamma_{\neq\Delta}$, γ_S, and γ_C be three parameters, all fixed at the beginning of the matching process.

$$\Big(|\mathbb{I}_{\neq\Delta}(t_1, t_2)| \ \geq \gamma_{\neq\Delta}\Big) \bigwedge \Big(\big[\forall I_k \in \mathbb{I}_{\neq\Delta}(t_1, t_2), \ \pi_{I_k}(t_1) \propto_{I_k} \pi_{I_k}(t_2) \geq \gamma_S\big] \vee \big[\Big(\sum_{I_k \in \mathbb{I}_{\neq\Delta}(t_1,t_2)} \mathbb{1}\left(\pi_{I_k}(t_1) \propto_{I_k} \pi_{I_k}(t_2) = 1\right)\Big) \geq \gamma_C\big]\Big) \Rightarrow t_1 \propto t_2$$

Rule 5 is applicable if at least $\gamma_{\neq\Delta}$ aggregated arguments are specified for both t_1 and t_2. Then, t_1 and t_2 are weakly related ($\propto$) whenever all these specified aggregated arguments have a similarity of at least γ_S or when at least γ_C of them are comparable. Notice that $\propto$ is symmetric. Results of Rule 5 are encoded by `skos:related` links.

Table 4.1: Statistics about the public version of PGxLOD. # denotes "number of". Instances linked by `owl:sameAs` are counted separately. `partOf` links are counted without transitivity inference. All PGx tuples were programmatically extracted from their sources, except the ten tuples from EHRs that were manually added as a proof of concept.

Class	# instances	Predicate	# links
`Drug`	47,584	`partOf`	16,697
`GeneticFactor`	464,302	`dependsOn`	23,976
`Phenotype`	61,330		
`PharmacogenomicRelationship`	50,435		
↳ From PharmGKB (structured data)	3,650		
↳ From PharmGKB (clinical annotations)	10,240		
↳ From biomedical literature	36,535		
↳ From EHRs	10		

4.4 Application to pharmacogenomic knowledge

Our methodology was motivated by the problem of matching pharmacogenomic (PGx) tuples. Accordingly, we tested this methodology on the public version of PGxLOD[28] [115].

We recall that, in PGxLOD, PGx tuples are represented using classes and predicates of the PGxO ontology. PGx tuples are n-ary, and thus, they are reified as instances of the `PharmacogenomicRelationship` class. All the individuals involved in PGx tuples instantiate the `Drug`, `GeneticFactor`, or `Phenotype` classes. They are linked with reified PGx tuples by the predicates depicted in Figure 3.2 that qualify their association to tuples. It is noteworthy that, in PGxLOD, `partOf` links indicate that instances of `GeneticFactor` compose others such instances. For example, a genomic variation may be part of a gene. Similarly, instances of `Phenotype` may have dependencies, expressed with `dependsOn` links. These dependencies enable representing complex phenotypes that refer to other phenotypes or drugs. For example *warfarin-caused hemorrhage* is a phenotype linked with `dependsOn` to *hemorrhage* and *warfarin*. The TBox of PGxLOD contains, alongside PGxO, three other ontologies: individuals representing drugs may instantiate classes from ATC or ChEBI, and individuals representing phenotypes may instantiate classes from MeSH. Table 4.1 provides global statistics about the public version PGxLOD.

To apply the matching rules on PGx tuples, we specified their arguments. Each argument of a tuple is the set of individuals with a specific type (`Drug`, `GeneticFactor`, or `Phenotype`) that are linked with a specific predicate to the tuple. For example, given *pgt* a PGx tuple, $\pi_{\texttt{Phenotype,causes}}(pgt)$ contains all the phenotypes caused by *pgt*. Hence, as there are 3 types of individuals and 38 predicates, PGx tuples have $3 \times 38 = 114$ arguments. Once arguments of tuples are specified, their associated preorders can be defined. Based on the available data and knowledge in PGxLOD, it makes sense to use the $\preccurlyeq^{\texttt{partOf}}$ preorder for arguments involving instances of `GeneticFactor`. Similarly, we use $\preccurlyeq^{\mathcal{O}_{\texttt{Drug}}}$ and $\preccurlyeq^{\mathcal{O}_{\texttt{Phenotype}}}$ as preorders for arguments respectively involving instances of `Drug` and `Phenotype`, where $\mathcal{O}_{\texttt{Drug}}$ is the concatenation of ATC and ChEBI, and $\mathcal{O}_{\texttt{Phenotype}}$ is the MeSH ontology.

Finally, to apply Rule 5, a natural three-way partition of arguments appears based on the

[28] See Chapter 3 – `https://pgxlod.loria.fr`

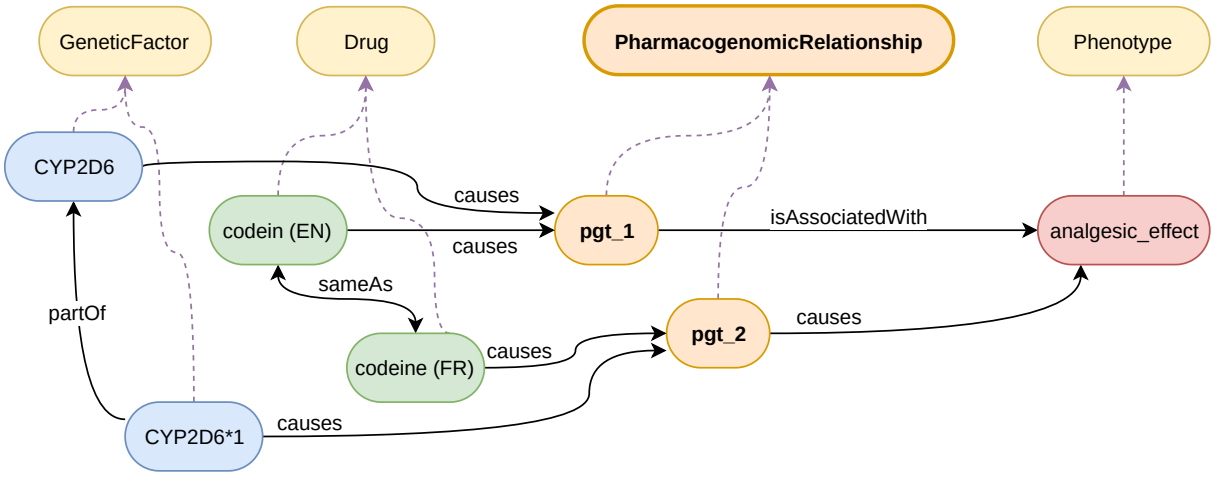


Figure 4.3: Example of two PGx tuples represented with PGxO. These tuples will be matched by Rule 3, resulting in `pgt_2` $\preccurlyeq$ `pgt_1`. Dashed arrows depict instantiations.

three types of involved individuals. Therefore, discarding predicates, we gather all drugs, genetic factors, and phenotypes involved in a tuple in three aggregated arguments. To benefit from dependencies of complex phenotypes, we choose to add them to the aggregated arguments corresponding to their type. For example, in *warfarin-caused hemorrhage*, *hemorrhage* is added to the aggregated argument representing phenotypes and *warfarin* is added to the one representing drugs. We arbitrarily set $\gamma_{\neq\Delta} = 3$, $\gamma_S = 0.8$, and $\gamma_C = 2$. These values mean that two PGx tuples pgt_1 and pgt_2 will be matched by Rule 5 if their three aggregated arguments are specified (*i.e.*, different from Δ). Additionally, each of the three aggregated arguments of pgt_1 must have at least 80% of comparable individuals with the same aggregated argument of pgt_2, or at least two aggregated arguments of pgt_1 must be comparable with the same aggregated arguments of pgt_2.

To illustrate the interest of this formalization as well as reasoning mechanisms from Description Logics, let us consider `pgt_1` and `pgt_2`, the two PGx tuples depicted in Figure 4.3. `pgt_2` causes the phenotype `analgesic_effect`, and `pgt_1` is associated with this phenotype. Thus, by applying reasoning mechanisms along the hierarchy of predicates, it follows that:

$$\begin{aligned}
\pi_{\texttt{Phenotype,causes}}(\texttt{pgt_1}) &= \Delta;\\
\pi_{\texttt{Phenotype,isAssociatedWith}}(\texttt{pgt_1}) &= \{\texttt{analgesic_effect}\};\\
\pi_{\texttt{Drug,isCausedBy}}(\texttt{pgt_1}) &= \{\texttt{codein (EN)}\};\\
\pi_{\texttt{GeneticFactor,isCausedBy}}(\texttt{pgt_1}) &= \{\texttt{CYP2D6}\};\\
\pi_{\texttt{Phenotype,causes}}(\texttt{pgt_2}) &= \{\texttt{analgesic_effect}\};\\
\pi_{\texttt{Phenotype,isAssociatedWith}}(\texttt{pgt_2}) &= \{\texttt{analgesic_effect}\};\\
\pi_{\texttt{Drug,isCausedBy}}(\texttt{pgt_2}) &= \{\texttt{codéine (FR)}\};\\
\pi_{\texttt{GeneticFactor,isCausedBy}}(\texttt{pgt_2}) &= \{\texttt{CYP2D6*1}\}.
\end{aligned}$$

By definition of $\preccurlyeq^{\mathcal{O}_{\texttt{Phenotype}}}$, $\preccurlyeq^{\mathcal{O}_{\texttt{Drug}}}$, and $\preccurlyeq^{\texttt{partOf}}$,

$$\pi_{\texttt{Phenotype,causes}}(\texttt{pgt_2}) \preccurlyeq^{\mathcal{O}_{\texttt{Phenotype}}} \pi_{\texttt{Phenotype,causes}}(\texttt{pgt_1});$$
$$\pi_{\texttt{Phenotype,isAssociatedWith}}(\texttt{pgt_2}) \preccurlyeq^{\mathcal{O}_{\texttt{Phenotype}}} \pi_{\texttt{Phenotype,isAssociatedWith}}(\texttt{pgt_1});$$
$$\pi_{\texttt{Drug,isCausedBy}}(\texttt{pgt_2}) \preccurlyeq^{\mathcal{O}_{\texttt{Drug}}} \pi_{\texttt{Drug,isCausedBy}}(\texttt{pgt_1});$$
$$\pi_{\texttt{GeneticFactor,isCausedBy}}(\texttt{pgt_2}) \preccurlyeq^{\texttt{partOf}} \pi_{\texttt{GeneticFactor,isCausedBy}}(\texttt{pgt_1}).$$

Therefore, by applying Rule 3, `pgt_2` is more specific than `pgt_1`. This makes sense as the predicate connecting `analgesic_effect` with `pgt_2` is more specific with than the one used with `pgt_1`. Additionally, the genetic factor involved in `pgt_2` is more specific than the one involved in `pgt_1`.

We implemented our matching methodology in C++ with multithreading. Our code is available on GitHub[29]. Our program interacts with the knowledge base thanks to SPARQL queries. Previously, we indicated that an unspecified argument of an n-ary tuple is set to Δ. Accordingly, when a SPARQL query returns $\emptyset$ for an argument of a tuple, it is interpreted as if it is returning Δ. On PGxLOD, the matching rules led to perform $\binom{50,435}{2} = 1,271,819,395$ comparisons in approximately 54 hours using 4 cores and 15 GB of RAM. We obtained the matching results summarized in Table 4.2 and discussed in Section 4.5

4.5 Discussion and perspectives

In Table 4.2, we observe only a few inter-source links as only Rules 2, 3, and 5 generated links across sources. This predominance of intra-source links may be caused by missing mappings between the vocabularies used in sources. Indeed, individuals in sources may be represented by different vocabularies. In this case, our matching process requires mappings between such vocabularies to compare individuals. Thus, missing mappings prevent the matching rules to be applied. This result underlines the relevance of enriching the knowledge base with ontology-to-ontology mappings, for example, those defined in the NCBO BioPortal. We also notice that Rule 5 generated more links than the other rules, which emphasizes the importance of weaker relatedness levels to align sources and overcome their heterogeneity. Here, by only considering a specified proportion of comparable individuals, this rule alleviates missing class instantiations or mappings.

Some results were expected and therefore seem to validate our approach. For example, we notice that all `owl:sameAs` links are intra-source and thus indicate duplicates. This is expected in the case of the literature since several articles could mention the same tuple. The 5 `skos:closeMatch` links between tuples from structured data and clinical annotations of PharmGKB highlight expected agreements between these two related sources. However, linked tuples are expressed with different individuals instantiating the same ontology classes, preventing their reconciliation with `owl:sameAs`. Some tuples from the literature appear more general than those of PharmGKB (with 15 and 42 `skos:broadMatch` links). These links are a foreseen consequence of the completion process of PharmGKB. Indeed, curators achieve this completion after a literature review, inevitably leading to tuples more specific or equivalent to the ones in reviewed articles. Interestingly, our methodology could ease such a review by pointing out articles describing similar tuples. Clinical annotations of PharmGKB are in several cases more specific than structured data (9,325 `skos:broadMatch` links). This is also expected as structured data are a broad-level summary of more complex phenotypes detailed in clinical annotations.

[29] `https://github.com/pmonnin/tcn3r`

Table 4.2: Number of links resulting from each rule. Links are generated between tuples of distinct sources or within the same source. PGKB stands for "PharmGKB", sd for "structured data", and ca for "clinical annotations". As Rules 1, 2, 4, and 5 satisfy symmetry, links from t_1 to t_2 as well as from t_2 to t_1 are counted. Similarly, as Rules 1 to 3 satisfy transitivity, transitivity-induced links are counted. Regarding `skos:broadMatch` links, rows represent origins and columns represent destinations.

		PGKB (sd)	PGKB (ca)	Literature	EHRs
Links from Rule 1	PGKB (sd)	166	0	0	0
	PGKB (ca)	0	10,134	0	0
	Literature	0	0	122,646	0
	EHRs	0	0	0	0
Links from Rule 2	PGKB (sd)	0	5	0	0
	PGKB (ca)	5	1,366	0	0
	Literature	0	0	16,692	0
	EHRs	0	0	0	0
Links from Rule 3	PGKB (sd)	87	3	15	0
	PGKB (ca)	9,325	605	42	0
	Literature	0	0	75,138	0
	EHRs	0	0	0	0
Links from Rule 4	PGKB (sd)	20	0	0	0
	PGKB (ca)	0	110	0	0
	Literature	0	0	18,050	0
	EHRs	0	0	0	0
Links from Rule 5	PGKB (sd)	100,596	287,670	414	2
	PGKB (ca)	287,670	706,270	1,103	19
	Literature	414	1,103	1,082,074	15
	EHRs	2	19	15	0

The results of Rule 4 underline that sources may contain tuples with comparable arguments. Source owners can benefit from such results by considering adding a tuple formed by the most specific arguments of the matched tuples. We notice that only Rule 5 generates links between the tuples from EHRs and other sources. As tuples from EHRs are manually represented, there are only a few of them, minimizing the chance of overlap with other sources. Additionally, phenotypes involved in tuples from EHRs are very specific, making their comparison with phenotypes from biomedical literature or PharmGKB difficult.

Regarding our method, using rules is somehow off the current machine learning trend [6, 124, 142]. However, writing simple and well-founded rules constitutes a valid first step before applying machine learning approaches. Indeed, such explicit rules enable generating a "silver" standard for matching, which may be useful to either train or evaluate supervised approaches. Additionally, our rules are simple enough to be generally true and useful in other domains. They are readable and thus provide a basis of explanation for the matching results. Their readability also facilitates their review by experts of another application domain, for example, to confirm their suitability and to define the arguments and preorders to use.

By relying on instantiated classes and links between individuals, we illustrate how domain knowledge and reasoning mechanisms can serve a structure-based matching. It is noteworthy that the $\preccurlyeq^{\mathcal{O}}$ preorder may result in many equivalences if the ontology $\mathcal{O}$ is not granular enough in terms of width and depth. Such equivalences may make sense, depending on the application domain and the ontology. If not, the two other preorders (*i.e.*, $\subseteq$ and $\preccurlyeq^{\mathtt{P}}$) could be used. It is for now up to experts to choose the correct preorder for each argument. However, in future works, we could investigate metrics about domain knowledge that may guide their choice. Conditions under which preorders $\preccurlyeq^{\mathtt{P}}$ and $\preccurlyeq^{\mathcal{O}}$ could be merged into one unique preorder also deserve a deeper study. Finally, it would be interesting to consider the integration of this purely symbolic approach with machine learning approaches.

Chapter 5

Rediscovering alignment relations with Graph Convolutional Networks

Contents

5.1 Motivation and learning task definition 88
5.2 Matching nodes with Graph Convolutional Networks and clustering 90
5.2.1 Approach outline 90
5.2.2 Learning node embeddings with Graph Convolutional Networks and the Soft Nearest Neighbor Loss 90
5.2.3 Matching nodes by clustering their embeddings 92
5.3 Evaluating the influence of applying inference rules associated with domain knowledge 92
5.4 Experiments 94
5.4.1 Knowledge graph and gold clusters of similar nodes 94
5.4.2 Learning node embeddings 96
5.4.3 Clustering 98
5.4.4 Distance analysis 99
5.5 Discussion and perspectives 99

In Chapter 4, results showed the need for flexible matching rules to cope with the heterogeneous representations of units to match. Additionally, manually capturing possible matching patterns in rules is a tedious and time-consuming task, especially to cover most of the possible matching patterns in voluminous knowledge graphs. This assessment motivated us to explore more flexible approaches that can automatically learn similarities given a set of examples, such examples being the output of the rules in a "knowledge graph as silver standard" approach [132]. Hence, in this chapter, we consider graph embedding since the continuous representation of embeddings may provide the needed flexibility to align heterogeneous units [75]. Particularly, we propose to match nodes of a knowledge graph by *(i)* learning node embeddings with Graph Convolutional Networks [92, 146] such that similar nodes have low distance in the embedding space, and *(ii)* clustering nodes based on their embeddings. We experiment this approach on our pharmacogenomic (PGx) use case and particularly investigate the interplay between domain knowledge and GCN models with the two following main focuses. First, we measure the improvement in matching results when applying various inference rules associated with domain

knowledge, independently or combined. Second, while our GCN model is agnostic of the exact alignment relations (*e.g.*, equivalence, weak similarity), we observe that distances in the embedding space are coherent with these different relations, somehow corresponding to their rediscovery by the model. The work presented in the chapter is an extension of the preliminary approach presented in an article published in the international workshop *DL4KG 2019* [120].

5.1 Motivation and learning task definition

In this chapter, we consider the task of matching nodes in a knowledge graph. Recall that knowledge graphs expressed using Semantic Web standards [15] can be seen as directed labeled multigraphs. In such knowledge graphs, nodes represent entities of a world (*e.g.*, places, drugs), literals (*e.g.*, dates, integers), or classes of individuals (*e.g.*, `Person`, `Drug`). Here, we focus on matching nodes representing entities. Due to the heterogeneous representations of nodes, *similarity links* existing between them may use different alignment relations: some relations may indicate that two nodes are equivalent (*e.g.*, `owl:sameAs`), weakly related (*e.g.*, `skos:related`), or that one is more specific than the other (*e.g.*, `skos:broadMatch`). Motivated by the need of flexibility highlighted by the results in Chapter 4, we explore graph embedding, *i.e.*, low-dimensional vectors that represent graph substructures (*e.g.*, nodes, edges, subgraphs) while preserving graph properties (see Subsection 2.3.2). Indeed, the continuous representation of embeddings may provide the needed flexibility to cope with heterogeneous representation [75].

It is noteworthy that the present task of matching nodes can be alternatively considered as a link prediction task (*i.e.*, predicting similarity links between nodes) or as a node clustering task (*i.e.*, grouping similar nodes in clusters). In our work, we adopt the node clustering approach. More precisely, we propose to match nodes that represent entities through the approach outlined in Figure 5.1. We learn node embeddings with Graph Convolution Networks (GCNs) [92, 146] such that similar nodes have a low distance between their embeddings. Then, we apply a clustering algorithm on the embedding space and consider nodes that belong to the same cluster as similar. These resulting clusters are evaluated by being compared with *gold clusters*, which we define as groups of nodes linked directly or indirectly through *similarity links* existing in the knowledge graph. Hence, our approach is *supervised* and requires the preexistence of such similarity links. Here, we use the results of our rule-based method presented in Chapter 4 in a "knowledge graph as silver standard" perspective [132]. However, results from other automatic matching approaches or a manual alignment by an expert could also provide the needed similarity links.

We choose to use GCNs to compute node embeddings since we believe they are well-adapted to a matching task. Indeed, they compute the embedding of a node by considering the embeddings of its neighbors in the graph. Hence, nodes having similar neighborhoods will have similar embeddings, which corresponds to a structural and relational matching approach. Existing works use GCNs to match nodes with the similar assumption that similar nodes have similar neighborhoods [130, 170]. However, on the contrary of these two approaches (see Subsection 2.3.2), we discard literals and use the *Soft Nearest Neighbor loss* [63] to consider all positive and negative examples instead of sampling with the Pairwise Hinge loss[30]. Additionally, such structural and relational matching is well-suited to our application in pharmacogenomics (PGx). Indeed, PGx relationships are reified as nodes whose neighborhood is formed by the involved drugs, genetic factors, and phenotypes (for example, see Figure 4.1 on page 74). Our task of matching PGx relationships thus reduces to matching the nodes resulting from their reification. Since similar

[30]GCNs and the Soft Nearest Neighbor loss are further detailed in Subsection 5.2.2

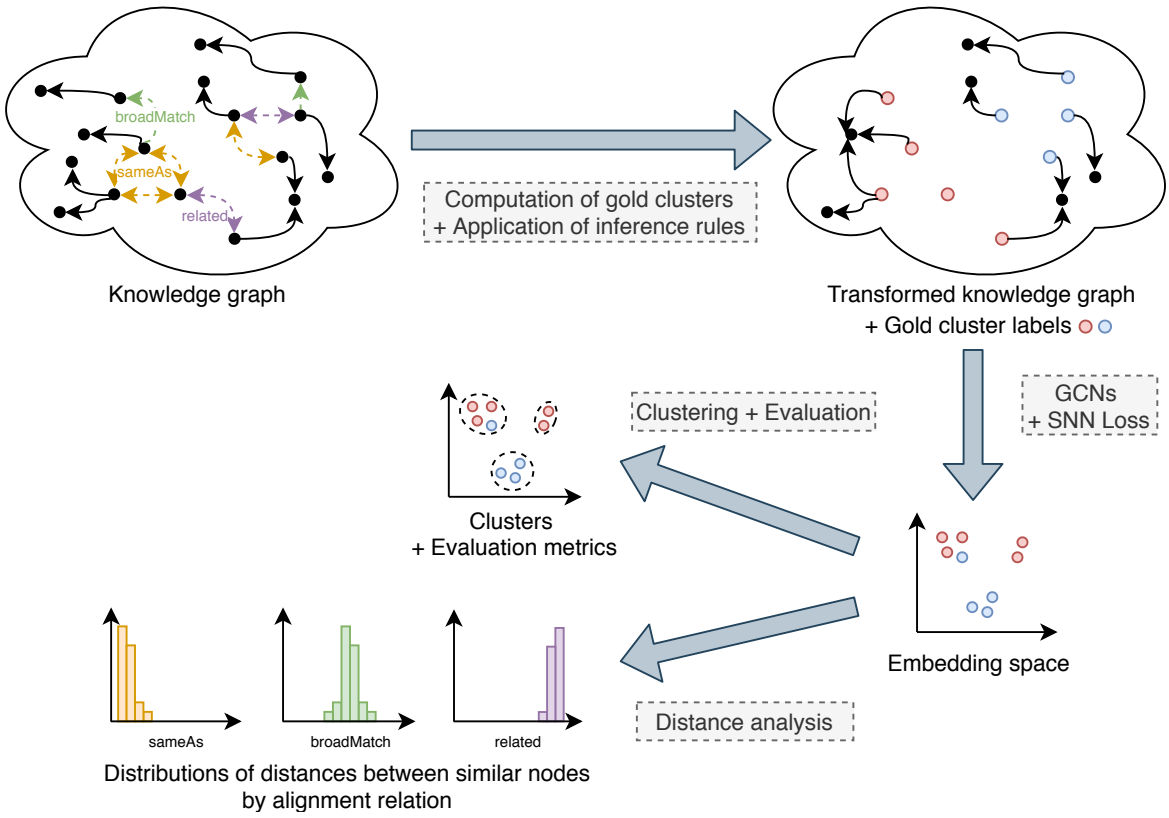


Figure 5.1: Outline of our approach. Gold clusters are computed from existing *similarity links* in the knowledge graph (*e.g.*, `owl:sameAs`, `skos:broadMatch`, `skos:related`, etc.). These similarity links are then removed and various inferences rules associated with domain knowledge are applied on the knowledge graph. Embeddings of nodes are learned with Graph Convolutional Networks (GCNs) and the Soft Nearest Neighbor (SNN) loss. Clustering algorithms are then applied on the embedding space and the resulting clusters are evaluated with regard to the gold clusters. A distance analysis is also performed for each alignment relation.

PGx relationships have similar neighborhoods, they will have similar embeddings computed using GCNs. Consequently, we suppose that they will belong to same cluster when applying a clustering algorithm on the embedding space.

Previous matching methods using GCNs do not consider domain knowledge and associated inference rules on the contrary of recent works and claims [133, 167]. These papers combining semantics and graph embedding and our preliminary results [120] inspired the present approach where we particularly investigate the two following aspects. First, we measure the improvement in clustering results when considering different inference rules associated with domain knowledge, *e.g.*, hierarchies of classes and predicates, symmetry of predicates, etc. Second, as aforementioned, similarity links may represent different alignment relations. We make our GCN model agnostic to these specific alignment relations holding between similar nodes during learning. However, we found that distances between embeddings of similar nodes are different for each alignment relation and are coherent with the relatedness represented by the relation (*e.g.*, smaller distances for equivalences, larger distances for weak similarities). Such results allow us to think that the model is able to "rediscover" these alignment relations. To the best of our knowledge, our approach is the first one to investigate these aspects in a matching task using GCNs and clustering.

5.2 Matching nodes with Graph Convolutional Networks and clustering

5.2.1 Approach outline

Our approach is outlined in Figure 5.1. It takes as input a knowledge graph $\mathcal{K}$ and a set S of nodes to match, where S is a subset of the nodes of $\mathcal{K}$. To illustrate, in our biomedical application, we only intend to match nodes that represent reified PGx relationships. We discard literals and edges incident to literals from $\mathcal{K}$ and S. Hence, a node is either an entity or a class. We consider that we have at our disposal *gold clusters*, *i.e.*, sets of nodes from S that are already labeled as similar. These gold clusters can have uneven sizes. We propose to match nodes in S as follows:

1. Learn embeddings for all nodes in $\mathcal{K}$ such that nodes in S labeled as similar (*i.e.*, belonging to the same gold cluster) have smaller distances between their embeddings (Subsection 5.2.2).

2. Apply a clustering algorithm on the embedding space and consider nodes belonging to the same cluster as similar (Subsection 5.2.3).

Gold clusters can result from another automatic matching method or a manual alignment by an expert. For example, in Section 6.3, our gold clusters are computed from similarity links semi-automatically obtained with rules manually written by experts (described in Chapter 4). As aforementioned, these similarity links can represent different alignment relations (*e.g.*, equivalence, weak similarity). We further detail in Subsection 5.4.1 how these different relations are taken into account in our experiments.

5.2.2 Learning node embeddings with Graph Convolutional Networks and the Soft Nearest Neighbor Loss

In the following, we adopt the notations and definitions of Schlichtkrull et al. [146]. As such, $\mathcal{R}$ denotes the set of predicates in the considered knowledge graph $\mathcal{K}$. Given a node i and a

predicate $r \in \mathcal{R}$, we denote by $\mathcal{N}_i^r$ the set of nodes reachable from i by an edge labeled by r.

Graph Convolutional Networks (GCNs) can be seen as a message-passing framework of multiple layers, in which the embedding $h_i^{(l+1)}$ of a node i at layer $(l+1)$ depends on the embeddings of its neighbors at level (l), as stated in Equation (5.1).

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right) \tag{5.1}$$

This convolution over the neighboring nodes j of i is computed with a specific weight matrix $W_r^{(l)}$ for each predicate $r \in \mathcal{R}$ and each layer (l). The convolution is regularized by a constant $c_{i,r}$, that can be set for each node and each predicate. Similarly to Schlichtkrull et al. [146], we use $c_{i,r} = |\mathcal{N}_i^r|$. The weight matrix $W_0^{(l)}$ enables a self-connection, *i.e.*, the embedding of i at layer $(l+1)$ also depends on its embedding at layer (l). σ is a non-linear function such as ReLU or tanh.

The number of predicates in $\mathcal{K}$ can lead to a high number of parameters $W_r^{(l)}$ to optimize. That is why we use the basis-decomposition proposed by Schlichtkrull et al. [146]. Hence, each $W_r^{(l)}$ is decomposed as follows:

$$W_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} V_b^{(l)} \tag{5.2}$$

For each level (l), B matrices $V_b^{(l)} \in \mathbb{R}^{d^{(l+1)}\times d^{(l)}}$ and $|\mathcal{R}| \times B$ coefficients $a_{rb}^{(l)} \in \mathbb{R}$ are learned, where $d^{(l)}$ and $d^{(l+1)}$ denote the dimension of embeddings at level (l) and level $(l+1)$ respectively. Then, each $W_r^{(l)}$ is computed as a linear combination of matrices $V_b^{(l)}$ and coefficients $a_{rb}^{(l)}$. As only these coefficients depend on predicates r, the number of parameters to learn is reduced.

In our objective of clustering similar nodes, we propose to train GCNs by minimizing the Soft Nearest Neighbor (SNN) loss defined by Frosst et al. [63] and presented in Equation (5.3).

$$\mathcal{L}_{\text{SNN}}(N, Y, T, h) = -\frac{1}{|N|}\sum_{i\in N}\log\left(\frac{\sum\limits_{\substack{j\in N\\ j\neq i\\ Y_i=Y_j}} e^{-\frac{||h_i-h_j||^2}{T}}}{\sum\limits_{\substack{k\in N\\ k\neq i}} e^{-\frac{||h_i-h_k||^2}{T}}}\right) \tag{5.3}$$

The input of the SNN loss consists of:

- A set N of nodes belonging to the gold clusters (see Subsection 5.4.2).
- A set Y of labels for nodes in N. These labels corresponds to the assignments of nodes in N to the gold clusters.
- A temperature T.
- Embeddings h of nodes. These embeddings are the output of the last layer of the GCN model.

Minimizing the SNN loss corresponds to minimizing intra-cluster distances and maximizing inter-cluster distances for the gold clusters of nodes in N. The temperature T determines how

Table 5.1: Clustering algorithms applied on the embeddings of nodes in S. Nodes that belong to the same predicted cluster are considered as similar.

Algorithm	Parameter	Description
Ward	Number of clusters to find	Hierarchical clustering algorithm that successively merges clusters by minimizing the variance of merged clusters
Single	Number of clusters to find	Hierarchical clustering algorithm that successively merges clusters whose distance between their closest observations is minimal
OPTICS [7]	Minimum size of clusters	Algorithm that finds zones of high density and expand clusters from them

distances influence the loss. Indeed, distances between widely separated embeddings are taken into account when T is large whereas only distances between close embeddings are taken into account when T is small. To avoid T as an hyperparameter of the model, we adopt the same learning procedure as Frosst et al. [63]: T is initialized to a predefined value and is optimized by learning $\frac{1}{T}$ as a model parameter.

The computation of $\mathcal{L}_{\text{SNN}}$ (Equation (5.3)) considers all positive and negative examples from N. Indeed, distances between nodes with the same label are minimized (*i.e.*, positive examples) whereas distances between nodes with different labels are maximized (*i.e.*, negative examples). However, it is noteworthy that $\mathcal{K}$ is based on the *Open World Assumption*. Hence, nodes with different labels are regarded as dissimilar (*i.e.*, negative examples) while their (dis)similarity may only be unknown.

5.2.3 Matching nodes by clustering their embeddings

After embeddings of all nodes in the graph have been learned and output by the last layer of GCNs, we perform a clustering on embeddings h_i for all nodes $i \in S$, *i.e.*, all nodes to match. Nodes that belong to the same predicted cluster are considered as similar and these predicted clusters are compared and evaluated with regard to gold clusters.

Here, we experiment with the three distinct clustering algorithms presented in Table 5.1. Their choice was motivated by their availability in scikit-learn [135]. Interestingly, these algorithms differ in their parameters: they take either the number of clusters to find or the minimum size of clusters. This difference allows us to evaluate the influence of inference rules associated with domain knowledge in different settings (see Section 5.3). To compare predicted clusters with gold clusters, we use the three usual metrics presented in Table 5.2.

5.3 Evaluating the influence of applying inference rules associated with domain knowledge

Semantic Web knowledge graphs are represented within formalims such as Description Logics [9] that are equipped with inference rules. Hence, we propose to evaluate the improvements in the results of our matching approach (detailed in Section 5.2) when considering such inference

Table 5.2: Performance metrics used to compare the clusters predicted by the algorithms presented in Table 5.1 with gold clusters.

Metric	Abbr.	Domain	Description
Unsupervised Clustering Accuracy	ACC	$[0, 1]$	Counts nodes whose predicted cluster label is the same as their gold cluster label divided by the total number of nodes. As labels may be permuted between predicted and gold clusters, the mapping with the best ACC is used.
Adjusted Rand Index	ARI	$[-1, 1]$	Considers all pairs of nodes and counts those whose nodes are assigned to the same or different clusters both in predicted and gold clusters. ARI is equal to 0 for a random labeling, and equal to 1 for a perfect labeling (up to a permutation). ARI is adjusted for chance.
Normalized Mutual Information	NMI	$[0, 1]$	Measures the mutual information between the predicted and gold clusters, normalized by the entropy of both types of clusters. NMI is equal to 1 for a perfect labeling (up to a permutation).

rules, independently or combined. Here, we only consider the following logic axioms: class and predicate assertions, equivalence axioms between entities or classes, subsumption axioms between classes or predicates, and axioms defining predicate inverses. Accordingly, we generate six different graphs by running over $\mathcal{K}$ the inference rules associated with these different axioms until saturation. Then, we test our approach on each of these six graphs. These graphs are summarized in Table 5.3 and further described below.

$\mathcal{G}_0$ constitutes the baseline in which no inference rules are run and with the systematic addition of abstract inverses. Indeed, Schlichtkrull et al. [146] consider that for every predicate $r \in \mathcal{R}$, there exists an inverse $r_{\text{inv}} \in \mathcal{R}$. Thus, for every $r \in \mathcal{R}$, we add an abstract inverse $r_{\text{inv}} \in \mathcal{R}$ such that its adjacency matrix represents the inverse of r. This addition of abstract inverses is performed in all other graphs, except when explicitly stated otherwise. $\mathcal{G}_1$ results from the contraction of `owl:sameAs` edges. Indeed, in $\mathcal{K}$, several nodes representing the same entity can co-exist. In this case, they may be linked (directly or indirectly) by `owl:sameAs` edges and should be considered as one, which is enabled by this contraction. In $\mathcal{G}_2$, we do not always add abstract inverses but consider definitions of inverses and symmetry of predicates instead. That is to say:

(i) For a predicate r_1 defined as symmetric (*i.e.*, $r_1 \equiv r_1^{-1}$), we do not add an abstract inverse $r_{1\ \text{inv}}$ and complete its adjacency matrix to ensure its symmetry.

(ii) For a predicate r_2 that has a defined inverse r_3 (*i.e.*, $r_3 \equiv r_2^{-1}$), we do not add an abstract inverse $r_{2\ \text{inv}}$ and complete their adjacency matrices to ensure they represent inverse predicates.

(iii) Otherwise, for a predicate r_4 that neither is symmetric nor have a defined inverse, we add an abstract inverse $r_{4\ \text{inv}}$ such that its adjacency matrix represents the inverse of r_4.

$\mathcal{G}_3$ takes into account the hierarchy of predicates. Indeed, if a predicate r_1 is a subpredicate of r_2 (*i.e.*, $r_1 \sqsubseteq r_2$) and a triple $\langle i, r_1, j\rangle$ exists, then we make sure the triple $\langle i, r_2, j\rangle$ also exists in the graph. This completion is performed by considering the transitive closure of the subsumption relation $\sqsubseteq$. That is to say, if $r_1 \sqsubseteq r_2$ and $r_2 \sqsubseteq r_3$, we also consider $r_1 \sqsubseteq r_3$. Similarly, $\mathcal{G}_4$ completes `type` edges based on the hierarchy of ontology classes defined by `subClassOf` edges. Hence, if $\langle i, \texttt{type}, j\rangle$ and $\langle j, \texttt{subClassOf}, k\rangle$ exist in the graph, then we ensure that $\langle i, \texttt{type}, k\rangle$ is also in the graph. Here again, `subClassOf` edges are considered by computing their transitive closure. Finally, $\mathcal{G}_5$ is the graph resulting from all transformations from $\mathcal{G}_1$ to $\mathcal{G}_4$.

5.4 Experiments

We experimented with the public version of PGxLOD[31]. Our approach is implemented in Python, using PyTorch and the Deep Graph Library for learning embeddings, and scikit-learn for clustering. Our code is available on GitHub[32].

5.4.1 Knowledge graph and gold clusters of similar nodes

We chose to use PGxLOD as the input knowledge graph of our approach since it presents several needed characteristics. First, PGxLOD contains nodes whose matching is well-adapted to a structure-based approach such as ours. Additionally, alignments are expected to be found between these nodes. Indeed, recall that PGxLOD contains 50,435 PGx relationships resulting from:

- an automatic extraction from the reference database PharmGKB;
- an automatic extraction from the biomedical literature;
- a manual representation of 10 studies made from Electronic Health Records of hospitals.

Alignments are expected to be found between such relationships since, for example, PharmGKB is manually curated by experts after a literature review. Recall that PGx relationships are n-ary, and thus they are reified as nodes, as illustrated in Figure 4.1 on page 74 [127]. Hence, nodes representing these relationships form our set S of nodes to match. The reification process entails that neighbors of such nodes are the drugs, genetic factors, and phenotypes involved in the relationships. Consequently, similar relationships have similar neighborhoods, which makes a structure-based approach such as ours well-adapted for their matching.

Second, PGxLOD contains `owl:sameAs` edges (or equivalence axioms), which makes possible the transformation represented in $\mathcal{G}_1$. Indeed, PGxLOD integrates several Linked Open Data sets: ClinVar, DrugBank, SIDER, DisGeNET, PharmGKB, and CTD. These LOD sets contain facts describing components of PGx relationships (*i.e.*, drugs, phenotypes, and genetic factors). Several LOD sets may describe the same entities and we know it explicitly, *i.e.*, some nodes belonging to different LOD sets are linked with `owl:sameAs` edges (see Subsection 3.3.5). For example, this could be the case of a drug represented both in PharmGKB and DrugBank. Thus, we can apply the `owl:sameAs` identification.

Third, PGxLOD contains subsumption axioms between classes and between predicates, which makes possible the transformations represented in $\mathcal{G}_3$ and $\mathcal{G}_4$. Indeed, PGxLOD includes the ATC, MeSH, PGxO, and ChEBI ontologies.

[31] See Chapter 3 – `https://pgxlod.loria.fr`
[32] `https://github.com/pmonnin/gcn-matching`

Table 5.3: Visual summary of the transformations of $\mathcal{K}$ to evaluate the influence of the application of inference rules associated with domain knowledge on node matching. $\mathcal{G}_0$ is the baseline that corresponds to no inference rules being run and the systematic addition of abstract inverses.

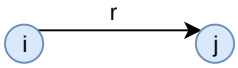
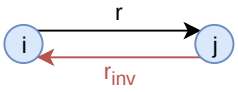
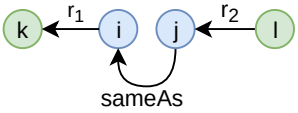
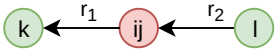
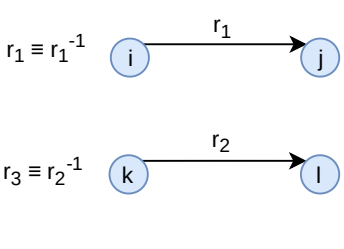
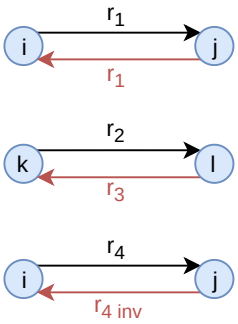
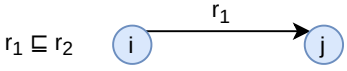
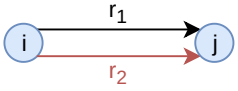
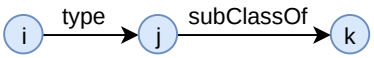
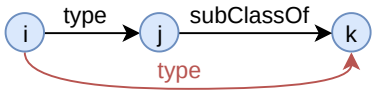
Graph	Before	After
$\mathcal{G}_0$		
$\mathcal{G}_1$		
$\mathcal{G}_2$		
$\mathcal{G}_3$		
$\mathcal{G}_4$		
$\mathcal{G}_5$	All transformations from $\mathcal{G}_1$ to $\mathcal{G}_4$	

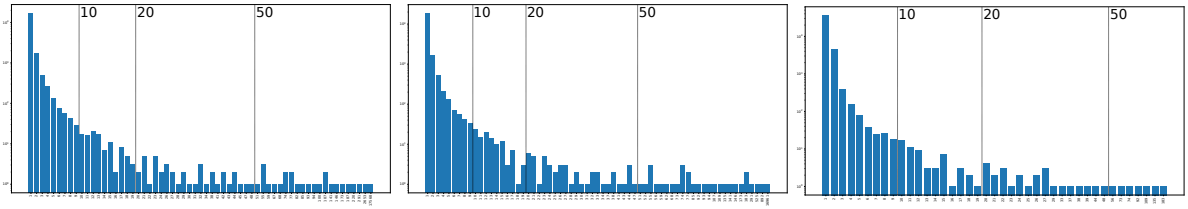
Table 5.4: Alignment relations considered in each gold clustering to compute the gold clusters used in our experiments. We indicate whether a relation is transitive (T or ¬ T) and symmetric (S or ¬ S).

	`owl:sameAs` T / S	`skos:closeMatch` T / S	`skos:relatedMatch` T / S	`skos:related` ¬ T / S	`skos:broadMatch` T / ¬ S
$\mathcal{C}_0$	×	×	×	×	×
$\mathcal{C}_1$	×	×	×	×	
$\mathcal{C}_2$	×				
$\mathcal{C}_3$		×			
$\mathcal{C}_4$			×		
$\mathcal{C}_5$				×	
$\mathcal{C}_6$					×

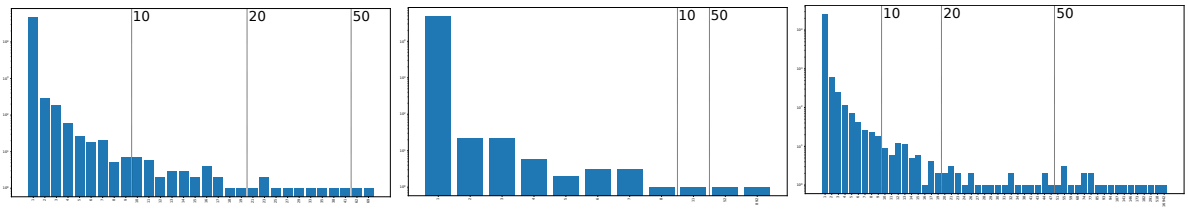
Fourth, some PGx relationships in S are already labeled as similar through similarity links. These links use the five following alignment relations: `owl:sameAs`, `skos:closeMatch`, `skos:relatedMatch`, `skos:related`, and `skos:broadMatch`. Links using `owl:sameAs` and `skos:closeMatch` indicate strong similarities, whereas `skos:relatedMatch` and `skos:related` indicate weaker similarities. Links using `skos:broadMatch` indicate that a PGx relationship is more specific than another. These links result from the application of the matching rules described in Chapter 4 and are removed before running inference rules over $\mathcal{K}$, learning embeddings, and clustering. However, they allow to compute *gold clusters*, *i.e.*, sets of nodes that are considered as similar since they are directly or indirectly connected through similarity links. These gold clusters are used to evaluate our approach in a "knowledge graph as silver standard" perspective [132]. We propose the different *gold clusterings* detailed in Table 5.4. They variously consider the five alignment relations to evaluate our approach in different settings (*e.g.*, all the different alignment relations in $\mathcal{C}_0$, only symmetric relations in $\mathcal{C}_1$, only equivalences in $\mathcal{C}_2$). For each gold clustering, gold clusters correspond to the connected components computed by only considering the (undirected) similarity links of the selected alignment relations between nodes in S. Hence, all alignment relations are regarded as symmetric (undirected links) and transitive (connected components), which is coherent with the majority of alignment relations (see Table 5.4). Figure 5.2 presents the sizes of the resulting gold clusters. We notice that many gold clusters have a size lower or equal to 10, and that considering `skos:related` or `skos:broadMatch` links increases the maximal size of gold clusters. The availability of all these different alignment relations also allows to perform the distance analysis described in Subsection 5.4.4 and indicated in Figure 5.1.

5.4.2 Learning node embeddings

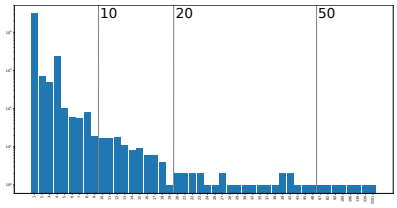
We experimented our approach with different pairs $(\mathcal{C}_i, \mathcal{G}_j)$ that were selected for their experimental interest. All gold clusterings were experimented with graphs $\mathcal{G}_0$ and $\mathcal{G}_5$ to have a global view of the impact on performance of applying inference rules associated with domain knowledge. All graphs were experimented with $\mathcal{C}_0$ to have a finer evaluation of each inference rule on the most heterogeneous gold clustering. For each experimented pair $(\mathcal{C}_i, \mathcal{G}_j)$, a 5-fold cross-validation was performed as follows. For each $\mathcal{C}_i$, S is split into five sets S^i_k ($k \in \{1, 2, 3, 4, 5\}$). All S^i_k contain the same number of nodes for each gold cluster of $\mathcal{C}_i$ larger than 5 nodes. Each



(a) $\mathcal{C}_0$ (max = 17,568) (b) $\mathcal{C}_1$ (max = 16,961) (c) $\mathcal{C}_2$ (max = 183)



(d) $\mathcal{C}_3$ (max = 69) (e) $\mathcal{C}_4$ (max = 892) (f) $\mathcal{C}_5$ (max = 16,942)



(g) $\mathcal{C}_6$ (max = 2,501)

Figure 5.2: Number of gold clusters (y-axis) by size (x-axis) for each gold clustering. The max value is the maximum size of gold clusters (in terms of number of nodes). The minimum size is 1 for every gold clustering. Only gold clusters larger than 10, 20, and 50 nodes are later used to compute performance metrics. Gold clusterings are defined in Table 5.4.

Table 5.5: Statistics of PGxLOD and its transformations as described in Section 5.3. Statistics for PGxLOD discard literals and edges incident to literals. As we use a 3-layer architecture, statistics for all $\mathcal{G}_i$ only consider neighboring nodes up to 3 hops of nodes in S (*i.e.*, PGx relationships to match). # denotes "number of".

	# nodes	# edges	# predicates
PGxLOD	11,808,396	43,341,712	416
$\mathcal{G}_0$	3,758,814	39,956,844	689
$\mathcal{G}_1$	3,879,081	46,960,365	733
$\mathcal{G}_2$	3,758,814	22,085,701	347
$\mathcal{G}_3$	3,758,814	41,048,190	697
$\mathcal{G}_4$	3,758,928	42,691,984	701
$\mathcal{G}_5$	3,882,945	27,277,789	375

set S_k^i is successively used as the test set S_{test}, while set $S_{(k+1)}^i$ is used as the validation set S_{val}[33]. Remaining sets form the train set S_{train}.

An architecture formed by 3 GCN layers is used to learn node embeddings. The input layer consists in a featureless approach as in [92, 146], *i.e.*, the input is just a one-hot vector for each node of the graph. All three layers have an output dimension of 16. Therefore, output embeddings for all nodes in the knowledge graph are in $\mathbb{R}^{16}$. The activation function used on the input and hidden layers is tanh while the output layer uses a linear function. We use a basis-decomposition of 10 bases and set $c_{i,r} = |\mathcal{N}_i^r|$ for all i and all r. In such a 3-layer architecture, it follows from Equation (5.1) that only neighboring nodes up to 3 hops[34] of nodes in S will have an impact on their embeddings, output at layer 3. Thus, to save memory, we reduce graphs to such 3-hop neighborhoods. Statistics about these reduced graphs are available in Table 5.5.

Only the embeddings of nodes in S (here, the PGx relationships) are considered in our clustering task. Hence, only these embeddings are constrained in the SNN loss. However, in $\mathcal{L}_{\text{SNN}}$ (Equation (5.3)), each node needs at least one other node assigned to the same gold cluster (*i.e.*, having the same label). Thus, only gold clusters of size greater or equal to 10 are used in the learning process since each S_k^i contains at least 2 nodes of these clusters. This is particularly needed for the validation and test losses but we chose to use the same constraint for the train loss for homogeneity. We use the Adam optimizer [91] with a starting learning rate of 0.01. T is initialized to 1. We learn during 200 epochs with an early-stopping mechanism: if the validation loss does not decrease of 0.0001 after 10 epochs, the learning process is stopped.

5.4.3 Clustering

Clustering algorithms are only applied on the embeddings of nodes in S_{test} since they are the nodes we aim to match. Recall that the learning process only considers nodes belonging to gold clusters whose size is greater or equal to 10. Accordingly, we apply the three clustering algorithms introduced in Table 5.1 and evaluate their performance on embeddings of nodes in S_{test} that belong to gold clusters whose size is greater or equal to 50, 20, and 10. These different sizes

[33] S_1^i is the validation set when $S_{\text{test}} = S_5^i$.

[34] The 3-hop neighborhood of a node n consists of all the nodes that can be reached with a breadth-first traversal that starts at n and traverses at most 3 edges.

allow to evaluate the influence of inference rules in the performance of our matching approach when considering only large or all gold clusters.

Results on all gold clusterings and graphs $\mathcal{G}_0$ and $\mathcal{G}_5$ are displayed in Table 5.6, Table 5.7 and Table 5.8. In these tables, gray cells indicate the best results among clustering algorithms given a gold clustering, a graph, and a metric. For example, in Table 5.6, considering $\mathcal{C}_0$ and $\mathcal{G}_0$, the best ACC is obtained with the Single clustering algorithm. Underlined values indicate the best result between $\mathcal{G}_0$ and $\mathcal{G}_5$ given a gold clustering and a metric. For example, in Table 5.6, given $\mathcal{C}_1$, the best NMI for Ward is obtained with $\mathcal{G}_0$ whereas the best ACC is obtained with $\mathcal{G}_5$. We notice that applying all inference rules (*i.e.*, $\mathcal{G}_5$) generally increases performance for $\mathcal{C}_0$ and $\mathcal{C}_1$ whereas results for the other gold clusterings do not show such an homogeneous and important increase in performance.

Results on $\mathcal{C}_0$ and all graphs are displayed in Table 5.9, Table 5.10, and Table 5.11. In these tables, gray cells indicate the best result among clustering algorithms and underlined values indicate the best result between graphs. For example, in Table 5.9, given $\mathcal{G}_0$, the best ACC is obtained with the Single clustering algorithm. Given the Single algorithm, the best ARI is obtained with $\mathcal{G}_3$ and $\mathcal{G}_5$. Here again, we notice that applying all inference rules (*i.e.*, $\mathcal{G}_5$) leads to the best results. However, computing all instantiations based on the transitive closure of the subsumption (*i.e.*, $\mathcal{G}_4$) seems to degrade clustering performance.

5.4.4 Distance analysis

During learning and clustering, our model is unaware of the different alignment relations holding between similar nodes. Indeed, the SNN loss only considers labels of gold clusters that do not indicate the alignment relations used to compute these clusters. This is particularly relevant for gold clusterings $\mathcal{C}_0$ and $\mathcal{C}_1$ that mix different alignment relations to compute the gold clusters. However, inspired by our preliminary results [120], we display in Figure 5.3 the distributions of distances between similar nodes in the test set by alignment relation. This analysis is presented for $\mathcal{C}_0$ and graphs $\mathcal{G}_0$ and $\mathcal{G}_5$. Interestingly, similarly to our preliminary results [120], such distributions of distances are coherent with the "strength" of the alignment relations. Indeed, for example, nodes that are weakly similar tend to be further apart than equivalent nodes. Only the `skos:broadMatch` relation presents different distance distributions with regard to the distance distributions of the other relations across the different test sets. This could be explained since this is the only non-symmetric relation (see Table 5.4).

5.5 Discussion and perspectives

Table 5.6, Table 5.7, and Table 5.8 show that performance of clustering are generally better for gold clusterings $\mathcal{C}_2$ to $\mathcal{C}_6$ than $\mathcal{C}_0$ and $\mathcal{C}_1$. Recall that these two gold clusterings mix different alignment relations when computing gold clusters, and thus their matching task is expected to be more difficult. It can also be noticed that performance tends to decrease when considering additional gold clusters (*i.e.*, when decreasing their minimum size). Here again, such a task is more difficult. Indeed, clustering algorithms need to find more clusters (for Ward and Single), or clusters with a reduced minimum size (for OPTICS). However, this is not the case of $\mathcal{C}_2$, $\mathcal{C}_3$, and $\mathcal{C}_4$. This can be explained because, for such gold clusterings, only few gold clusters have a size greater or equal to 50 or 20 (see Figure 5.2), and thus only few training examples are available. Hence, reducing the minimum size leads to consider more training examples, and, despite the task being more difficult, improves performance.

Table 5.6: Results of clustering nodes that belong to gold clusters whose size is greater or equal to 50 for graphs $\mathcal{G}_0$ and $\mathcal{G}_5$. Average and standard deviation for each metric are computed on test folds during a 5-fold cross validation. Given a gold clustering, gray cells indicate the best results among clustering algorithms and underlined values indicate the best result between $\mathcal{G}_0$ and $\mathcal{G}_5$.

		$\mathcal{G}_0$			$\mathcal{G}_5$		
		ACC	ARI	NMI	ACC	ARI	NMI
$\mathcal{C}_0$	Ward	0.24 ± 0.02	0.07 ± 0.01	$\underline{0.37 \pm 0.01}$	$\underline{0.25 \pm 0.02}$	0.07 ± 0.01	0.35 ± 0.01
	Single	0.84 ± 0.08	0.66 ± 0.13	0.59 ± 0.05	$\underline{0.90 \pm 0.00}$	$\underline{0.75 \pm 0.01}$	$\underline{0.64 \pm 0.02}$
	OPTICS	0.61 ± 0.05	0.21 ± 0.08	0.25 ± 0.04	$\underline{0.68 \pm 0.02}$	$\underline{0.27 \pm 0.05}$	$\underline{0.27 \pm 0.02}$
$\mathcal{C}_1$	Ward	0.19 ± 0.02	0.05 ± 0.00	$\underline{0.33 \pm 0.01}$	$\underline{0.20 \pm 0.03}$	0.05 ± 0.01	0.31 ± 0.01
	Single	0.85 ± 0.01	0.55 ± 0.04	0.51 ± 0.03	0.85 ± 0.01	$\underline{0.57 \pm 0.03}$	0.51 ± 0.03
	OPTICS	0.64 ± 0.03	0.19 ± 0.03	0.28 ± 0.01	$\underline{0.71 \pm 0.04}$	$\underline{0.26 \pm 0.06}$	$\underline{0.30 \pm 0.02}$
$\mathcal{C}_2$	Ward	0.88 ± 0.03	0.84 ± 0.03	0.94 ± 0.01	0.88 ± 0.03	0.84 ± 0.03	0.94 ± 0.01
	Single	$\underline{0.88 \pm 0.03}$	$\underline{0.84 \pm 0.03}$	$\underline{0.94 \pm 0.01}$	0.86 ± 0.04	0.81 ± 0.07	0.93 ± 0.02
	OPTICS	$\underline{0.94 \pm 0.06}$	$\underline{0.92 \pm 0.07}$	$\underline{0.97 \pm 0.03}$	0.91 ± 0.06	0.88 ± 0.08	0.95 ± 0.04
$\mathcal{C}_3$	Ward	0.52 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	$\underline{0.53 \pm 0.01}$	0.00 ± 0.00	0.00 ± 0.00
	Single	0.53 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.53 ± 0.01	0.00 ± 0.00	0.00 ± 0.00
	OPTICS	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
$\mathcal{C}_4$	Ward	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Single	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	OPTICS	$\underline{0.45 \pm 0.14}$	-0.02 ± 0.06	0.08 ± 0.08	0.38 ± 0.11	$\underline{0.01 \pm 0.05}$	$\underline{0.11 \pm 0.08}$
$\mathcal{C}_5$	Ward	$\underline{0.20 \pm 0.02}$	$\underline{0.04 \pm 0.00}$	$\underline{0.24 \pm 0.01}$	0.15 ± 0.02	0.03 ± 0.00	0.20 ± 0.01
	Single	0.88 ± 0.00	$\underline{0.31 \pm 0.03}$	$\underline{0.29 \pm 0.03}$	$\underline{0.89 \pm 0.00}$	0.30 ± 0.03	0.27 ± 0.02
	OPTICS	$\underline{0.71 \pm 0.03}$	$\underline{0.18 \pm 0.07}$	$\underline{0.18 \pm 0.04}$	0.68 ± 0.06	0.07 ± 0.07	0.11 ± 0.04
$\mathcal{C}_6$	Ward	0.69 ± 0.02	0.48 ± 0.03	0.64 ± 0.03	$\underline{0.76 \pm 0.12}$	$\underline{0.58 \pm 0.22}$	$\underline{0.67 \pm 0.12}$
	Single	$\underline{0.86 \pm 0.02}$	$\underline{0.60 \pm 0.09}$	$\underline{0.63 \pm 0.08}$	0.82 ± 0.02	0.46 ± 0.12	0.52 ± 0.11
	OPTICS	$\underline{0.59 \pm 0.07}$	$\underline{0.21 \pm 0.09}$	0.44 ± 0.06	0.58 ± 0.05	0.19 ± 0.06	$\underline{0.45 \pm 0.04}$

Table 5.7: Results of clustering nodes that belong to gold clusters whose size is greater or equal to 20 for graphs $\mathcal{G}_0$ and $\mathcal{G}_5$. Average and standard deviation for each metric are computed on test folds during a 5-fold cross validation. Given a gold clustering, gray cells indicate the best results among clustering algorithms and underlined values indicate the best result between $\mathcal{G}_0$ and $\mathcal{G}_5$.

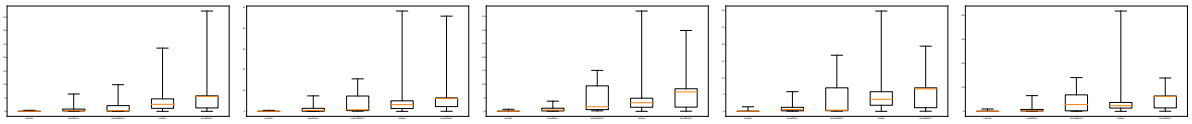
		$\mathcal{G}_0$			$\mathcal{G}_5$		
		ACC	ARI	NMI	ACC	ARI	NMI
$\mathcal{C}_0$	Ward	0.17 ± 0.01	0.04 ± 0.00	$\underline{0.32 \pm 0.01}$	0.17 ± 0.02	0.04 ± 0.00	0.31 ± 0.01
	Single	0.79 ± 0.08	0.64 ± 0.11	0.54 ± 0.05	$\underline{0.86 \pm 0.01}$	$\underline{0.69 \pm 0.01}$	$\underline{0.57 \pm 0.01}$
	OPTICS	0.45 ± 0.03	0.09 ± 0.02	0.17 ± 0.01	$\underline{0.50 \pm 0.01}$	$\underline{0.13 \pm 0.01}$	$\underline{0.19 \pm 0.01}$
$\mathcal{C}_1$	Ward	0.15 ± 0.01	0.03 ± 0.00	$\underline{0.31 \pm 0.01}$	0.15 ± 0.01	0.03 ± 0.00	0.30 ± 0.00
	Single	0.64 ± 0.22	0.38 ± 0.19	0.45 ± 0.06	$\underline{0.82 \pm 0.01}$	$\underline{0.58 \pm 0.03}$	$\underline{0.52 \pm 0.03}$
	OPTICS	0.47 ± 0.02	0.08 ± 0.01	0.20 ± 0.01	$\underline{0.51 \pm 0.02}$	$\underline{0.11 \pm 0.03}$	0.20 ± 0.01
$\mathcal{C}_2$	Ward	0.98 ± 0.00	0.98 ± 0.02	0.99 ± 0.00	0.98 ± 0.00	$\underline{0.99 \pm 0.01}$	0.99 ± 0.00
	Single	0.97 ± 0.03	0.95 ± 0.05	0.98 ± 0.01	$\underline{0.98 \pm 0.00}$	$\underline{0.98 \pm 0.01}$	$\underline{0.99 \pm 0.00}$
	OPTICS	0.69 ± 0.01	0.44 ± 0.04	0.78 ± 0.01	$\underline{0.73 \pm 0.03}$	$\underline{0.48 \pm 0.04}$	$\underline{0.81 \pm 0.02}$
$\mathcal{C}_3$	Ward	$\underline{0.92 \pm 0.06}$	$\underline{0.89 \pm 0.08}$	$\underline{0.95 \pm 0.03}$	0.89 ± 0.05	0.84 ± 0.08	0.93 ± 0.03
	Single	$\underline{0.91 \pm 0.05}$	$\underline{0.87 \pm 0.06}$	$\underline{0.95 \pm 0.03}$	0.88 ± 0.07	0.84 ± 0.09	0.93 ± 0.04
	OPTICS	0.89 ± 0.07	0.87 ± 0.08	0.94 ± 0.08	$\underline{0.92 \pm 0.06}$	$\underline{0.90 \pm 0.09}$	$\underline{0.95 \pm 0.04}$
$\mathcal{C}_4$	Ward	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Single	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.94 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	OPTICS	0.29 ± 0.05	0.01 ± 0.01	0.09 ± 0.02	$\underline{0.34 \pm 0.05}$	$\underline{0.03 \pm 0.01}$	$\underline{0.11 \pm 0.02}$
$\mathcal{C}_5$	Ward	$\underline{0.12 \pm 0.01}$	$\underline{0.02 \pm 0.00}$	$\underline{0.21 \pm 0.01}$	0.10 ± 0.00	0.01 ± 0.00	0.17 ± 0.00
	Single	0.85 ± 0.01	$\underline{0.32 \pm 0.09}$	$\underline{0.28 \pm 0.06}$	$\underline{0.86 \pm 0.00}$	0.20 ± 0.03	0.27 ± 0.02
	OPTICS	0.48 ± 0.02	0.05 ± 0.01	$\underline{0.10 \pm 0.01}$	$\underline{0.52 \pm 0.02}$	0.05 ± 0.02	0.09 ± 0.01
$\mathcal{C}_6$	Ward	$\underline{0.56 \pm 0.05}$	$\underline{0.39 \pm 0.10}$	$\underline{0.67 \pm 0.02}$	0.50 ± 0.06	0.29 ± 0.08	0.65 ± 0.03
	Single	0.64 ± 0.07	0.43 ± 0.13	0.62 ± 0.05	$\underline{0.78 \pm 0.01}$	$\underline{0.67 \pm 0.06}$	$\underline{0.71 \pm 0.03}$
	OPTICS	0.44 ± 0.03	0.08 ± 0.03	$\underline{0.38 \pm 0.02}$	$\underline{0.47 \pm 0.05}$	0.08 ± 0.08	0.37 ± 0.05

Table 5.8: Results of clustering nodes that belong to gold clusters whose size is greater or equal to 10 for graphs $\mathcal{G}_0$ and $\mathcal{G}_5$. Average and standard deviation for each metric are computed on test folds during a 5-fold cross validation. Given a gold clustering, gray cells indicate the best results among clustering algorithms and underlined values indicate the best result between $\mathcal{G}_0$ and $\mathcal{G}_5$.

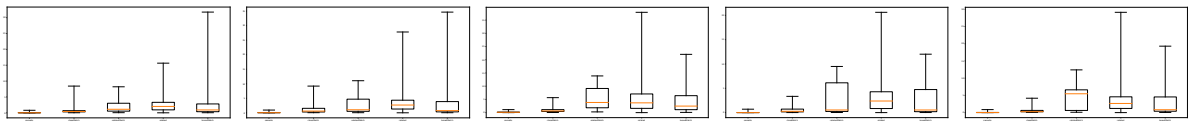
		$\mathcal{G}_0$			$\mathcal{G}_5$		
		ACC	ARI	NMI	ACC	ARI	NMI
$\mathcal{C}_0$	Ward	0.14 ± 0.01	0.02 ± 0.00	0.29 ± 0.01	0.13 ± 0.01	0.02 ± 0.00	0.28 ± 0.02
	Single	0.66 ± 0.17	0.53 ± 0.22	0.52 ± 0.06	0.74 ± 0.15	0.61 ± 0.16	0.54 ± 0.06
	OPTICS	0.25 ± 0.02	0.02 ± 0.01	0.12 ± 0.01	0.27 ± 0.01	0.03 ± 0.01	0.11 ± 0.01
$\mathcal{C}_1$	Ward	0.13 ± 0.01	0.01 ± 0.00	0.28 ± 0.01	0.14 ± 0.01	0.01 ± 0.00	0.27 ± 0.01
	Single	0.41 ± 0.12	0.18 ± 0.07	0.41 ± 0.02	0.72 ± 0.15	0.53 ± 0.14	0.52 ± 0.04
	OPTICS	0.28 ± 0.01	0.02 ± 0.00	0.13 ± 0.01	0.28 ± 0.00	0.02 ± 0.00	0.13 ± 0.01
$\mathcal{C}_2$	Ward	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00
	Single	0.99 ± 0.01	0.99 ± 0.02	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00
	OPTICS	0.63 ± 0.01	0.31 ± 0.01	0.67 ± 0.01	0.62 ± 0.01	0.29 ± 0.01	0.66 ± 0.01
$\mathcal{C}_3$	Ward	0.92 ± 0.00	0.90 ± 0.10	0.94 ± 0.05	0.86 ± 0.04	0.81 ± 0.05	0.89 ± 0.02
	Single	0.90 ± 0.07	0.88 ± 0.12	0.93 ± 0.05	0.83 ± 0.05	0.77 ± 0.08	0.88 ± 0.04
	OPTICS	0.75 ± 0.02	0.58 ± 0.03	0.78 ± 0.02	0.72 ± 0.03	0.49 ± 0.07	0.73 ± 0.05
$\mathcal{C}_4$	Ward	0.99 ± 0.00	0.90 ± 0.07	0.86 ± 0.08	0.99 ± 0.00	0.91 ± 0.05	0.88 ± 0.04
	Single	0.98 ± 0.01	0.83 ± 0.10	0.78 ± 0.15	0.99 ± 0.00	0.88 ± 0.05	0.85 ± 0.07
	OPTICS	0.18 ± 0.03	0.01 ± 0.01	0.06 ± 0.01	0.18 ± 0.01	0.00 ± 0.00	0.06 ± 0.01
$\mathcal{C}_5$	Ward	0.09 ± 0.01	0.01 ± 0.00	0.18 ± 0.01	0.07 ± 0.00	0.01 ± 0.00	0.14 ± 0.01
	Single	0.81 ± 0.01	0.31 ± 0.12	0.25 ± 0.08	0.82 ± 0.01	0.32 ± 0.08	0.26 ± 0.05
	OPTICS	0.27 ± 0.01	0.01 ± 0.00	0.06 ± 0.01	0.28 ± 0.01	0.01 ± 0.01	0.05 ± 0.01
$\mathcal{C}_6$	Ward	0.48 ± 0.03	0.24 ± 0.05	0.64 ± 0.01	0.44 ± 0.02	0.16 ± 0.03	0.60 ± 0.02
	Single	0.63 ± 0.07	0.56 ± 0.14	0.70 ± 0.04	0.74 ± 0.02	0.76 ± 0.05	0.76 ± 0.03
	OPTICS	0.37 ± 0.02	0.02 ± 0.01	0.29 ± 0.02	0.37 ± 0.02	0.03 ± 0.01	0.29 ± 0.01

Table 5.9: Results of clustering nodes that belong to gold clusters whose size is greater or equal to 50 for $\mathcal{C}_0$ and all graphs. Average and standard deviation for each metric are computed on test folds during a 5-fold cross validation. Gray cells indicate the best result among clustering algorithms. Underlined values indicate the best result between graphs. ↓ indicates a lower value with regard to $\mathcal{G}_0$.

		Ward	Single	OPTICS
$\mathcal{G}_0$	ACC	0.24 ± 0.02	0.84 ± 0.08	0.61 ± 0.05
	ARI	0.07 ± 0.01	0.66 ± 0.13	0.21 ± 0.08
	NMI	$\underline{0.37 \pm 0.01}$	0.59 ± 0.05	0.25 ± 0.04
$\mathcal{G}_1$	ACC	0.24 ± 0.02	0.86 ± 0.00 ↓	0.58 ± 0.03
	ARI	0.07 ± 0.01	0.70 ± 0.02 ↓	0.16 ± 0.03
	NMI ↓	0.35 ± 0.02 ↓	0.58 ± 0.02 ↓	0.23 ± 0.01
$\mathcal{G}_2$	ACC	0.24 ± 0.01	0.89 ± 0.02	$\underline{0.70 \pm 0.01}$
	ARI	0.07 ± 0.00	0.72 ± 0.03	$\underline{0.32 \pm 0.03}$
	NMI ↓	0.34 ± 0.01	0.61 ± 0.04	$\underline{0.28 \pm 0.01}$
$\mathcal{G}_3$	ACC ↓	0.22 ± 0.03	0.89 ± 0.02	0.63 ± 0.03
	ARI	0.07 ± 0.01	$\underline{0.75 \pm 0.02}$	0.29 ± 0.04
	NMI ↓	0.36 ± 0.01	0.63 ± 0.03	$\underline{0.28 \pm 0.02}$
$\mathcal{G}_4$	ACC ↓	0.23 ± 0.02 ↓	0.80 ± 0.16	0.62 ± 0.02
	ARI	0.07 ± 0.01 ↓	0.63 ± 0.21 ↓	0.20 ± 0.03
	NMI ↓	0.36 ± 0.01 ↓	0.58 ± 0.08 ↓	0.24 ± 0.01
$\mathcal{G}_5$	ACC	$\underline{0.25 \pm 0.02}$	$\underline{0.90 \pm 0.00}$	0.68 ± 0.02
	ARI	0.07 ± 0.01	$\underline{0.75 \pm 0.01}$	0.27 ± 0.05
	NMI ↓	0.35 ± 0.01	$\underline{0.64 \pm 0.02}$	0.27 ± 0.02



(a) $\mathcal{G}_0$ – Fold 1 (b) $\mathcal{G}_0$ – Fold 2 (c) $\mathcal{G}_0$ – Fold 3 (d) $\mathcal{G}_0$ – Fold 4 (e) $\mathcal{G}_0$ – Fold 5



(f) $\mathcal{G}_5$ – Fold 1 (g) $\mathcal{G}_5$ – Fold 2 (h) $\mathcal{G}_5$ – Fold 3 (i) $\mathcal{G}_5$ – Fold 4 (j) $\mathcal{G}_5$ – Fold 5

Figure 5.3: Distributions of distances between similar nodes by alignment relation for each test set, the $\mathcal{C}_0$ gold clustering and the two graphs $\mathcal{G}_0$ and $\mathcal{G}_5$. In each subpicture, links are from left to right: `owl:sameAs`, `skos:closeMatch`, `skos:relatedMatch`, `skos:related`, and `skos:broadMatch`.

Table 5.10: Results of clustering nodes that belong to gold clusters whose size is greater or equal to 20 for $\mathcal{C}_0$ and all graphs. Average and standard deviation for each metric are computed on test folds during a 5-fold cross validation. Gray cells indicate the best result among clustering algorithms. Underlined values indicate the best result between graphs. ↓ indicates a lower value with regard to $\mathcal{G}_0$.

		Ward	Single	OPTICS
$\mathcal{G}_0$	ACC	0.17 ± 0.01	0.79 ± 0.08	0.45 ± 0.03
	ARI	$\underline{0.04 \pm 0.00}$	0.64 ± 0.11	0.09 ± 0.02
	NMI	$\underline{0.32 \pm 0.01}$	0.54 ± 0.05	0.17 ± 0.01
$\mathcal{G}_1$	ACC	$\underline{0.19 \pm 0.01}$	0.81 ± 0.01 ↓	0.43 ± 0.02
	ARI	$\underline{0.04 \pm 0.00}$	0.64 ± 0.02 ↓	0.07 ± 0.03
	NMI	$\underline{0.32 \pm 0.01}$ ↓	0.52 ± 0.01 ↓	0.16 ± 0.02
$\mathcal{G}_2$	ACC	0.17 ± 0.01	0.81 ± 0.08	0.48 ± 0.01
	ARI	$\underline{0.04 \pm 0.00}$ ↓	0.63 ± 0.09	0.11 ± 0.02
	NMI ↓	0.30 ± 0.01	0.54 ± 0.05	0.17 ± 0.01
$\mathcal{G}_3$	ACC ↓	0.15 ± 0.01	0.81 ± 0.06	0.46 ± 0.01
	ARI ↓	0.03 ± 0.00	0.64 ± 0.12	0.12 ± 0.02
	NMI	$\underline{0.32 \pm 0.01}$	0.55 ± 0.05	0.18 ± 0.01
$\mathcal{G}_4$	ACC	0.17 ± 0.01 ↓	0.69 ± 0.22 ↓	0.43 ± 0.02
	ARI ↓	0.03 ± 0.00 ↓	0.54 ± 0.24 ↓	0.08 ± 0.02
	NMI	$\underline{0.32 \pm 0.01}$ ↓	0.52 ± 0.08	0.17 ± 0.01
$\mathcal{G}_5$	ACC	0.17 ± 0.02	$\underline{0.86 \pm 0.01}$	$\underline{0.50 \pm 0.01}$
	ARI	$\underline{0.04 \pm 0.00}$	$\underline{0.69 \pm 0.01}$	$\underline{0.13 \pm 0.01}$
	NMI ↓	0.31 ± 0.01	$\underline{0.57 \pm 0.01}$	$\underline{0.19 \pm 0.01}$

Table 5.11: Results of clustering nodes that belong to gold clusters whose size is greater or equal to 10 for $\mathcal{C}_0$ and all graphs. Average and standard deviation for each metric are computed on test folds during a 5-fold cross validation. Gray cells indicate the best result among clustering algorithms. Underlined values indicate the best result between graphs. ↓ indicates a lower value with regard to $\mathcal{G}_0$.

		Ward	Single	OPTICS
$\mathcal{G}_0$	ACC	0.14 ± 0.01	0.66 ± 0.17	0.25 ± 0.02
	ARI	0.02 ± 0.00	0.53 ± 0.22	0.02 ± 0.01
	NMI	0.29 ± 0.01	0.52 ± 0.06	$\underline{0.12 \pm 0.01}$
$\mathcal{G}_1$	ACC	$\underline{0.15 \pm 0.01}$	0.73 ± 0.10	0.25 ± 0.01
	ARI	0.02 ± 0.00	0.58 ± 0.13	0.02 ± 0.01
	NMI	$\underline{0.30 \pm 0.01}$ ↓	0.51 ± 0.03	$\underline{0.12 \pm 0.01}$
$\mathcal{G}_2$	ACC ↓	0.12 ± 0.01 ↓	0.62 ± 0.16	$\underline{0.27 \pm 0.01}$
	ARI	0.02 ± 0.00 ↓	0.47 ± 0.19	$\underline{0.03 \pm 0.01}$
	NMI ↓	0.26 ± 0.01 ↓	0.48 ± 0.05 ↓	0.11 ± 0.00
$\mathcal{G}_3$	ACC ↓	0.12 ± 0.00	0.70 ± 0.18	0.26 ± 0.01
	ARI	0.02 ± 0.00	0.58 ± 0.23	$\underline{0.03 \pm 0.01}$
	NMI ↓	0.28 ± 0.01	0.52 ± 0.06	$\underline{0.12 \pm 0.01}$
$\mathcal{G}_4$	ACC	0.14 ± 0.01 ↓	0.56 ± 0.18	0.25 ± 0.01
	ARI	0.02 ± 0.00 ↓	0.42 ± 0.20	0.02 ± 0.00
	NMI	0.29 ± 0.01 ↓	0.50 ± 0.06	$\underline{0.12 \pm 0.00}$
$\mathcal{G}_5$	ACC ↓	0.13 ± 0.01	$\underline{0.74 \pm 0.15}$	$\underline{0.27 \pm 0.01}$
	ARI	0.02 ± 0.00	$\underline{0.61 \pm 0.16}$	$\underline{0.03 \pm 0.01}$
	NMI ↓	0.28 ± 0.02	$\underline{0.54 \pm 0.06}$ ↓	0.11 ± 0.01

Among the considered clustering algorithms, Single generally performs better than the others. For $\mathcal{C}_0$ and $\mathcal{C}_1$, OPTICS is the second best algorithm. For the other gold clusterings, Single and Ward give the best performance. In particular, we notice that OPTICS tends to have a decent ACC but reduced ARI and NMI. As this algorithm is unaware of the number of clusters to find and only knows their minimum size, low ARI and NMI may indicate a different clustering output in terms of both number and size of clusters. Indeed, ARI counts the pairs of nodes that have similar of different assignments both in predicted and gold clusters while NMI measures the mutual information between two different clusterings. On the contrary, ACC counts the number of nodes correctly assigned. Hence, big gold clusters (partially) correctly assigned may increase the ACC value even between different clusterings. Such a situation arises here since some of our gold clusterings lead to gold clusters with numerous nodes. For example, Figure 5.2 shows that a gold cluster in $\mathcal{C}_0$ contains 17,568 nodes.

Table 5.6, Table 5.7, and Table 5.8 allow to compare results between $\mathcal{G}_0$, *i.e.*, no inference rules, and $\mathcal{G}_5$, *i.e.*, all inference rules. It appears that $\mathcal{G}_5$ generally increases performance for $\mathcal{C}_0$ and $\mathcal{C}_1$. Results for the other gold clusterings do not show such an homogeneous and important increase in performance between $\mathcal{G}_0$ and $\mathcal{G}_5$. As aforementioned, $\mathcal{C}_0$ and $\mathcal{C}_1$ mix different alignment relations, which leads to a more difficult matching task. Hence, our results indicate that inference rules associated with domain knowledge provide useful improvements when dealing with heterogeneous similarities and clusters. It is frequent in matching task to consider different alignment relations or "levels" of similarity. Hence, matching approaches could benefit from taking into account inference rules to improve matching results.

Results for $\mathcal{C}_0$ in Table 5.9, Table 5.10, and Table 5.11 detail the clustering performance for each graph transformation. Performances for Ward do not present noticeable modifications. For Single and OPTICS, inference rules seem to mostly improve results, except for $\mathcal{G}_4$. This graph contains all the instantiation links that can be inferred. Consequently, "general" classes are directly linked to entities that instantiate them instead of indirectly. For example, in Table 5.3, k is directly linked to i instead of indirectly. Hence, when computing the embeddings of such entities, embeddings of both general and specific classes are directly considered through the same predicate `type`, which makes difficult for GCNs to weight these classes differently. As specific classes are more important than general classes to discriminate similar and dissimilar nodes, their undifferentiated influence in embeddings may explain the decrease in performance. We notice that $\mathcal{G}_5$ performs best, which advocates for considering all inference rules together. However, based on the degraded performance of $\mathcal{G}_4$ with regard to $\mathcal{G}_0$, one may want to solely focus on inference rules represented by $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$. As expected from the first three tables, Table 5.9, Table 5.10, and Table 5.11 also confirm that Single is the best performing clustering algorithm for $\mathcal{C}_0$, even across the different graph transformations.

Regarding the distance analysis of node embeddings, Figure 5.3 shows that distances between similar nodes are different depending on the alignment relation holding between them. Recall that our GCN model is agnostic to these alignment relations when computing the SNN loss. Interestingly, distances reflect the "strength" of the alignment relations: strong similarities (*i.e.*, `owl:sameAs` and `skos:closeMatch` links) have smaller distances than weaker ones (*i.e.*, `skos:relatedMatch` and `skos:related` links). The `skos:broadMatch` relation appears more difficult to position with regard to others. This can be explained as it is the only alignment relation that is not symmetric. Such coherent distributions of distances seem to indicate the "rediscovery" of alignment relations by GCNs and encourage to consider the distance between embeddings of nodes in a "semantic" way, *i.e.*, smaller distances indicate stronger similarities. Additionally, such different distances also seem to confirm that the neighborhood aggregation of embeddings in GCNs makes them well-suited to a structural and relational matching.

Our results highlight the interest of considering domain knowledge associated with knowledge graphs in embedding approaches and seem to advocate for a further integration of domain knowledge within embedding models. Future works may investigate the same targets with different embedding techniques, whether based on graph neural networks [56] or others (*e.g.*, translational approaches such as TransE). Additionally, we did not use attention mechanisms, which could also consider domain knowledge as in Logic Attention Network [167]. Here, inference rules associated with domain knowledge are used to transform the knowledge graph as a pre-processing operation. However, we could envision to consider such mechanisms directly in the model (*e.g.*, weight sharing between predicates and their super-predicates). Literals could also be taken into account [170]. In a larger perspective, one major future work lies in investigating if and how other semantics than types of similarity links can emerge in the output embedding space.

Chapter 6

Tackling scalability issues in mining path patterns from knowledge graphs

Contents

6.1 Motivation and mining task definition 110
6.2 Towards a scalable approach to mine interesting paths and path patterns 112
6.2.1 Canonicalizing $\mathcal{K}$ 112
6.2.2 Mining interesting neighbors and types 114
6.2.3 Mining interesting paths and path patterns 116
6.2.4 Optional and domain-dependent filtering 122
6.3 Experimental setup 122
6.4 Discussion and perspectives 125

Knowledge graphs are increasing in size, making crucial criteria the scalability and complexity of mining approaches that use them [132]. We presented in Subsection 2.2.2 the central role of knowledge graphs in knowledge discovery tasks. For example, Linked Open Data [17] have been used in all steps of the knowledge discovery process [143]. In particular, features mined from knowledge graphs have been used in multiple applications such as knowledge base completion [64], explanations [131, 164], "semantic" embeddings [133], or fact-checking [150]. The scalability issues that arise when mining features from knowledge graphs constitute the main concern of this chapter. Particularly, we consider a given set of vertices, called *seed vertices*, and focus on mining their associated *neighboring vertices*, *paths*, and, more generally, *path patterns* that involve classes of ontologies linked with knowledge graphs. Due to the combinatorial nature and the increasing size of real-world knowledge graphs, the task of mining these patterns immediately entails scalability issues. We address these issues by proposing a pattern mining approach that relies on a set of constraints (*e.g.*, support or degree thresholds) and the *monotonicity* property. As our motivation comes from the mining of real-world knowledge graphs, we illustrate by experimenting with PGxLOD. This chapter is based on an article published in the international conference ALGOS 2020 [112].

6.1 Motivation and mining task definition

Recall that knowledge graphs expressed using Semantic Web standards [15] can be seen as actual graphs. In this context, *vertices* are either individuals that represent entities of a world (*e.g.*, places, drugs, etc.), literals (*e.g.*, integers, dates, etc.), or classes of individuals (*e.g.*, `Person`, `Drug`, etc.). *Arcs* are defined by triples $\langle$`subject`, `predicate`, `object`$\rangle$ and state that the `subject` is linked to the `object` by a relationship qualified by the `predicate` (*e.g.*, `has-side-effect`, `has-name`, etc.). In this chapter, we view such a knowledge graph as a directed labeled multi-graph $\mathcal{K} = (\Sigma_V, \Sigma_A, V, A, s, t, \ell_V, \ell_A)$, where

- V is the set of vertices.
- A is the set of arcs connecting vertices through predicates[35].
- Σ_V is the set of vertex labels, here, their URI[36].
- Σ_A is the set of arc labels, here, URIs of predicates of $\mathcal{K}$.
- $s : A \to V$ (respectively $t : A \to V$) associates an arc to its source (respectively target) vertex.
- $\ell_V : V \to \Sigma_V$ (respectively $\ell_A : A \to \Sigma_A$) maps a vertex (respectively an arc) to its label.

Hence, a triple $\langle \mathtt{s}, \mathtt{p}, \mathtt{o} \rangle$ is represented by two vertices $v_\mathtt{s}, v_\mathtt{o} \in V$ and an arc $a_{\langle \mathtt{s},\mathtt{p},\mathtt{o} \rangle} \in A$. The source and target vertices of $a_{\langle \mathtt{s},\mathtt{p},\mathtt{o} \rangle}$ are respectively $v_\mathtt{s}$ and $v_\mathtt{o}$, *i.e.*, $s(a_{\langle \mathtt{s},\mathtt{p},\mathtt{o} \rangle}) = v_\mathtt{s}$ and $t(a_{\langle \mathtt{s},\mathtt{p},\mathtt{o} \rangle}) = v_\mathtt{o}$. The labels of $v_\mathtt{s}$, $v_\mathtt{o}$, and $a_{\langle \mathtt{s},\mathtt{p},\mathtt{o} \rangle}$ are respectively `s`, `o`, and `p`, *i.e.*, $\ell_V(v_\mathtt{s}) = \mathtt{s}$, $\ell_V(v_\mathtt{o}) = \mathtt{o}$, and $\ell_A(a_{\langle \mathtt{s},\mathtt{p},\mathtt{o} \rangle}) = \mathtt{p}$.

In this work, we consider the task of mining features from $\mathcal{K}$ that are associated with a set of vertices of interest, which we call *seed vertices*. The set of seed vertices can be defined in intension (*i.e.*, all vertices that instantiate a specified ontology class) or in extension (*i.e.*, by specifying the list of their URIs). For example, in the biomedical domain, an expert may be interested in mining features associated with vertices that represent drugs causing a specific side effect. We propose to mine from $\mathcal{K}$ the three following kinds of features: *neighboring vertices*, *paths*, and *path patterns*.

Neighboring vertices are vertices that can be reached in $\mathcal{K}$ from at least one seed vertex. A neighbor is associated with all seed vertices from which it is reachable. Its *support* counts such seed vertices. For example, in the knowledge graph depicted in Figure 6.1, the neighbor v_6 is reachable from the seed vertices n_1^C and n_2^C, and thus its support is 2.

Paths are sequences of pairs $\xrightarrow{p} e$ that represent an arc labeled by the predicate p incident to an individual e. A path is associated with all seed vertices that root it in $\mathcal{K}$. The *support* of a path counts such seed vertices. For example, the support of $\xrightarrow{p_1} v_2 \xrightarrow{p_2} v_3$ is 1 since only n_1^C root it, *i.e.*, $n_1^C \xrightarrow{p_1} v_2 \xrightarrow{p_2} v_3$ exists in $\mathcal{K}$.

More generally, paths may share several characteristics. For instance, intermediate vertices in paths may instantiate the same ontology classes. We propose to capture these characteristics by considering *path patterns* in addition to paths. Path patterns are sequences of pairs $\xrightarrow{p} E$, where p is a predicate and E is either an individual or a class. Such a pair indicates that an arc labeled by p is incident to *(i)* E if E is an individual or *(ii)* an individual that instantiates E if E is a class. A path pattern is associated with all seed vertices that root a path captured

[35] Here, we discard literals from V and arcs that are incident to literals from A.
[36] Hence, $|\Sigma_V| = |V|$.

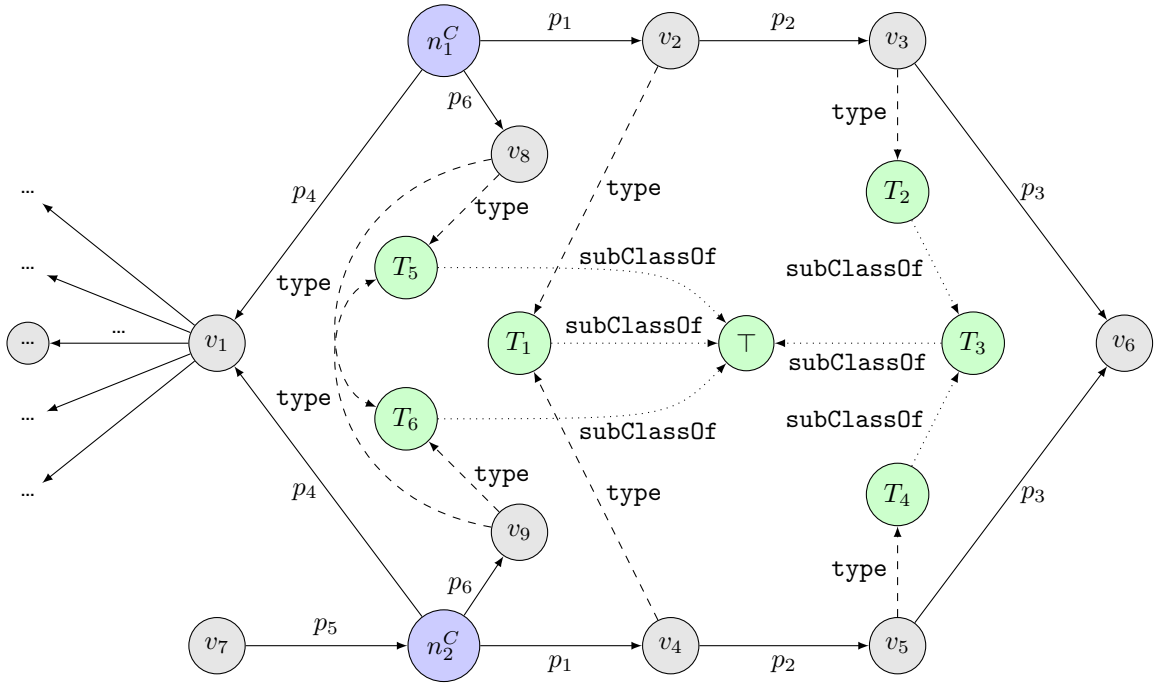


Figure 6.1: Example of a canonical graph $\mathcal{K}^C$. n_1^C and n_2^C are canonical seed vertices, all v_i are canonical individuals, and all T_i are canonical ontology classes. Prefixes of URIs were omitted for readability purposes. The definition of "canonical" is given in Subsection 6.2.1.

by the path pattern. Its *support* counts such seed vertices. In the example graph depicted in Figure 6.1, v_2 instantiates T_1, and v_3 instantiates T_2. Thus, $\xrightarrow{p_1} v_2 \xrightarrow{p_2} v_3$ is captured by $\xrightarrow{p_1} T_1 \xrightarrow{p_2} v_3$, $\xrightarrow{p_1} v_2 \xrightarrow{p_2} T_2$, and $\xrightarrow{p_1} T_1 \xrightarrow{p_2} T_2$. Since T_2 is a subclass of T_3, it is also captured by the pattern $\xrightarrow{p_1} T_1 \xrightarrow{p_2} T_3$. Note that $\xrightarrow{p_1} T_1 \xrightarrow{p_2} T_3$ also captures $\xrightarrow{p_1} v_4 \xrightarrow{p_2} v_5$, which is rooted by n_2^C. Consequently, the support of $\xrightarrow{p_1} T_1 \xrightarrow{p_2} T_3$ is 2. This illustrates the fact that path patterns may capture additional common characteristics of seed vertices, and thus interestingly complete paths. Path patterns have been widely studied in different settings, for example graph rewriting [21] and query answering [11]. We also introduced some related approaches in Subsection 2.2.2.

Mining these patterns constitutes a challenging task due to the combinatorial nature and the size of real-world knowledge graphs, which naturally entail scalability issues. For example, $\xrightarrow{p_1} v_2 \xrightarrow{p_2} v_3$ can be generalized by up to 11 path patterns. This mining task and its inherent scalability issues constitute the main concerns of the present work. To the best of our knowledge, works available in the literature do not address such issues in knowledge graphs with the adopted granular modeling of path patterns. However, inspired by existing graph mining works [2], we propose an Apriori-based approach that alleviates these scalability issues by relying on:

(i) a set of constraints (*e.g.*, support or degree thresholds)

(ii) a pruning of redundant patterns using the hierarchy of ontology classes[37]

(iii) an incremental expansion of paths and patterns

[37]Similarly to works that prune redundant generalized rules [52].

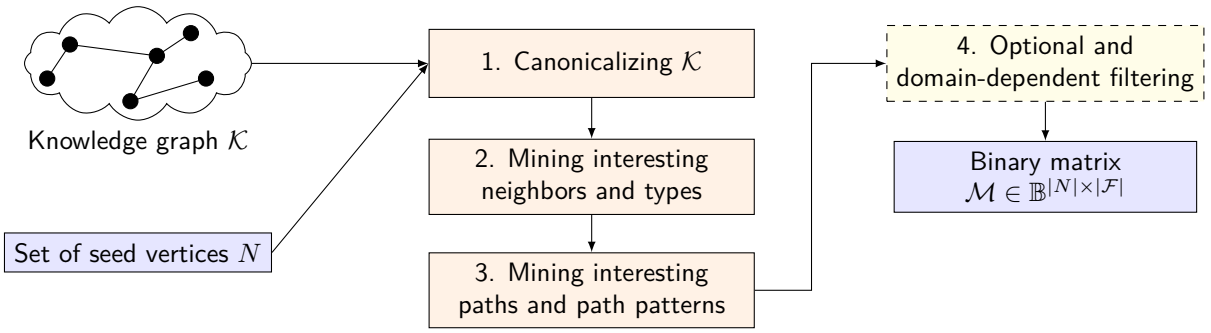


Figure 6.2: Main steps to mine a set $\mathcal{F}$ of features (*i.e.*, neighbors, paths, and path patterns) associated with a set N of seed vertices from a knowledge graph $\mathcal{K}$. Step 4 is optional and depends on the application domain.

(iv) the monotonic character of the support of paths and patterns

We provide a reusable implementation on GitHub[38].

6.2 Towards a scalable approach to mine interesting paths and path patterns

We consider a knowledge graph $\mathcal{K}$ and a set of seed vertices $N = \{n_1, n_2, \ldots, n_p\} \subseteq V$. The task is to mine neighbors, paths, and path patterns from $\mathcal{K}$ that are associated with these seed vertices. For example, given a set of drugs that cause or not a side effect, we aim to mine features that can later be used to classify these drugs.

In the following subsections, we propose algorithms to build a binary matrix $\mathcal{M}$ of size $|N| \times |\mathcal{F}|$ from the knowledge graph $\mathcal{K}$ and the set of seed vertices N. The set $\mathcal{F}$ consists of *interesting* neighbors, paths, and path patterns mined from $\mathcal{K}$, *i.e.*, neighbors, paths, and path patterns that satisfy the constraints defined in terms of the parameters summarized in Table 6.1. These parameters will be detailed in the following subsections. $\mathcal{M}$ associates a seed vertex $n \in N$ with its features $f \in \mathcal{F}$, *i.e.*, if $\mathcal{M}_{n,f} =$ `true`, then n has feature f. We outline our approach in Figure 6.2 where steps 1, 2, and 3 are mandatory, while step 4 is optional and depends on the application domain.

6.2.1 Canonicalizing $\mathcal{K}$

The first step of our approach consists in *canonicalizing* the knowledge graph $\mathcal{K}$, *i.e.*, unifying vertices that represent the same real-world entity. We use the *canonicalization* word by analogy with the canonicalization of knowledge bases, which consists in unifying equivalent individuals into one [64]. Indeed, in knowledge bases under the Open Information Extraction paradigm, facts and entities can be represented by synonymous terms, which leads to co-existing and equivalent individuals. For example, in such knowledge bases, two individuals *Obama* and *Barack Obama* can co-exist. Similarly, in $\mathcal{K}$, vertices can be connected through arcs labeled by the `owl:sameAs` predicate, indicating that these vertices are actually representing the same real-world entity.

[38] `https://github.com/pmonnin/kgpm`

Table 6.1: Parameters that configure the mining of interesting neighbors, paths, and path patterns in a knowledge graph $\mathcal{K}$. Each parameter is associated with a domain and is used in specific steps (see Figure 6.2 for step numbers). Parameter m is specific to the considered application. Here, we illustrate the role of m with the biomedical domain.

Parameter	Domain	Steps	Description
k	$\mathbb{N}^+$	2, 3	Maximum length of paths and path patterns
t	$\mathbb{N}$	3	Maximum level for generalization in class hierarchies
d	$\mathbb{N}$	2, 3	Maximum degree ($u =$ `true`) or out degree ($u =$ `false`) to allow expansion
$l_{\min}$	$\mathbb{N}$	2, 3	Minimum support for features
$l_{\max}$	$\mathbb{N}$	2, 3	Maximum support for features
u	$\mathbb{B}$	2, 3	Whether only out arcs ($u =$ `false`) or all arcs ($u =$ `true`) are traversed
$b_{\text{predicates}}$	List of URIs	2, 3	Blacklist of predicates not to traverse
$b_{\text{exp-types}}$	List of URIs	2, 3	Blacklist of classes whose instances are not to reach
$b_{\text{gen-types}}$	List of URIs	2, 3	Blacklist of classes not to use in generalization
m	{`none`, `p`, `g`, `m`, `pg`, `pgm`}	4	Optional and domain-dependent filtering strategy *Illustrated here with the biomedical domain*

Such a situation typically arises when $\mathcal{K}$ comprises several data sets. For example, a drug can be represented by two vertices linked by an `owl:sameAs` arc, resulting from the information extraction of two independent drug-related databases. Therefore, their merging allows an easy access to the full extent of the knowledge in $\mathcal{K}$ about the drug they represent. Such a canonicalization process corresponds to edge contraction in graph theory (*i.e.*, taking graph quotient). In our framework, it reduces to contracting arcs whose label is the `owl:sameAs` predicate.

To perform this canonicalization, we must respect the semantics associated with the `owl:sameAs` predicate, and thus take into account its symmetry and transitivity. Indeed, an `owl:sameAs` arc between two vertices either is explicitly stated in $\mathcal{K}$ or follows from existing arcs and these two properties. Let us consider a vertex v in $\mathcal{K}$. The canonicalization step merges v with all its identical vertices based on `owl:sameAs` arcs. To compute this set of vertices, it suffices to compute the connected component of v in the undirected spanning subgraph formed by the `owl:sameAs` arcs of $\mathcal{K}$. Indeed, undirected edges comply with the symmetry of `owl:sameAs` and connected components comply with the transitivity of `owl:sameAs`.

As a result, this step takes $\mathcal{K}$ as input and outputs its *canonical* graph $\mathcal{K}^C$. Similarly to $\mathcal{K}$, $\mathcal{K}^C$ is a directed labeled multigraph, *i.e.*, $\mathcal{K}^C = \left(\Sigma_V^C, \Sigma_A^C, V^C, A^C, s^C, t^C, \ell_V^C, \ell_A^C\right)$, where

- V^C is the set of canonical vertices.
- A^C is the set of canonical arcs connecting canonical vertices through predicates.
- Σ_V^C is the set of canonical vertex labels.
- Σ_A^C is the set of canonical arc labels.
- $s^C : A^C \to V^C$ (respectively $t^C : A^C \to V^C$) associates a canonical arc to its canonical source (respectively target) vertex.

- $\ell_V^C : V^C \to \Sigma_V^C$ (respectively $\ell_A^C : A^C \to \Sigma_A^C$) maps a canonical vertex (respectively a canonical arc) to its label.

Each canonical vertex in $\mathcal{K}^C$ represents a vertex from $\mathcal{K}$ and all its identical vertices. It is possible for a canonical vertex in $\mathcal{K}^C$ to only represent one vertex v from $\mathcal{K}$ if v has no identical vertices. This corresponds to creating a surjective mapping $\lambda : V \to V^C$ associating a vertex from $\mathcal{K}$ to its equivalent canonical vertex in $\mathcal{K}^C$. Canonical arcs in $\mathcal{K}^C$ are constructed by using λ to map the source and target vertices of arcs in $\mathcal{K}$ to canonical vertices. Similarly, the set of seed vertices N is mapped to the set of canonical seed vertices, denoted by N^C.

Remark 3. Note that storing URIs has a high memory footprint. Thus, in $\mathcal{K}^C$, we use indices in $\mathbb{N}$ instead of URIs to label vertices and arcs, *i.e.*, $\Sigma_V^C \subseteq \mathbb{N}$ and $\Sigma_A^C \subseteq \mathbb{N}$. This leads to a reduced memory consumption in subsequent algorithms. Each canonical vertex has one unique label, differing from labels of other canonical vertices, *i.e.*, $|\Sigma_V^C| = |V^C|$. This "relabeling" is inspired by the work of de Vries and de Rooij [47] that use a structure named *pathMap* to represent a path by an integer. We developed our own structure for this relabeling, which we named `CacheManager`.

6.2.2 Mining interesting neighbors and types

Mining interesting neighbors

Here, we select all vertices that are neighbors of at least one seed vertex in N^C by performing a breadth-first search constrained by parameters k, d, u, $b_{\text{predicates}}$, and $b_{\text{exp-types}}$. Neighbors are selected by traversing at most k arcs from the seed vertices in N^C. If $u =$ `false`, then only outgoing arcs are traversed; otherwise, all arcs are traversed regardless of their orientation.

However, not all neighboring vertices are of interest. For example, we want to avoid provenance metadata vertices. Indeed, they may not constitute discriminative features as they are specific to the vertex they describe. As we aim to use ontology classes to generate path patterns, we also need to keep the graph exploration over the individuals of $\mathcal{K}$ and avoid traversing `rdf:type` arcs. To this aim, we do not traverse arcs that are labeled by a predicate whose URI or prefix of URI is blacklisted in $b_{\text{predicates}}$. For example, we blacklist in $b_{\text{predicates}}$ the prefix of the provenance ontology PROV-O[39] and the URI of the `rdf:type` predicate[40].

Additionally, we provide a blacklist $b_{\text{exp-types}}$ of URIs or prefixes of classes whose instances must not be reached. Hence, we do not reach individuals that instantiate directly or indirectly a blacklisted class, by following `rdf:type` and `rdfs:subClassOf` arcs. For example, in a use case of classifying drugs that cause or not a side effect, one may want to avoid neighbors that represent the side effect. That is why the ontology class representing the side effect is blacklisted in $b_{\text{exp-types}}$.

When mining neighboring vertices, we may encounter vertices with a high degree, hereafter named *hubs*. If the graph exploration considered their numerous neighbors, then the size of the selected neighborhood would increase exponentially, thus causing a scalability issue. Additionally, hub neighbors may not constitute specific and discriminative features. Indeed, if a hub can be reached from some seed vertices, *i.e.*, appears in their neighborhood, the neighbors of the hub will be reached by the same seed vertices. That is why, in our approach, we propose to stop the graph exploration at vertices whose degree is strictly greater than parameter d[41].

[39] `http://www.w3.org/ns/prov#`

[40] `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`

[41] From a similar assessment, de Vries and de Rooij [48] tackle the hub issue by removing edges based on frequency of pairs (source, predicate) and (predicate, target).

Remark 4. If $u = \texttt{false}$, then the degree of a vertex only counts outgoing arcs, otherwise all arcs are counted. The degree does not count arcs whose predicate is blacklisted in $b_{\text{predicates}}$. The degree counts arcs incident to a vertex that instantiates a blacklisted class in $b_{\text{exp-types}}$.

As a result, with k, d, u, $b_{\text{predicates}}$, and $b_{\text{exp-types}}$ fixed, we obtain a set of neighboring vertices, denoted by $\mathcal{N}(N^C) \subseteq V^C$. Each neighboring vertex $v \in \mathcal{N}(N^C)$ may only appear in the neighborhood of some seed vertices from N^C with regard to the parameters. Thus, v is associated with these seed vertices, which we indicate by defining the *support set* of v.

Definition 10 (Support set of a neighbor)**.** For a given choice of these parameters, the *support set* of a neighboring vertex $v \in \mathcal{N}(N^C)$ is denoted by $\textsc{SupportSet}(v) \subseteq N^C$ and defined as the set of seed vertices from N^C having v as neighbor. The support of a neighbor is defined as the cardinal of its support set.

Note that some vertices in $\mathcal{N}(N^C)$ are not very discriminative: when they are associated with very few vertices from N^C or nearly all of them. This motivates the use of parameters $l_{\min}$ and $l_{\max}$ that define the minimum and maximum support for a neighbor to appear in the set $\mathcal{F}$ of features. Hence, a neighbor $v \in \mathcal{N}(N^C)$ constitutes a feature in the output matrix $\mathcal{M}$ if and only if

$$l_{\min} \leq |\textsc{SupportSet}(v)| \leq l_{\max}.$$

We denote the set of interesting neighbors to appear in $\mathcal{F}$ by

$$\mathcal{N}_l(N^C) = \left\{ v \mid v \in \mathcal{N}(N^C) \text{ and } l_{\min} \leq |\textsc{SupportSet}(v)| \leq l_{\max} \right\}.$$

In the output matrix $\mathcal{M}$, for $n^C \in N^C$ and $v \in \mathcal{N}_l(N^C)$, we have $\mathcal{M}_{n^C,v} = \texttt{true}$ if and only if $n^C \in \textsc{SupportSet}(v)$.

Example 5. From $\mathcal{K}^C$ in Figure 6.1 and with $k = 3$, $d = 4$, $l_{\min} = 2$, $l_{\max} = 3$, $u = \texttt{false}$, $b_{\text{predicates}} = \{\texttt{type}, \texttt{subClassOf}\}$, and $b_{\text{exp-types}} = \emptyset$, we obtain:

$$\begin{aligned}
\mathcal{N}(N^C) &= \{v_1, v_2, v_3, v_4, v_5, v_6, v_8, v_9\}; \\
\textsc{SupportSet}(v_1) = \textsc{SupportSet}(v_6) &= \left\{n_1^C, n_2^C\right\}; \\
\textsc{SupportSet}(v_2) = \textsc{SupportSet}(v_3) = \textsc{SupportSet}(v_8) &= \left\{n_1^C\right\}; \\
\textsc{SupportSet}(v_4) = \textsc{SupportSet}(v_5) = \textsc{SupportSet}(v_9) &= \left\{n_2^C\right\}; \\
\mathcal{N}_l(N^C) &= \{v_1, v_6\}.
\end{aligned}$$

The graph exploration stops at v_1. Indeed, it is considered a hub as its degree is greater than d. Thus, vertices on the left of Figure 6.1 are not explored. Because $u = \texttt{false}$, the graph exploration cannot reach v_7. If $b_{\text{exp-types}} = \{T_3\}$, then v_3 and v_5 cannot be traversed, resulting in $\mathcal{N}(N^C) = \{v_1, v_2, v_4, v_8, v_9\}$.

Mining interesting types

Observe that we can use $\mathcal{N}(N^C)$ to compute interesting types over the considered neighborhood. These interesting types will alleviate a scalability issue arising when building path patterns in Subsection 6.2.3. Interesting types must be computed over $\mathcal{N}(N^C)$ and not $\mathcal{N}_l(N^C)$ as vertices whose support is below $l_{\min}$ can instantiate interesting types. As an intuitive example, in Figure 6.1, T_3 is associated with both n_1^C (because of v_3) and n_2^C (because of v_5). For $l_{\min} = 2$, v_3 and v_5 will not be selected as features, however T_3 can be used in path patterns.

Parameters t and $b_{\text{gen-types}}$ constrain the ontology classes considered in the construction of path patterns, and thus they are integrated in the computation of interesting types. Parameter t specifies the maximum level of considered classes in ontology hierarchies. This level is computed by starting at vertices to generalize and following `rdf:type` and `rdfs:subClassOf` arcs. $t = 0$ only allows to generalize vertices with $\top$, which is considered to be instantiated by all vertices. For example, v_3 can be generalized by T_2 and $\top$ if $t = 1$, and by T_2, T_3, and $\top$ if $t = 2$. Additionally, types used for generalization must not be blacklisted in $b_{\text{gen-types}}$. This blacklist consists of URIs or prefixes of ontology classes not to be used during the construction of path patterns. For example, we refrain from considering general classes such as `pgxo:Drug`[42].

To compute interesting types, we must first compute their support set. This motivates the following predicate:

$$\text{inst}(v, T, t, b_{\text{gen-types}}) = \begin{cases} \texttt{true} \text{ if } v \text{ instantiates } T \text{ under parameters } t \text{ and } b_{\text{gen-types}} \\ \texttt{false} \text{ otherwise} \end{cases}$$

We can then define the support set of an ontology class T as follows:

Definition 11 (Support set of an ontology class)**.** The support set of an ontology class T is the union of support sets of vertices $v \in \mathcal{N}(N^C)$ that can be generalized by T under parameters t and $b_{\text{gen-types}}$. Formally:

$$\textsc{SupportSet}(T) = \bigcup_{\substack{v \in \mathcal{N}(N^C) \\ \text{inst}(v,T,t,b_{\text{gen-types}})}} \textsc{SupportSet}(v)$$

Finally, the set of interesting types used to build path patterns is defined as

$$\mathcal{T}_{\geq l_{\min}} = \{T \mid l_{\min} \leq |\textsc{SupportSet}(T)|\}\,.$$

Example 6. With the same parameters as in Example 5, we obtain

$$\textsc{SupportSet}(T_2) = \left\{n_1^C\right\}; \quad \textsc{SupportSet}(T_1) = \left\{n_1^C, n_2^C\right\}; \quad \mathcal{T}_{\geq l_{\min}} = \{T_1, T_3, T_5, T_6, \top\}\,.$$

6.2.3 Mining interesting paths and path patterns

This step focuses on mining interesting paths and path patterns rooted by seed vertices $n^C \in N^C$. We use the term *path feature* (PF) to indicate a path or a path pattern.

Definition 12 (Path feature)**.** A path feature is a sequence of atomic elements that are pairs $\xrightarrow{p} E$ where p is a predicate and E is either *(i)* an individual (for paths), or *(ii)* an individual or an ontology class (for path patterns). The length of a path feature counts the number of its atomic elements.

Example 7. In Figure 6.1, the path $\xrightarrow{p_1} v_2 \xrightarrow{p_2} v_3 \xrightarrow{p_3} v_6$ can be rooted by n_1^C and is of length 3. The path pattern $\xrightarrow{p_1} T_1 \xrightarrow{p_2} T_3$ can be rooted by n_1^C and n_2^C and is of length 2.

Interesting path features are built by a breadth-first expansion starting at vertices in N^C. As previously, k defines the maximum number of arcs traversed. Hence, path features are of length 1 to k. Observe that a scalability issue arises when mining interesting path features. Indeed,

[42] `http://pgxo.loria.fr/Drug`

Algorithm 6.1 Mining interesting paths and path patterns

Input: The canonical knowledge graph $\mathcal{K}^C$, the set of canonical seed vertices N^C, the set of interesting types $\mathcal{T}_{\geq l_{\min}}$
Parameters: k, t, d, $l_{\min}$, $l_{\max}$, u, $b_{\text{predicates}}$, $b_{\text{exp-types}}$, $b_{\text{gen-types}}$
Output: $\mathcal{F}$ and $\mathcal{M}$ completed with interesting paths and path patterns

1: $h \leftarrow 1$
2: **repeat**
3: Expand paths in $\mathcal{P}_h$
4: Generalize expanded paths into path patterns
5: Keep most specific path features
6: Select *(i)* generated path features to add to $\mathcal{F}$ (complete $\mathcal{M}$ accordingly), and *(ii)* paths to add to $\mathcal{P}_{h+1}$
7: $h \leftarrow h + 1$
8: **until** $h > k$ **or** $\mathcal{P}_h = \emptyset$

there may be several paths between two vertices and each vertex in a path can be generalized by several ontology classes. For instance, for $t = 2$, $\xrightarrow{p_1} v_2 \xrightarrow{p_2} v_3 \xrightarrow{p_3} v_6$ can be generalized by up to 23 path patterns. We propose a mining procedure that alleviates the scalability issues associated with the mining of path patterns. This mining procedure relies on the *monotonicity* of the support set of path features, which is defined as follows:

Definition 13 (Support set of a path feature)**.** The support set of a path consists of all vertices from N^C that root it in $\mathcal{K}^C$. Formally,

$$\textsc{SupportSet}(\xrightarrow{p_a} v_a \dots \xrightarrow{p_b} v_b) = \left\{ n^C \in N^C \mid n^C \xrightarrow{p_a} v_a \dots \xrightarrow{p_b} v_b \text{ exists in } \mathcal{K}^C \right\}.$$

The support set of a path pattern consists of all vertices from N^C that root a path in $\mathcal{K}^C$ that is captured by the path pattern. Formally,

$$\textsc{SupportSet}(\xrightarrow{p_a} E_a \dots \xrightarrow{p_b} E_b) = \Big\{ n^C \in N^C \mid n^C \xrightarrow{p_a} v_a \dots \xrightarrow{p_b} v_b \text{ exists in } \mathcal{K}^C \text{ and } \forall v_i, v_i = E_i \text{ or } \text{inst}(v_i, E_i, t, b_{\text{gen-types}}) \Big\}.$$

The support of a path feature is defined as the cardinal of its support set.

Example 8. From Figure 6.1, we have $\textsc{SupportSet}(\xrightarrow{p_1} v_2 \xrightarrow{p_2} v_3 \xrightarrow{p_3} v_6) = \left\{ n_1^C \right\}$.

Our approach of mining interesting path features is guided by the *dependency structure* as illustrated in Figure 6.3. At first, this structure is empty and it is then augmented at each iteration of Algorithm 6.1, whose operations are described and illustrated below.

Remark 5. As introduced in Remark 3, there are some hacks to help mitigating some of the scalability drawbacks. Storing and manipulating a path feature as a list of elements has a high memory footprint. Thus, such a list is only stored once in our `CacheManager` structure and the returned index (from $\mathbb{N}$) is used in mining algorithms.

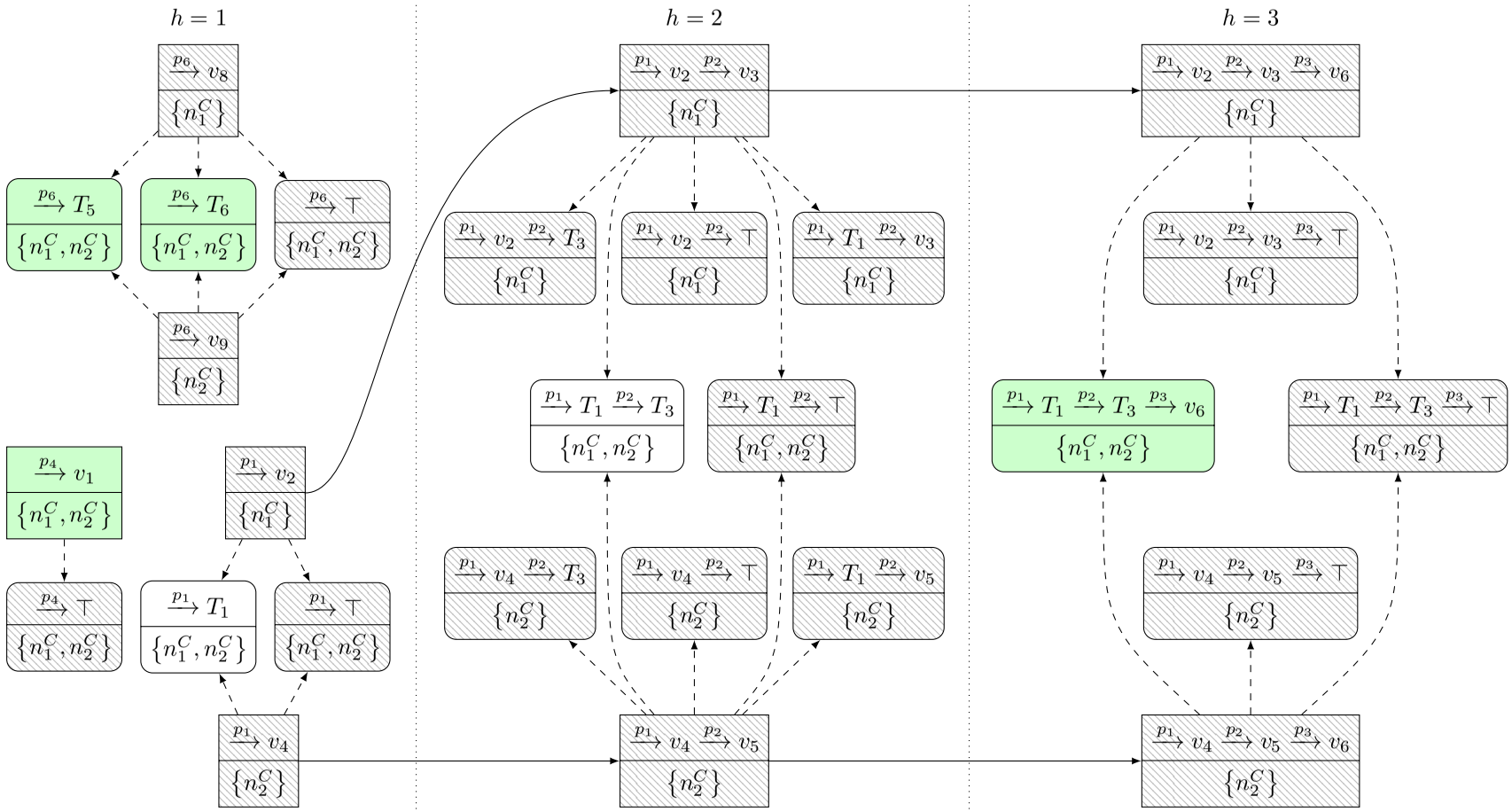


Figure 6.3: Dependency structure used when mining interesting path features from the canonical graph in Figure 6.1. Parameters are $k = 3$, $t = 2$, $d = 4$, $u = \texttt{false}$, $b_{\text{predicates}} = \{\texttt{type}, \texttt{subClassOf}\}$, $b_{\text{exp-types}} = \emptyset$, $b_{\text{gen-types}} = \emptyset$, $l_{\min} = 2$, and $l_{\max} = 3$. Path features are displayed with their support set. Solid arrows represent expansion, dashed arrows represent generalization, rectangles represent paths, and rounded rectangles represent path patterns. Hatched rectangles represent discarded path features because of support limits or more specific path features with identical support set. Green rectangles represent path features ultimately added to $\mathcal{F}$. Blank rectangles represent path features that respect specificity and support constraints but are not in $\mathcal{F}$ because of other features (in green). For readability purposes, path features are displayed as the list of their elements instead of indices from $\mathbb{N}$ actually used to save memory.

Expand paths in $\mathcal{P}_h$

Each path $P \in \mathcal{P}_h$[43] is expanded with pairs $\xrightarrow{p_e} v_e$ that are chosen in the neighborhood of the last individual of P. This choice is constrained by parameters k, d, u, $b_{\text{predicates}}$, and $b_{\text{exp-types}}$ as in Subsection 6.2.2[44]. In the first iteration, the neighborhood of seed vertices in N^C is used. If no path in $\mathcal{P}_h$ can be expanded, *i.e.*, the neighborhood of their last vertex does not contain reachable vertices under the constraints, then Algorithm 6.1 ends.

Example 9. From the graph in Figure 6.1, the first expansion generates the following paths: $\xrightarrow{p_4} v_1$, $\xrightarrow{p_1} v_2$, $\xrightarrow{p_1} v_4$, $\xrightarrow{p_6} v_8$, and $\xrightarrow{p_6} v_9$. In the second expansion, $\xrightarrow{p_4} v_1$ is not expanded as v_1 is a hub under $d = 4$. Since their respective neighborhood does not contain reachable vertices, $\xrightarrow{p_6} v_8$ and $\xrightarrow{p_6} v_9$ are also not expanded. The expansion of $\xrightarrow{p_1} v_2$ generates $\xrightarrow{p_1} v_2 \xrightarrow{p_2} v_3$ whereas the expansion of $\xrightarrow{p_1} v_4$ generates $\xrightarrow{p_1} v_4 \xrightarrow{p_2} v_5$.

Generalize expanded paths into path patterns

Let P_e be the expansion of $P \in \mathcal{P}_h$ as previously described, *i.e.*, $P_e = P \xrightarrow{p_e} v_e$. We generalize P_e by:

- Generating patterns $P \xrightarrow{p_e} T$ for all types $T \in \mathcal{T}_{\geq l_{\min}}$ instantiated by v_e with regard to parameters t and $b_{\text{gen-types}}$, *i.e.*, for all types $T \in \mathcal{T}_{\geq l_{\min}}$ for which the predicate $\text{inst}(v_e, T, t, b_{\text{gen-types}})$ is verified.
- Retrieving from the dependency structure the path patterns that generalize P[45] and expanding them with $\xrightarrow{p_e} v_e$, and $\xrightarrow{p_e} T$ for all $T \in \mathcal{T}_{\geq l_{\min}}$ for which the predicate $\text{inst}(v_e, T, t, b_{\text{gen-types}})$ is verified.

Intuitively, this generalization operation allows to expand path patterns.

Example 10. In the first iteration, $\xrightarrow{p_1} v_2$ is generalized by $\xrightarrow{p_1} T_1$ and $\xrightarrow{p_1} \top$. As we will see, $\xrightarrow{p_1} \top$ is not kept in the dependency structure at the end of the first iteration (see Subsections 6.2.3 and 6.2.3). In the second iteration, $\xrightarrow{p_1} v_2$ expands into $\xrightarrow{p_1} v_2 \xrightarrow{p_2} v_3$. In the dependency structure, we retrieve $\xrightarrow{p_1} T_1$ as the path pattern generalizing $\xrightarrow{p_1} v_2$, which is expanded into $\xrightarrow{p_1} T_1 \xrightarrow{p_2} v_3$, $\xrightarrow{p_1} T_1 \xrightarrow{p_2} T_3$, and $\xrightarrow{p_1} T_1 \xrightarrow{p_2} \top$.

We only generalize paths with types $T \in \mathcal{T}_{\geq l_{\min}}$ to avoid the generation an important number of uninteresting path patterns that would then be discarded, thus reducing the memory footprint. Indeed, by definition, if $T \notin \mathcal{T}_{\geq l_{\min}}$, then $|\textsc{SupportSet}(T)| < l_{\min}$. Additionally, given a path feature P, $|\textsc{SupportSet}(P)| \leq \min_{E \in P} |\textsc{SupportSet}(E)|$, where E can be a class or an individual involved in P. Thus, if $T \notin \mathcal{T}_{\geq l_{\min}}$ is used in a path pattern P, we would have $|\textsc{SupportSet}(P)| < l_{\min}$, therefore generating an uninteresting path pattern that would be discarded later.

Keep most specific path features

Inspired by works that prune redundant generalized rules during their generation [52], we keep only the "most specific" path patterns among those that have the same support set.

[43] $\mathcal{P}_h$ is explained in Subsection 6.2.3.

[44] Additionally, to avoid loops, P can only be expanded at iteration h with individuals v_e such that there exists at least one seed vertex in $\textsc{SupportSet}(P)$ whose shortest distance to v_e is h.

[45] Not all generated path patterns remain in the dependency structure at the end of an iteration, see Subsections 6.2.3 and 6.2.3.

Definition 14 (More specific path pattern)**.** A path pattern P_1 is *more specific* than another path pattern P_2 if every atomic element of P_1 is more specific than the atomic element of P_2 at the same position. An atomic element $\xrightarrow{p_1} E_1$ is more specific than another atomic element $\xrightarrow{p_2} E_2$ if and only if:

(i) $p_1 = p_2$, *i.e.*, both atomic elements involve the same predicate[46], and

(ii) E_1 is more specific than E_2[47].

When path patterns have the same support set, keeping the most specific ones remove redundant generalizations, thus reducing their number and the computational burden. Additionally, we ensure a high descriptive power because the most specific paths are the most descriptive. Intuitively, a path pattern involving a class T is less descriptive than another pattern involving a subclass or an instance of the class. However, keeping the most specific path patterns is computationally expensive, which led us to propose the following computational procedure.

We notice that the support set of a path pattern is the union of the support sets of the paths it generalizes. Therefore, we discard path patterns that generalize only one path. Indeed, such path patterns have the same support set as their original path and are more general, by definition.

Example 11. For $h = 3$, we discard the following path patterns: $\xrightarrow{p_1} v_2 \xrightarrow{p_2} v_3 \xrightarrow{p_3} \top$ and $\xrightarrow{p_1} v_4 \xrightarrow{p_2} v_5 \xrightarrow{p_3} \top$.

However, there may exist path patterns that generalize several more specific path features while having the same support set. Such path patterns should also be discarded.

Example 12. For $h = 1$, $\xrightarrow{p_1} \top$ shares the same support set as the more specific path pattern $\xrightarrow{p_1} T_1$ and thus should be discarded.

To efficiently discard path patterns, we avoid computing their whole hierarchy. Instead, we focus on retaining only the most specific ones in the *prefix tree* depicted in Figure 6.4. This prefix tree is incrementally augmented and stores the most specific path patterns for a specific iteration and support set. In this tree, individuals/classes and predicates involved in path patterns are indexed separately. Thus, its depth is twice the length of path patterns of the current iteration.

The prefix tree enables an efficient storage and selection of the most specific path patterns. Indeed, let P be a path pattern to be compared with those already stored. A breadth-first traversal is performed to detect more specific patterns than P: we only traverse identical or more specific elements according to Definition 14. At any depth, if such elements cannot be found, then P is one of the most specific patterns and the traversal stops. On the contrary, if the traversal reaches a leaf containing more specific elements, then there is a pattern more specific than P with the same support set. Consequently, P is discarded and removed from the dependency structure.

If P is to be stored, another breadth-first traversal is performed by considering identical or more general elements than the ones in P. When the traversal reaches a leaf, it means that more general patterns than P are currently stored and have the same support set. These are removed from the prefix tree before storing P. They are also removed from the dependency structure.

[46]It is noteworthy that we do not consider the hierarchy of predicates in this work.

[47]A class is more specific than all its super-classes and an individual is more specific than all classes it instantiates.

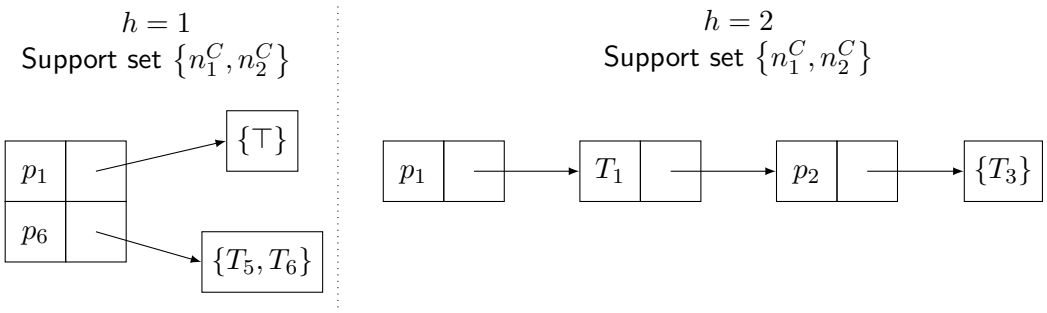


Figure 6.4: Prefix tree used when computing most specific path patterns during the first and second iterations considering the support set $\{n_1^C, n_2^C\}$. This structure is reset at each expansion and for each support set. For $h = 1$, when considering $\xrightarrow{p_1} T_1$, the structure will evolve: $\xrightarrow{p_1} \top$ will be removed and replaced by the considered path. For $h = 3$, when considered, $\xrightarrow{p_1} T_1 \xrightarrow{p_2} \top$ will be discarded as more general than $\xrightarrow{p_1} T_1 \xrightarrow{p_2} T_3$.

Remark 6. As the prefix tree relies on associative arrays and sets, the computational cost of traversal, insertion, and removal is reduced. Additionally, resetting the tree at each expansion and each support set reduces the number of patterns to traverse and thus the global computational cost.

Select generated path features to add to $\mathcal{F}$, and paths to add to $\mathcal{P}_{h+1}$

In this subsection, we determine *(i)* which paths and path patterns should be added as features in $\mathcal{F}$, and consequently, *(ii)* which paths to add to $\mathcal{P}_{h+1}$, *i.e.*, to expand during the next iteration.

A path feature P can be added in $\mathcal{F}$ if:

(C1) $l_{\min} \leq \textsc{SupportSet}(P) \leq l_{\max}$,

(C2) Prefixes of P with the same support set do not already exist in $\mathcal{F}$,

(C3) If P is a path pattern, it does not generalize a path with the same support set.

When P is added to $\mathcal{F}$, the output binary matrix $\mathcal{M}$ is completed such that $\mathcal{M}_{n^C,P} = \texttt{true}$ for all $n^C \in \textsc{SupportSet}(P)$.

Remark 7. Note that (C2) allows to focus on shorter paths in $\mathcal{F}$. However, (C2) is not applied if P ends with an individual and there exist a prefix of P in $\mathcal{F}$ that ends with a class. P is considered more descriptive than its prefix, because of the individual in the last position. Hence, we add P in $\mathcal{F}$ and remove its prefix. For example, in Figure 6.3, $\xrightarrow{p_1} T_1 \xrightarrow{p_2} T_3$ is replaced by $\xrightarrow{p_1} T_1 \xrightarrow{p_2} T_3 \xrightarrow{p_3} v_6$ in $\mathcal{F}$ for $h = 3$.

Remark 8. (C3) is motivated as the path is more specific and thus more descriptive than P and should be added instead of P.

We also select the paths and path patterns to expand during the next iteration. To reduce their number, we rely on the $l_{\min}$ constraint and the *monotonicity* of the support set. It is clear that, for a path feature P, we have $|\textsc{SupportSet}(P)| \leq \min_{E \in P} |\textsc{SupportSet}(E)|$, where E can be a class or an individual involved in P. Thus, when expanding a path feature, its support set remains identical (for paths and path patterns) or decreases (for path patterns).

Consequently, we add to $\mathcal{P}_{h+1}$ paths whose expansion may generate path features complying with the $l_{\min}$ constraint, *i.e.*, paths with a support greater than $l_{\min}$ or paths that are generalized by a pattern with a support greater than $l_{\min}$. This monotonicity property also lets us remove from the dependency structure patterns whose support is lower than $l_{\min}$. Indeed, such patterns cannot be used during the generalization step of the next iteration since they will inevitably generate a pattern whose support is smaller than $l_{\min}$. As a result, the monotonicity property does entail a reduction in the number of paths and path patterns considered in the next iteration, thus reducing the computational cost.

Example 13. For example, in the first iteration, the path $\xrightarrow{p_1} v_2$ is added to $\mathcal{P}_2$ as it is generalized by $\xrightarrow{p_1} T_1$ whose support is greater than $l_{\min} = 2$. At the end of the second iteration, we remove $\xrightarrow{p_1} v_2 \xrightarrow{p_2} T_3$ because its support is lower than $l_{\min} = 2$, and thus its expansion cannot generate an interesting pattern.

6.2.4 Optional and domain-dependent filtering

After the previous steps, we obtain a feature set $\mathcal{F}$ containing interesting neighbors, paths, and path patterns. These features have been mined without taking into account domain constraints known to experts. We propose to apply domain-dependent filtering on $\mathcal{F}$ with parameter m. Such filters reduce the size of $\mathcal{F}$ and integrate interestingness constraints based on expert knowledge.

Example 14. To classify drugs causing or not a side effect, experts may want to focus on features containing a biological pathway, a gene or a GO class, or a MeSH class. Therefore, we propose three atomic filters, only keeping neighbors, paths, and path patterns containing at least a pathway ($m = \texttt{p}$), a gene or a GO class ($m = \texttt{g}$), or a MeSH class ($m = \texttt{m}$). Such atomic filters can be combined to form disjunctive filters. For example, the $m = \texttt{pg}$ filter keeps features from $\mathcal{F}$ containing at least a pathway *or* a gene or a GO class. When a filter is applied to a neighbor, this neighbor must be, *e.g.*, a pathway for the `p` filter. When a filter is applied to a path or a path pattern, it means that one of its individuals / ontology classes must be, *e.g.*, a pathway for the `p` filter.

This domain-dependent filtering is similar to approaches that generalize association rules and prune those that involve some specified ontology classes [52, 102].

6.3 Experimental setup

To illustrate our approach, we will address the following task: from a knowledge graph, mine a set of features to classify drugs depending on whether they cause a specific side effect. We explore PGxLOD[48] [115]. Recall that this knowledge graph aggregates several sets of Linked Open Data (LOD) describing drugs, phenotypes, and genetic factors: PharmGKB, ClinVar, DrugBank, SIDER, DisGeNET, and CTD. This aggregation may lead to features combining units from several LOD sets. Indeed, LOD sets may contain different and incomplete knowledge. Their combined use then enables leveraging a greater amount of knowledge, where some LOD sets complete information provided by others. This asks for a canonical knowledge graph as described in Subsection 6.2.1. For instance, it is possible to complete the knowledge related to a drug described in PharmGKB if it is linked with an `owl:sameAs` arc to the same drug described

[48] See Chapter 3 – `https://pgxlod.loria.fr`

in DrugBank. This constitutes the key interest in combining LOD sets in knowledge discovery and data mining tasks, as discussed by Ristoski and Paulheim [142].

We will use the following data sets that comprise positive ($\oplus$) and negative ($\ominus$) drug examples:

Data set 1 (Drug Induced Liver Injury (DILI) [32]). It is formed by 1,036 drugs in 4 classes:

- "most DILI concern" (192 drugs)
- "ambiguous DILI concern" (254 drugs)
- "less DILI concern" (278 drugs)
- "no DILI concern" (312 drugs)

We mapped these drugs from their PubChem identifiers to identifiers from PharmGKB, otherwise DrugBank, otherwise KEGG, resulting in the set of seed vertices $N^{\text{DILI}} = N^{\text{DILI}}_{\oplus} \cup N^{\text{DILI}}_{\ominus}$ such that:

- $|N^{\text{DILI}}_{\oplus}| = 146$ drugs (118 from PharmGKB, 17 from DrugBank, and 11 from KEGG). The positive drug examples are from the "most DILI concern" class.
- $|N^{\text{DILI}}_{\ominus}| = 224$ drugs (206 from PharmGKB, 9 from DrugBank, and 9 from KEGG). The negative drug examples are from the "no DILI concern" class.

Data set 2 (Severe Cutaneous Adverse Reactions (SCAR)[49]). It is formed by 874 drugs in 5 classes:

- "very probable" (18 drugs)
- "probable" (19 drugs)
- "possible" (94 drugs)
- "unlikely" (697 drugs)
- "very unlikely" (46 drugs)

We mapped these drugs from their PubChem identifiers to identifiers from PharmGKB, otherwise DrugBank, otherwise KEGG, resulting in the set of seed vertices $N^{\text{SCAR}} = N^{\text{SCAR}}_{\oplus} \cup N^{\text{SCAR}}_{\ominus}$ such that:

- $|N^{\text{SCAR}}_{\oplus}| = 102$ drugs (100 from PharmGKB and 2 from DrugBank). The positive drug examples are from the "very probable", "probable", and "possible" classes.
- $|N^{\text{SCAR}}_{\ominus}| = 290$ drugs (286 from PharmGKB and 4 from DrugBank). The negative drug examples are from the "unlikely" and "very unlikely" classes.

We implemented our approach in Python[50]. We used a server with 700 GB of RAM and the following parameter values $k \in \{1, 2, 3, 4\}$, $t \in \{1, 2, 3\}$, $d = 500$, $u = \texttt{false}$, $l_{\min} = 5$, $l_{\max} = +\infty$, and $m \in \{\texttt{p}, \texttt{g}, \texttt{m}, \texttt{pg}, \texttt{pgm}\}$. It should be noted that $k = 4$ was only tested with $t = 1$ because of memory issues caused by the high number of generated features. Statistics about the features are detailed for $k = 3$, $t = 3$ and $k = 4$, $t = 1$ in Table 6.2 and discussed in the next section. We obtained the features associated with the DILI data set under $k = 3$, $t = 3$ in approximately 1 hour using 62 GB of RAM. However, computing the features with $k = 4$, $t = 1$ on the same data set required 4 days and 380 GB of RAM.

[49] `http://www.regiscar.org/`

[50] `https://github.com/pmonnin/kgpm`

Table 6.2: Number of features mined for the two considered data sets for parameters $d = 500$, $u =$ `false` before and after applying $l_{\min} = 5$, $l_{\max} = +\infty$, and $m =$ `pgm`. Path features are always computed considering at least $l_{\min}$ to avoid combinatorial explosion. We display the number of path features generated (gen.) during the exploration and the number of path features ultimately in $\mathcal{F}$ after keeping the most specific and applying limit constraints. Types are not considered as features but are displayed to illustrate the possible combinatorial explosion in path patterns. Statistics about the full neighborhood are given as comparison.

		DILI		SCAR	
		$k = 3$, $t = 3$	$k = 4$, $t = 1$	$k = 3$, $t = 3$	$k = 4$, $t = 1$
Before $l_{\min}$ and $l_{\max}$	Neighbors	175,652	628,681	179,694	639,050
	Types	13,580	18,940	13,677	18,526
After $l_{\min}$ and $l_{\max}$	Neighbors	71,560	281,657	90,690	312,029
	Types	7,372	12,421	8,800	13,177
	Path features in $\mathcal{F}$	790,605	10,169,975	1,146,585	10,729,002
	Path features gen.	20,145,635	251,791,519	29,011,996	255,672,772
After m	Neighbors	4,069	31,804	4,214	33,361
	Path features in $\mathcal{F}$	102,674	2,291,846	147,936	2,400,642
Full neighborhood ($d = 500$)	Neighbors	2,419,957		2,419,920	
	Types	51,477		51,472	
	Reached at k, t	$k = 23$, $t = 21$		$k = 23$, $t = 21$	
Full neighborhood ($d = +\infty$)	Neighbors	5,488,531		5,488,510	
	Types	53,486		53,484	
	Reached at k, t	$k = 19$, $t = 21$		$k = 20$, $t = 21$	

6.4 Discussion and perspectives

The first two lines of Table 6.2 show the number of neighbors and types reachable before applying support limits. Enforcing these limits constitutes a first reduction of these numbers, thus reducing the memory and computational footprints. Here, numbers are approximately divided by 2. The mining of path features always relies on $l_{\min}$, and thus it is not possible to count the number of all possible paths and path patterns. However, we show the number of path features generated during the mining, which already illustrates the combinatorial explosion. Enforcing support constraints and removing redundant generalizations allow to reduce their number in $\mathcal{F}$ (here, approximately by 20). Finally, the domain-dependent filtering defined by m also radically scales down the number of features ultimately output. However, this filtering only happens as post-processing and does not alleviate the scalability issues arising during the mining of patterns.

We observe a drastic increase in the number of neighbors and path features alongside k, which highlights the scalability issues of mining large knowledge graphs. Considering additional levels in ontology hierarchies by increasing t also multiplies the number of path features. To illustrate, with 700 GB of RAM, we could not set t to values greater than 1 for $k = 4$. Consequently, there is still room for improvements in terms of memory consumption, *e.g.*, through an efficient storage of paths and path patterns. These future improvements could enable to consider the full neighborhood of seed vertices, which is reached for greater values of k and t and involves a far greater amount of vertices. For example, for the DILI data set, the full neighborhood is reached for $k = 23$ and $t = 21$ (for $d = 500$), or $k = 19$ and $t = 21$ (when the degree constraint is disabled). The full amount of reachable neighbors is 4 times ($d = 500$) or 9 times greater ($d = +\infty$) than with $k = 4$.

The values of parameters depend on the objectives and domain knowledge of the analyst guiding the mining process, especially for blacklists and support thresholds. However, metrics about the knowledge graph may provide guidance. Indeed, statistics about node degrees can help to find a trade-off between exploration and combinatorial explosion with parameter d. Similarly, the depth of class hierarchies influences the value of t. For example, general classes may not be of interest to the analyst, thus reducing t. The parameter k can be set by considering the graph diameter. As it is common in mining processes, iterations may be required to find the best configuration. Regarding m, it is for now hard-coded and only suitable to some biomedical applications. Inspired by ontologies that allow to interactively define mining workflows [143], we could adapt this parameter to other applications by proposing such an interactive definition.

When manually reviewing the output features, we noticed multiple path features across the aggregated LOD sets. This is made possible by aggregating and canonicalizing multiple LOD sets in the knowledge graph. This result particularly illustrates one of the fundamental aspects of Linked Open Data: the combination of different data sets enables to go beyond their original purposes and coverage. However, it is clear that combining LOD sets leads to bigger knowledge graphs, exacerbating the scalability issues.

Regarding our approach, we only canonicalize vertices, *i.e.*, individuals and ontology classes. Nevertheless, predicates used on arcs can also be identified as identical, leading to a canonicalization of arcs. In this context, we could benefit from matching approaches, such as PARIS [158]. By identifying identical classes, predicates, and individuals, these matching approaches could further improve the canonicalization and, therefore, increase the number of common features between seed vertices from different data sets. Similarly, we could consider literals and arcs incident to literals that were purposely discarded here. However, the canonicalization of literals raises several challenging issues due to their heterogeneity in terms of syntactic variations, unit measures, and the precision of numerical values. Other reasoning mechanisms and semantics

associated with Semantic Web standards could be taken into account. For example, predicates can be defined as transitive, and thus the canonical knowledge graph could also result from their transitive closure.

Regarding the modeling of path patterns, we could generalize paths with both the hierarchy of classes and the hierarchy of predicates, which would exacerbate the scalability issues. In addition to keeping the most specific patterns, we could use other metrics to further reduce their redundancy (*e.g.*, approaches relying on hierarchies [43, 141] and extents of ontological classes [43]). This could also reduce the number of generated patterns, therefore improving the scalability of the mining approach. Neighbors could be enriched with the distance between them and seed vertices, which would correspond to the generalized paths of KGPTree [164]. We could also use other approaches than binary features (*e.g.*, counting [47, 48], relative counting [140]). More importantly, it remains to test our mined features within a complete classification task to measure the influence of k, t, and the three kinds of features (neighbors, paths, and path patterns). Such a classification task naturally appears as a perspective of our running example and our experimentation mining features associated with drugs that cause or not a given side effect. Recall that the features considered in this chapter are interpretable by humans, and thus are useful in explainable approaches [164]. Hence, we also envision to use such features[51] to explain the occurrence of side effects with some drugs [28].

[51] Particularly, the features that are interesting from a biological point of view (*e.g.*, genes, pathways).

Chapter 7

Knowledge graph refinement and mining with Concept Annotation

Contents

7.1 Basics about Concept Annotation 128
7.2 Discovering subsumption axioms between ontology classes 128
7.2.1 Classifying entities in a concept lattice with regard to their outgoing predicates 129
7.2.2 Annotating the concept lattice with classes from the ontology $\mathcal{O}$ 130
7.2.3 Comparing discovered axioms with existing axioms in the ontology $\mathcal{O}$. 132
7.2.4 Preliminary experiment with DBpedia 133
7.3 Suggesting alignments between classes of ontologies 134
7.3.1 Preliminary: a pattern structure for an ontology $\mathcal{O}_i$ 134
7.3.2 Classifying entities with regard to $\mathcal{O}_{\texttt{ref}}$ in a concept lattice 135
7.3.3 Annotating the concept lattice with classes from $\mathcal{O}_1$ and $\mathcal{O}_2$ 136
7.3.4 Reading alignments from the annotated lattice 136
7.4 Discovering domain → range associations for predicates 137
7.4.1 Classifying predicate instantiations in a concept lattice 139
7.4.2 Annotating the concept lattice with $\mathcal{O}_1$ and $\mathcal{O}_2$ 139
7.4.3 Reading domain → range associations from the annotated lattice 140
7.5 Discussion and perspectives 141

In this chapter, we focus on the task of mining knowledge graphs to refine them or discover additional knowledge. Particularly, we refine ontologies associated with knowledge graphs. Recall that, in our work, we consider that ontologies consist of classes and predicates organized in two hierarchies by the subsumption relation. With the increasing number of available data sets following Semantic Web standards and technologies, numerous ontologies co-exist, for example, in ontology repositories such as the NCBO BioPortal [80]. Such ontologies are concurrently published, edited, and instantiated by entities in knowledge graphs. Thus, they may have different qualities and their hierarchical organization may not necessarily be adapted to all their associated knowledge graphs. In this context, we consider the three following refinement tasks:

(i) Discovering subsumption axioms between classes of an ontology;

(ii) Discovering equivalence and subsumption axioms between classes of two (or more) ontologies.

(iii) Discovering domain $\rightarrow$ range associations for predicates, *i.e.*, associations $C_1 \rightarrow C_2$ where C_1 and C_2 are two ontology classes frequently appearing respectively as domain and range of a predicate based on its assertions[52].

Here, we introduce an extension of Formal Concept Analysis that we name Concept Annotation. We use this framework to mine knowledge graphs associated with ontologies to discover regularities that we transform into axioms to enrich or refine ontologies. The work described in this chapter has been presented in an article in the international conference *ISMIS 2017* [116], an article in the international workshop *FCA4AI 2018* [118], and a poster in the national conference *BDA 2017* [117].

7.1 Basics about Concept Annotation

The process of Concept Annotation takes as input a formal lattice, annotates each of its concepts and returned the annotated lattice. The input lattice can result from a standard binary context (G, M, I), where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is the incidence relation [70]. In this case, concepts to annotate are of the form (A, B), where $A \subseteq G$ is a set of objects and $B \subseteq M$ is a set of attributes. Alternatively, it can result from a pattern structure [69] $(G, (D, \sqcap), \delta)$, where G is a set of objects, $(D, \sqcap)$ is a meet semi-lattice of descriptions D, and $\delta : G \rightarrow D$ maps an object g to its description $\delta(g) \in D$. In this case, concepts to annotate are of the form (A, d), where $A \subseteq G$ is a set of objects and $d \in D$ is a description. In both cases, concept annotation considers each concept (A, B) (or (A, d) in the case of a pattern structure) of the input lattice and adds as many dimensions $C_1, \ldots, C_n$ (or *annotations*) as needed to output the resulting *annotated concept* $(A, B, C_1, \ldots, C_n)$ ($(A, d, C_1, \ldots, C_n)$ respectively). Each new dimension is computed by applying a derivation operator either on A or B (respectively d). Accordingly, we define an *annotated lattice* as a lattice where each concept is replaced by its corresponding annotated concept. This annotation procedure allows us to add new elements to the lattice while preserving its original structure. On the contrary, adding such elements in the original context or pattern structure could lead to a different structure for the resulting lattice. In Concept Annotation, the original lattice is considered as a *pivot* structure whose hierarchical organization is preserved but enriched with additional information from which new knowledge is discovered.

In the following sections, we illustrate how Concept Annotation can enable refining domain knowledge represented in ontologies, with the three following use cases:

- Section 7.2: Discovering subsumption axioms between classes of an ontology
- Section 7.3: Suggesting alignments between classes of different ontologies
- Section 7.4: Discovering domain $\rightarrow$ range associations for predicates

7.2 Discovering subsumption axioms between ontology classes

In this section, we consider classes of an ontology $\mathcal{O}$ hierarchically organized by subsumption such as Figure 7.1. These classes are instantiated by entities of a knowledge graph $\mathcal{K}$. An

[52]Such associations could also be seen as generalized association rules.

$\langle$ e_1 , type , C_1 $\rangle$	$\langle$ e_2 , $pred_3$, o_5 $\rangle$
$\langle$ e_1 , type , C_2 $\rangle$	$\langle$ e_3 , type , C_1 $\rangle$
$\langle$ e_1 , $pred_1$, o_1 $\rangle$	$\langle$ e_3 , type , C_2 $\rangle$
$\langle$ e_1 , $pred_2$, o_2 $\rangle$	$\langle$ e_3 , $pred_1$, o_6 $\rangle$
$\langle$ e_2 , type , C_1 $\rangle$	$\langle$ e_4 , type , C_1 $\rangle$
$\langle$ e_2 , type , C_2 $\rangle$	$\langle$ e_4 , type , C_2 $\rangle$
$\langle$ e_2 , type , C_4 $\rangle$	$\langle$ e_4 , type , C_5 $\rangle$
$\langle$ e_2 , type , C_5 $\rangle$	$\langle$ e_4 , $pred_2$, o_7 $\rangle$
$\langle$ e_2 , $pred_1$, o_3 $\rangle$	$\langle$ e_4 , $pred_3$, o_8 $\rangle$
$\langle$ e_2 , $pred_2$, o_4 $\rangle$	

Table 7.1: RDF triples forming an abstract knowledge graph $\mathcal{K}$.

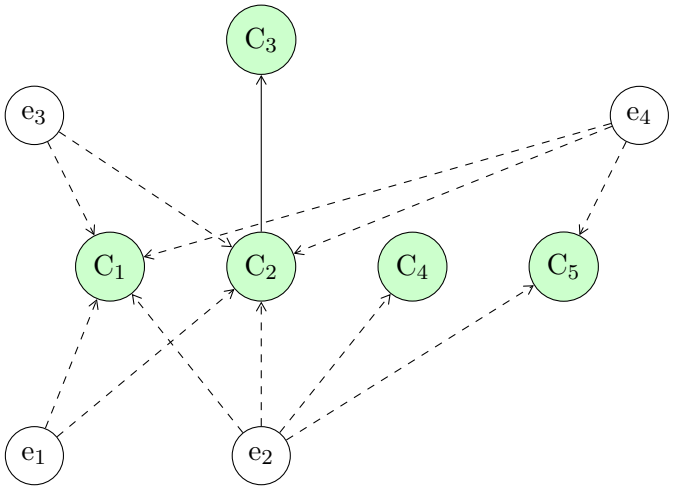


Figure 7.1: Classes (C_1, C_2, C_3, C_4, and C_5 in green) of the ontology $\mathcal{O}$ being instantiated by entities (e_1, e_2, e_3, and e_4). Instantiations are represented using dashed arrows and subsumption relations using solid arrows. For example, e_1 instantiates C_1 and C_2, and C_2 is subsumed by C_3.

example of such triples is depicted in Table 7.1. We propose to hierarchically organize such entities in a lattice using Formal Concept Analysis, based on the regularities in the knowledge graph triples. Each formal concept of the lattice is then annotated by ontology classes, enabling a comparison between the hierarchy of the annotated lattice and the one of the ontology. Such a comparison allows us to confirm existing subsumption axioms or suggest new ones.

7.2.1 Classifying entities in a concept lattice with regard to their outgoing predicates

To hierarchically organize the entities of the knowledge graph with regard to regularities in triples, we build a formal context (G, M, I), where formal objects[53] G are subjects of triples and attributes M are predicates of the same triples. The incidence relation I indicates that there exists at least one triple where a subject and a predicate appear simultaneously. Here, only two elements of the triples are considered: the subject and the predicate, while the object

[53]Here, we use *formal* objects to differentiate them from objects of triples.

	type	pred_1	pred_2	pred_3
e_1	×	×	×	
e_2	×	×	×	×
e_3	×	×		
e_4	×		×	×

Table 7.2: Formal context used to classify entities of a knowledge graph with regard to their predicates. It is built from the RDF triples in Table 7.1. Formal objects (in rows) are subjects of the triples and attributes (in columns) are their outgoing predicates. A cross in the table indicates that there exists at least one triple where the subject and the predicate appear simultaneously.

is not taken into account. Then, we apply standard FCA on this context to build a concept lattice. For example, the context built from the triples in Table 7.1 is depicted in Table 7.2. The resulting concept lattice is displayed in Figure 7.2.

7.2.2 Annotating the concept lattice with classes from the ontology $\mathcal{O}$

Concepts of the lattice in Figure 7.2 are formed by an extent containing subjects and an intent containing predicates. To compare the lattice hierarchical organization with the ontology $\mathcal{O}$, we need to associate formal concepts with ontology classes. To this aim, given $A \subseteq G$ and $\mathcal{C} \subseteq \text{classes}(\mathcal{O})$, we define two new dual derivation operators denoted by $(\cdot)^\diamond : 2^G \to 2^{\text{classes}(\mathcal{O})}$ and $(\cdot)^\diamond : 2^{\text{classes}(\mathcal{O})} \to 2^G$ such that:

$$A^\diamond = \bigcap_{e \in A} \{C \in \text{classes}(\mathcal{O}) \mid \mathcal{K} \models C(e)\} \quad \text{and} \quad \mathcal{C}^\diamond = \{e \in G \mid \forall C \in \mathcal{C},\ \mathcal{K} \models C(e)\}$$

$A^\diamond$ contains the ontology classes instantiated by all the subjects in A. $\mathcal{C}^\diamond$ is the set of subjects instantiating all the ontology classes in $\mathcal{C}$. It is noteworthy that $(\cdot)^{\diamond\diamond} : 2^{\text{classes}(\mathcal{O})} \to 2^{\text{classes}(\mathcal{O})}$ and $(\cdot)^{\diamond\diamond} : 2^G \to 2^G$ are *closure operators*.

Proof. An operator on a partially ordered set is a closure operator if it is monotonous (*i.e.*, $X_1 \subseteq X_2$ implies $X_1^{\diamond\diamond} \subseteq X_2^{\diamond\diamond}$), extensive (*i.e.*, $X \subseteq X^{\diamond\diamond}$) and idempotent (*i.e.*, $X^{\diamond\diamond} = X^{\diamond\diamond\diamond\diamond}$).

First, let us prove that:

$$X_1 \subseteq X_2 \Rightarrow X_2^\diamond \subseteq X_1^\diamond \tag{7.1}$$

Let us consider $E_1 \subseteq G$ and $E_2 \subseteq G$ such that $E_1 \subseteq E_2$. We have:

$$\begin{aligned} E_2^\diamond &= \bigcap_{e \in E_2} \{C \in \text{classes}(\mathcal{O}) \mid \mathcal{K} \models C(e)\} \\ &= (\cap_{e \in E_1} \{C \in \text{classes}(\mathcal{O}) \mid \mathcal{K} \models C(e)\}) \bigcap \left(\cap_{e \in E_2 \setminus E_1} \{C \in \text{classes}(\mathcal{O}) \mid \mathcal{K} \models C(e)\}\right) \\ &= E_1^\diamond \bigcap (E_2 \setminus E_1)^\diamond \end{aligned}$$

It then follows that $E_2^\diamond \subseteq E_1^\diamond$. Dually, let us consider $\mathcal{C}_1 \subseteq \text{classes}(\mathcal{O})$ and $\mathcal{C}_2 \subseteq \text{classes}(\mathcal{O})$ such

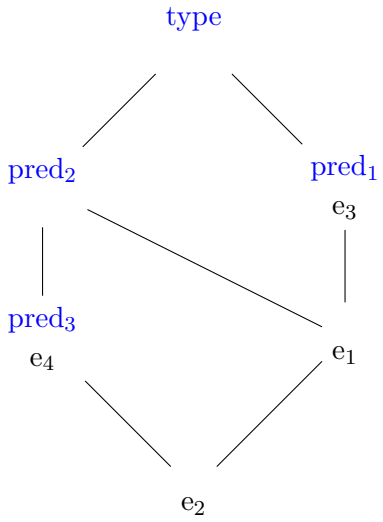


Figure 7.2: Line diagram representing the concept lattice built from the formal context in Table 7.2. The lattice is displayed using the reduced notation. Following this notation, formal objects, depicted in black, are associated with one concept and, implicitly, with all its superconcepts. Attributes, in blue, are associated with one concept and, implicitly, with all its subconcepts.

that $\mathcal{C}_1 \subseteq \mathcal{C}_2$. We have:

$$\begin{aligned}
\mathcal{C}_2^{\diamond} &= \{e \in G \mid \forall C \in \mathcal{C}_2,\ \mathcal{K} \models C(e)\} \\
&= \{e \in G \mid \forall C \in \mathcal{C}_1,\ \mathcal{K} \models C(e) \text{ and } \forall C \in \mathcal{C}_2 \setminus \mathcal{C}_1,\ \mathcal{K} \models C(e)\} \\
&= \{e \in G \mid \forall C \in \mathcal{C}_1,\ \mathcal{K} \models C(e)\} \bigcap \{e \in G \mid \forall C \in \mathcal{C}_2 \setminus \mathcal{C}_1,\ \mathcal{K} \models C(e)\} \\
&= \mathcal{C}_1^{\diamond} \bigcap (\mathcal{C}_2 \setminus \mathcal{C}_1)^{\diamond}
\end{aligned}$$

It then follows that $\mathcal{C}_2^{\diamond} \subseteq \mathcal{C}_1^{\diamond}$. This proves Equation (7.1).

To prove the monotonicity of the operator $(\cdot)^{\diamond\diamond}$, let us consider $X_1 \subseteq X_2$. From Equation (7.1), it follows that $X_2^{\diamond} \subseteq X_1^{\diamond}$, and then $X_1^{\diamond\diamond} \subseteq X_2^{\diamond\diamond}$.

To prove the extensivity of the operator $(\cdot)^{\diamond\diamond}$, let us consider $E \subseteq G$. We have:

$$E^{\diamond} = \bigcap_{e \in E} \{C \in \text{classes}(\mathcal{O}) \mid \mathcal{K} \models C(e)\} \quad \text{and} \quad E^{\diamond\diamond} = \{f \in G \mid \forall C \in E^{\diamond},\ \mathcal{K} \models C(f)\}$$

By definition of $E^{\diamond}$, it is clear that $\forall e \in E,\ \mathcal{K} \models C(e)$ for all $C \in E^{\diamond}$. Therefore, $\forall e \in E,\ e \in E^{\diamond\diamond}$, which means that $E \subseteq E^{\diamond\diamond}$. Dually, let us consider $\mathcal{C} \subseteq \text{classes}(\mathcal{O})$. We have:

$$\mathcal{C}^{\diamond} = \{e \in G \mid \forall C \in \mathcal{C},\ \mathcal{K} \models C(e)\} \quad \text{and} \quad \mathcal{C}^{\diamond\diamond} = \bigcap_{f \in C^{\diamond}} \{C \in \text{classes}(\mathcal{O}) \mid \mathcal{K} \models C(f)\}$$

By definition of $C^{\diamond}$, it is clear that $\forall C \in \mathcal{C},\ \mathcal{K} \models C(f)$ for all $f \in C^{\diamond}$. Therefore, $\forall C \in \mathcal{C},\ C \in \mathcal{C}^{\diamond\diamond}$, which means that $\mathcal{C} \subseteq \mathcal{C}^{\diamond\diamond}$.

To prove the idempotence of the operator $(\cdot)^{\diamond\diamond}$, from the extensivity of operator $(\cdot)^{\diamond\diamond}$, we know that $X \subseteq X^{\diamond\diamond}$. Therefore, from Equation (7.1), $X^{\diamond\diamond\diamond} \subseteq X^{\diamond}$, and $X^{\diamond\diamond} \subseteq X^{\diamond\diamond\diamond\diamond}$. Then, from the extensivity property, we also have $X^{\diamond} \subseteq X^{\diamond\diamond\diamond}$. Therefore, from Equation (7.1), $X^{\diamond\diamond\diamond\diamond} \subseteq X^{\diamond\diamond}$. Hence, $X^{\diamond\diamond} = X^{\diamond\diamond\diamond\diamond}$. □

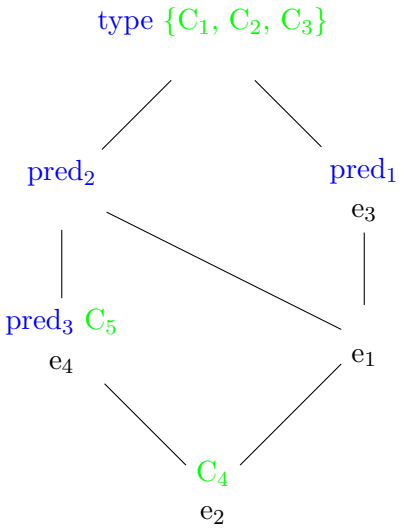


Figure 7.3: Line diagram representing the annotated lattice based on the concept lattice in Figure 7.2 and the ontology in Figure 7.1. It is displayed using our extension of the reduced notation. Formal objects are depicted in black, attributes in blue and concept annotations in green.

To each formal concept (A, B), we add the annotation $A^\diamond$ to form the annotated concept $(A, B, A^\diamond)$. From the definition of the operator $(\cdot)^\diamond$, $A^\diamond$ can be seen as the "common classes" instantiated by subjects in A. For example, in Figure 7.2, let us consider the concept $(\{e_2, e_4\}, \{\text{type}, \text{pred}_2, \text{pred}_3\})$. From the ontology in Figure 7.1, it follows that $\{e_2, e_4\}^\diamond = \{C_1, C_2, C_3, C_5\}$. So, the annotated concept is $(\{e_2, e_4\}, \{\text{type}, \text{pred}_2, \text{pred}_3\}, \{C_1, C_2, C_3, C_5\})$. Given two concepts such that $(A_1, B_1) \leqslant (A_2, B_2)$, as $A_1 \subseteq A_2$, it follows from Equation (7.1) that $A_2^\diamond \subseteq A_1^\diamond$. Therefore, we extend the reduced notation to represent an annotated lattice: extents and intents are depicted as usual and annotations are depicted showing only the new classes that are not already in the annotations of the superconcepts (following the same principle as for attributes in reduced notation). For example, the annotation of the lattice in Figure 7.2 results in the annotated lattice depicted in Figure 7.3. The annotation of the annotated concept $(\{e_2, e_4\}, \{\text{type}, \text{pred}_2, \text{pred}_3\}, \{C_1, C_2, C_3, C_5\})$ only shows C_5 as C_1, C_2, and C_3 are already in the annotation of its superconcepts.

7.2.3 Comparing discovered axioms with existing axioms in the ontology $\mathcal{O}$

Subsumption axioms between classes can be discovered from the annotated lattice resulting from the previous step. Indeed, let us consider an annotated concept $(A, B, A^\diamond)$. We denote $A_p^\diamond$ the *proper* annotation of $(A, B, A^\diamond)$, *i.e.*, the set of classes in the annotation that are not inherited from the annotations of its superconcepts. For example, the proper annotation of the annotated concept $(\{e_1, e_2, e_3, e_4\}, \{\text{type}, \text{pred}_1, \text{pred}_2, \text{pred}_3\}, \{C_1, C_2, C_3, C_4, C_5\})$ in Figure 7.3 is $\{C_4\}$. Let us consider one of the covering annotated superconcept $(E, F, E^\diamond)$ of $(A, B, A^\diamond)$. Similarly, we denote $E_p^\diamond$ its proper annotation. For example, we can consider $(\{e_2, e_4\}, \{\text{type}, \text{pred}_2, \text{pred}_3\}, \{C_1, C_2, C_3, C_5\})$ whose proper annotation is $\{C_5\}$. For all the pairs $(C_i, C_j) \in A_p^\diamond \times E_p^\diamond$, C_i is instantiated by all the subjects in A and C_j is instantiated by all the subjects in E. As $A \subseteq E$, C_j is also instantiated by all subjects in A and is instantiated by

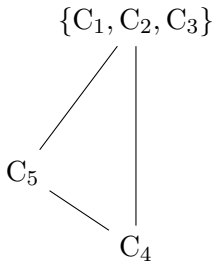


Figure 7.4: Line diagram representing the induced order on proper annotations from the annotated lattice in Figure 7.3.

more subjects than C_i. Thus, we consider this result as the discovery of a subsumption axiom $C_i \sqsubseteq C_j$. For example, from the two previously considered annotated concepts, we discover $C_4 \sqsubseteq C_5$.

If we only considered the covering superconcepts, the discovery process would miss some interesting axioms. Indeed, for example, in Figure 7.3, the proper annotation of $(\{e_1, e_2, e_4\}, \{type, pred_2\}, \{C_1, C_2, C_3\})$ is empty as $\{C_1, C_2, C_3\}$ are in the annotation of its covering superconcept. Therefore, we propose to discover subsumption axioms from the order on annotations induced by the annotated lattice. It is obtained by ordering annotations by set inclusion following the order between formal concepts. For example, the order on annotations induced by the annotated lattice in Figure 7.3 is depicted using the reduced notation in Figure 7.4. Set inclusion of annotations is read from top to bottom. Thus, subsumption axioms can be read from this reduced notation from bottom to top. In our running example, we discover the following axioms:

$$C_5 \sqsubseteq C_1;\ C_5 \sqsubseteq C_2;\ C_5 \sqsubseteq C_3$$

$$C_4 \sqsubseteq C_5;\ C_4 \sqsubseteq C_1;\ C_4 \sqsubseteq C_2;\ C_4 \sqsubseteq C_3$$

Discovered axioms $C_i \sqsubseteq C_j$ are then compared with axioms defined in the ontology $\mathcal{O}$:

(i) if $C_i \sqsubseteq C_j$ is already explicitly stated in $\mathcal{O}$, this is a *confirmed axiom*;

(ii) if $C_i \sqsubseteq C_j$ is not already stated, but can be inferred, this is an *inferable axiom*;

(iii) if $C_i \sqsubseteq C_j$ is neither already explicitly stated nor inferable, it is a *new subsumption axiom*.

In our example, the discovered axiom $C_4 \sqsubseteq C_5$ is a new axiom.

7.2.4 Preliminary experiment with DBpedia

We performed a preliminary experiment with DPpedia [99] (2016-04 version). Among others, we selected the three following data sets:

- All subjects instantiating `dbo:Person` and having a death date between January 1st 2000 and January 7th 2000;
- All subjects instantiating `dbo:FloweringPlant`;
- All subjects instantiating `dbo:Beverage`.

As entities in DBpedia are associated with classes of several ontologies, we performed the annotation process twice: once with classes of the DBpedia Ontology, and once with classes of YAGO.

As an example, the lattice resulting from the `dbo:Person` data set was formed by 15,234 concepts. After annotating with classes from the DBpedia ontology, we found 11 confirmed axioms and no inferable or new axioms. After annotating with classes from YAGO, we found 199 confirmed axioms, 2,250 inferable axioms, and 1,372 new axioms. Over the three considered data sets, some new axioms were particularly interesting. For example, we discovered the two following axioms:

$$\texttt{yago:FilipinoChildActors} \sqsubseteq \texttt{yago:FilipinoActors}$$

$$\texttt{yago:WikicatRedWineGrapeVarieties} \sqsubseteq \texttt{yago:WikicatGrapeVarieties}$$

These two axioms should be considered for addition in YAGO as they seem valid in a general case. On the contrary, some other axioms, such as $\texttt{yago:WikicatTreesOfEcuador} \sqsubseteq \texttt{yago:WikicatMedicinalPlants}$ are only locally valid, *i.e.* they seem to make sense in the considered data set but not in a general case. Here, this axiom indicates that, in the considered data set, all trees of Ecuador are also medicinal plants. It should be noted that one can find such locally valid axioms interesting in a use case of *describing* local regularities or identifying missing entities (here, trees of Ecuador that are not medicinal plants). We noticed that some of the discovered axioms introduced "cycles", for example:

$$\texttt{dbo:Organization} \sqsubseteq \texttt{dbo:Organisation} \quad \text{and} \quad \texttt{dbo:Organisation} \sqsubseteq \texttt{dbo:Organization}$$

Such cycles can be interpreted as an equivalence between involved classes, here:

$$\texttt{dbo:Organization} \equiv \texttt{dbo:Organisation}$$

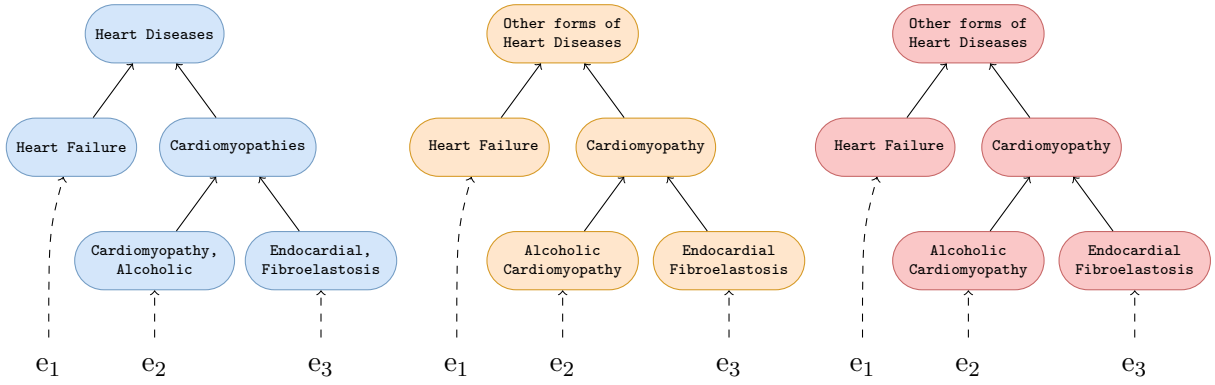
Similarly to discovered subsumption axioms, some of these equivalences seem to make sense in a general case while some others are only locally valid.

7.3 Suggesting alignments between classes of ontologies

In this section, we propose to apply Concept Annotation, as described in Section 7.1, to suggest alignments between classes of ontologies. This use case interestingly illustrates that Concept Annotation can also be applied on the lattice resulting from a pattern structure. Here, we aim at suggesting equivalences between classes of two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. To generate such axioms, we consider the entities of a knowledge graph $\mathcal{K}$ that may instantiate classes of both $\mathcal{O}_1$ and $\mathcal{O}_2$. These entities also instantiate classes of another ontology, $\mathcal{O}_{\texttt{ref}}$. The latter is considered as the reference ontology, *i.e.*, the feature we use to build the original lattice that organizes the instances hierarchically. As a running example for this section, we use the entities and ontologies depicted in Figure 7.5.

7.3.1 Preliminary: a pattern structure for an ontology $\mathcal{O}_i$

Here, as a preliminary, we define a pattern structure leveraging the taxonomy of an ontology $\mathcal{O}_i$. This pattern structure will be used in the remainder of this chapter. Given two classes $C_1, C_2 \in \text{classes}(\mathcal{O}_i)$, their *least common subsumer* (sometimes named their *lowest common ancestor*) is the most specific class subsuming both C_1 and C_2. We denote it by $\text{lcs}(C_1, C_2)$. In



(a) $\mathcal{O}_{\texttt{ref}}$ (MeSH) (b) $\mathcal{O}_1$ (ICD-9-CM) (c) $\mathcal{O}_2$ (ICD-10-CM)

Figure 7.5: Entities (e_1, e_2, and e_3) of a knowledge graph $\mathcal{K}$, instantiating classes of $\mathcal{O}_{\texttt{ref}}$ (MeSH), $\mathcal{O}_1$ (ICD-9-CM), and $\mathcal{O}_2$ (ICD-10-CM). Dashed arrows represent instantiations of classes by entities and solid arrows represent subsumption relations between classes.

$\mathcal{EL}$ ontologies where no cycle appears, the lcs of two classes always exists [10]. Here, we consider that $\mathcal{O}_{\texttt{ref}}$, $\mathcal{O}_1$, and $\mathcal{O}_2$ are $\mathcal{EL}$ ontologies.

We propose to use the pattern structure $(G, (2^{\text{classes}(\mathcal{O}_i)}, \sqcap_i), \delta_i)$. G is a set of entities of the knowledge graph $\mathcal{K}$. The function $\delta_i : 2^G \to 2^{\text{classes}(\mathcal{O}_i)}$ associates an entity from G with the set of the most specific classes of the ontology $\mathcal{O}_i$ that this entity instantiates. Formally, given $g \in G$, $\delta_i(g) = \text{msc}\{C \in \text{classes}(\mathcal{O}_i) \mid \mathcal{K} \models C(g)\}$[54]. $\delta_i(g)$ is considered as the description of the entity g. Given two entities, $g_1, g_2 \in G$, we define the similarity operator $\sqcap_i$ to compare their two descriptions as follows:

$$\delta_i(g_1) \sqcap_i \delta_i(g_2) = \text{msc}\{\text{lcs}(C_1, C_2) \mid \forall (C_1, C_2) \in \delta_i(g_1) \times \delta_i(g_2)\}$$

By using the *least common subsumer*, we are able to compute the most specific classes of $\mathcal{O}_i$ that both g_1 and g_2 instantiate. Finally, given a set of objects $A \subseteq G$, we define the derivation operator $(\cdot)^{\circ_i} : 2^G \to 2^{\text{classes}(\mathcal{O}_i)}$ as:

$$A^{\circ_i} = \prod_{g \in A}{}_{\text{i}}\, \delta_i(g)$$

7.3.2 Classifying entities with regard to $\mathcal{O}_{\texttt{ref}}$ in a concept lattice

As previously, the first step of the annotation process consists in building a concept lattice classifying the considered entities.

Here, we consider a classification of the entities with regard to the classes of $\mathcal{O}_{\texttt{ref}}$ they instantiate. To this aim, we use the pattern structure $(G, (2^{\text{classes}(\mathcal{O}_{\texttt{ref}})}, \sqcap_{\texttt{ref}}), \delta_{\texttt{ref}})$ defined according to the formalism from Subsection 7.3.1. From this pattern structure, we obtain pattern concepts (A, D) organized in a concept lattice, where $A \subseteq G$ is a set of entities from $\mathcal{K}$ and $D \in 2^{\text{classes}(\mathcal{O}_{\texttt{ref}})}$ is the set of the most specific classes from $\mathcal{O}_{\texttt{ref}}$ that all the entities in A instantiate. For example, the entities instantiating the classes of $\mathcal{O}_{\texttt{ref}}$ depicted in Figure 7.5a are organized in the concept lattice in Figure 7.6.

[54] The msc operator is defined in Subsection 4.2.2.

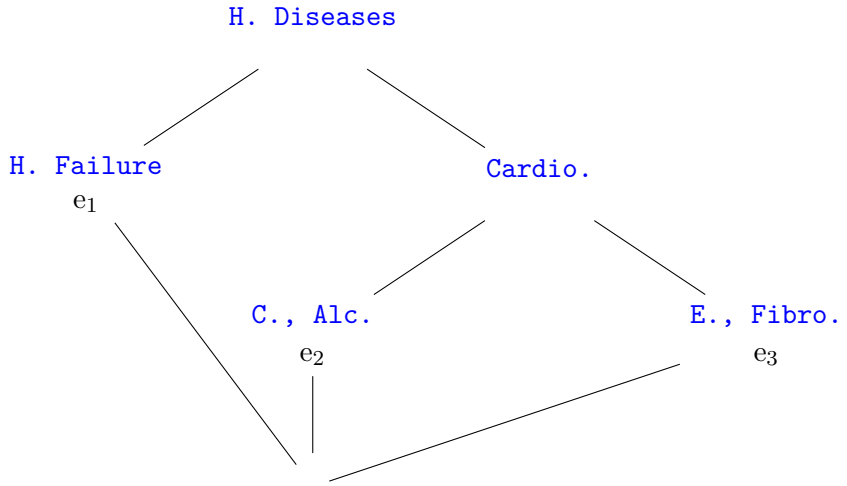


Figure 7.6: Line diagram representing the lattice resulting from classifying entities of $\mathcal{K}$ with regard to the classes of $\mathcal{O}_{\texttt{ref}}$ they instantiate (Figure 7.5a). Formal objects are depicted in black and descriptions in blue (classes from $\mathcal{O}_{\texttt{ref}}$).

7.3.3 Annotating the concept lattice with classes from $\mathcal{O}_1$ and $\mathcal{O}_2$

Next, once the concept lattice is built, it is annotated with classes from $\mathcal{O}_1$ and $\mathcal{O}_2$. To this aim, we define *two* annotations (one per ontology) using the formalism from Subsection 7.3.1.

Thereby, we use two functions δ_1 and δ_2 to associate an entity g with the set of the most specific classes from $\mathcal{O}_1$ and $\mathcal{O}_2$ that this entity instantiates. $\sqcap_1$ and $\sqcap_2$ are used to compute the similarity between the descriptions of two entities with regard to the two considered ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. Finally, to compute the two annotations for each pattern concept (A, D), two derivation operators $(\cdot)^{\circ_1} : 2^G \to 2^{\text{classes}(\mathcal{O}_1)}$ and $(\cdot)^{\circ_2} : 2^G \to 2^{\text{classes}(\mathcal{O}_2)}$ are defined as follows:

$$A^{\circ_1} = \bigsqcap_{g \in A}{}_1 \, \delta_1(g) \quad \text{and} \quad A^{\circ_2} = \bigsqcap_{g \in A}{}_2 \, \delta_2(g)$$

As a result, for each pattern concept (A, D), an annotated pattern concept $(A, D, A^{\circ_1}, A^{\circ_2})$ is obtained where A°_1} (respectively A°_2}) is the set of the most specific classes of $\mathcal{O}_1$ (respectively $\mathcal{O}_2$) that all the individuals in A instantiate. For example, the annotation of the lattice in Figure 7.6 by classes of $\mathcal{O}_1$ and $\mathcal{O}_2$ (Figures 7.5b and 7.5c) results in the annotated lattice depicted in Figure 7.7.

7.3.4 Reading alignments from the annotated lattice

After the annotation process, one can read suggested alignments between ontology classes directly from the annotated lattice.

Indeed, let us consider an annotated pattern concept $(A, D, A^{\circ_1}, A^{\circ_2})$. From the previous definitions, $A^{\circ_1} \subseteq \text{classes}(\mathcal{O}_1)$ contains the set of the most specific classes of $\mathcal{O}_1$ that all the entities in A instantiate. Similarly, $A^{\circ_2} \subseteq \text{classes}(\mathcal{O}_2)$ contains the set of the most specific classes of $\mathcal{O}_2$ that all the entities in A instantiate. Therefore, considering each pair of classes $(C_1, C_2) \in A^{\circ_1} \times A^{\circ_2}$, we know that they are instantiated by the same set of entities, *i.e.*, entities in A. Therefore, an equivalence relationship is suggested between C_1 and C_2. Additionally, it is

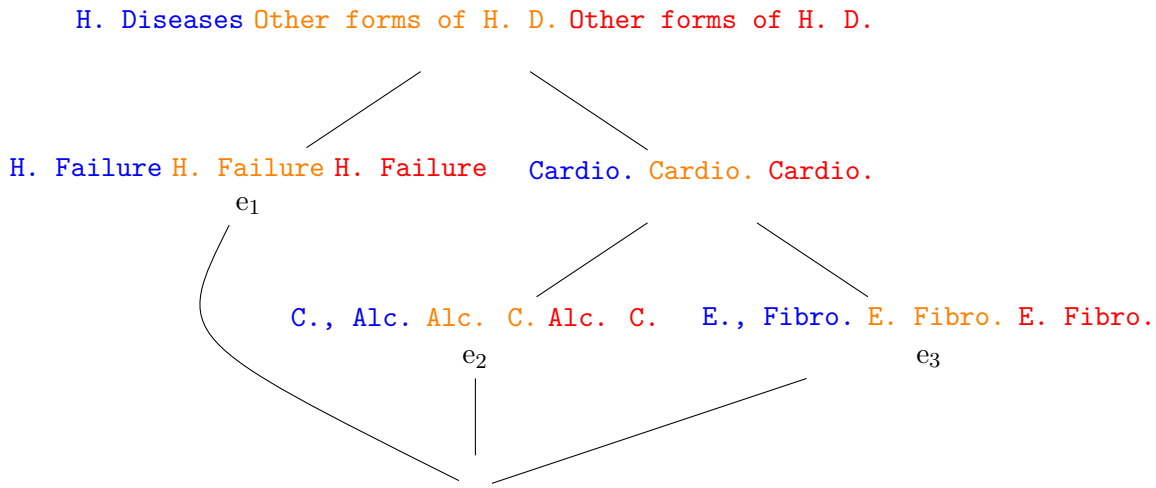


Figure 7.7: Line diagram representing the annotated lattice based on the concept lattice in Figure 7.6 and the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ in Figures 7.5b and 7.5c. Formal objects are depicted in black, descriptions in blue (classes from $\mathcal{O}_{\texttt{ref}}$), and annotations in orange (classes from $\mathcal{O}_1$) and red (classes from $\mathcal{O}_2$).

possible to read subsumption relations between classes of $\mathcal{O}_1$ and $\mathcal{O}_2$ by extending the process described in Subsection 7.2.3 to the two annotations. Finally, it is noteworthy that such reading of equivalence and subsumption relations could be performed with classes of $\mathcal{O}_{\texttt{ref}}$.

7.4 Discovering domain → range associations for predicates

In this section, we illustrate how Concept Annotation and Pattern Structures can be used to discover domain → range associations for predicate, *i.e.*, associations $C_1 \rightarrow C_2$, where C_1 and C_2 are two ontology classes that frequently appear as domain and range of a predicate based on its assertions in a knowledge graph $\mathcal{K}$. Such domain and range associations are interesting as they can indicate common behavior at the class level. Hence, they can somehow be seen similar to raising [177] or generalized association rules [155]

For instance, consider a predicate linking drugs to their side effects, as depicted in Figure 7.8. In our use case, as drugs and phenotypes respectively instantiate classes of ATC and MeSH, we could discover that instances of a MeSH class are frequently indicated as side effects of instances of an ATC class. This could be considered as the characterization of an effect between a family of drugs and a family of phenotypes.

In the following paragraphs, we consider entities, instantiating classes of an ontology $\mathcal{O}_1$, that are involved in the assertions of a predicate with other entities, instantiating classes of an ontology $\mathcal{O}_2$. Even if we use two ontologies, $\mathcal{O}_1$ and $\mathcal{O}_2$ don't need to be distinct. As a running example, we consider a relation, such as `side-effect`, between drugs and phenotypes, respectively instantiating (abstract) classes of ATC and MeSH as depicted in Figure 7.9.

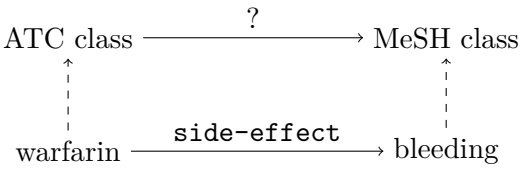
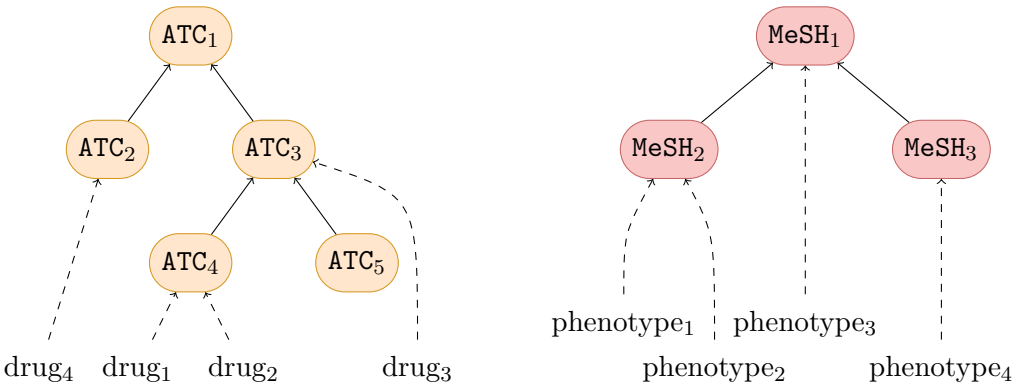


Figure 7.8: Example of a generalization from the predicate assertion between two entities to a domain → range association between two ontology classes.



(a) $\mathcal{O}_1$ (ATC) (b) $\mathcal{O}_2$1 (MeSH)

Figure 7.9: Drugs and phenotypes respectively instantiating classes of $\mathcal{O}_1$ (ATC) and $\mathcal{O}_2$ (MeSH). Dashed arrows represent instantiations of classes by entities and solid arrows represent subsumption relations between classes.

	phenotype_1	phenotype_2	phenotype_3	phenotype_4
drug_1	×	×		
drug_2	×	×		
drug_3		×	×	×
drug_4				×

Table 7.3: Formal context used to classify drugs and phenotypes involved in the assertions of a predicate, such as `side-effect`.

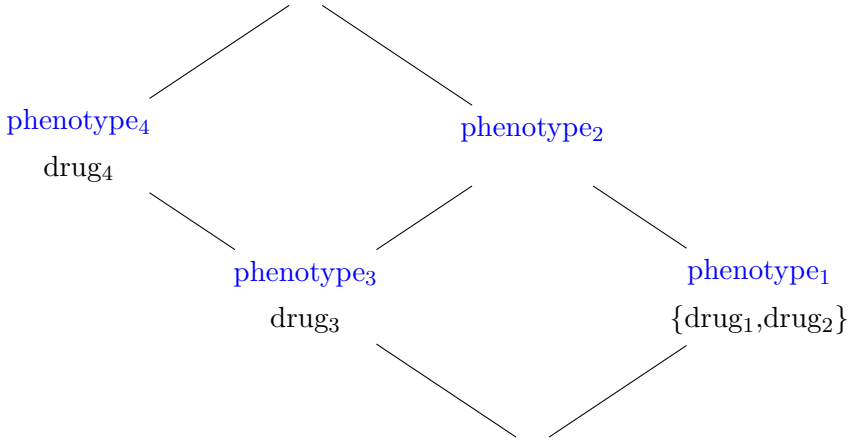


Figure 7.10: Line diagram representing the lattice resulting from the formal context in Table 7.3 and classifying drugs and phenotypes involved in the assertions of a predicate.

7.4.1 Classifying predicate instantiations in a concept lattice

As in previous applications of Concept Annotation, we first build a concept lattice that will then be annotated. This lattice represents the hierarchical organization of the entities involved in the assertions of a studied predicate (*e.g.*, `inhibitor`).

To this aim, we build the formal context (G, M, I) where G is the set of entities from $\mathcal{K}$ that instantiate classes from $\mathcal{O}_1$, and M is the set of entities from $\mathcal{K}$ that instantiate classes from $\mathcal{O}_2$. Given $g \in G$ and $m \in M$, $(g, m) \in I$ if and only if an assertion of the studied predicate exists between g and m. In our running example, G is the set of drugs and M is the set of phenotypes. Considering $g \in G$ and $m \in M$, $(g, m) \in I$ if and only if the drug g is linked to the phenotype m by the predicate `side-effect`. This could generate the formal context provided in Table 7.3. Then, standard binary FCA is applied on this formal context to generate the associated formal concepts organized in a concept lattice. The concept lattice resulting from the formal context of Table 7.3 is depicted in Figure 7.10.

7.4.2 Annotating the concept lattice with $\mathcal{O}_1$ and $\mathcal{O}_2$

The concept lattice resulting from the application of standard binary FCA can then be annotated by classes of $\mathcal{O}_1$ and $\mathcal{O}_2$. They are instantiated by objects and attributes respectively involved

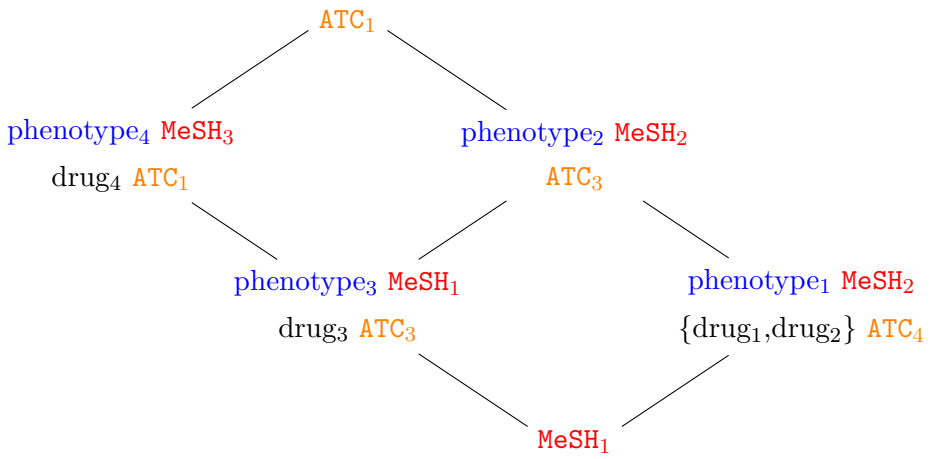


Figure 7.11: Line diagram representing the annotated lattice resulting from the lattice in Figure 7.10 and ontologies in Figure 7.9.

in the extent and the intent of formal concepts. Hence, we define *two* annotations, following the formalism from Subsection 7.3.1, that respectively derive from the extent and the intent of a formal concept.

We define two functions δ_1 and δ_2 that respectively associate an entity with the set of the most specific classes from $\mathcal{O}_1$ and $\mathcal{O}_2$ that this entity instantiates. $\sqcap_1$ and $\sqcap_2$ are used to compute the similarity between the descriptions of two entities with regard to the two considered ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. Finally, to compute the two annotations for each formal concept (A, B), two derivation operators $(\cdot)^{\circ_1} : 2^G \to 2^{\text{classes}(\mathcal{O}_1)}$ and $(\cdot)^{\circ_2} : 2^M \to 2^{\text{classes}(\mathcal{O}_2)}$ are defined as follows:

$$A^{\circ_1} = \bigsqcap_{g \in A}^{1} \delta_1(g) \quad \text{and} \quad B^{\circ_2} = \bigsqcap_{m \in B}^{2} \delta_2(m)$$

It is noteworthy that $(\cdot)^{\circ_1}$ is applied on the extent of a formal concept whereas $(\cdot)^{\circ_2}$ is applied on the intent.

As a result, the annotation process replaces each formal concept (A, B) by its annotated concept $(A, B, A^{\circ_1}, B^{\circ_2})$ where A°_1} (respectively B°_2}) is the set of the most specific classes of $\mathcal{O}_1$ (respectively $\mathcal{O}_2$) that all the individuals in A (respectively in B) instantiate. For example, the annotation of the lattice in Figure 7.10 by classes of ontologies in Figure 7.9 results in the annotated lattice in Figure 7.11.

7.4.3 Reading domain → range associations from the annotated lattice

After the annotation of the concept lattice, it is possible to read domain → range associations. Let us consider an annotated concept $(A, B, A^{\circ_1}, B^{\circ_2})$. Every entity in A is involved in an instantiation of the considered predicate with every entity in B. Furthermore, A°_1} is the set of the most specific classes of $\mathcal{O}_1$ that all the entities in A instantiate. Similarly, B°_2} is the set of the most specific classes of $\mathcal{O}_2$ that all the entities in B instantiate. Therefore, from this annotated concept, for the considered predicate, we can read the domain → range association $C_1 \to C_2$ for every pair $(C_1, C_2) \in A^{\circ_1} \times B^{\circ_2}$. For example, in the annotated lattice in Figure 7.11, we can

read the association $\texttt{ATC}_4 \rightarrow \texttt{MeSH}_2$.

Along the lattice hierarchy, the two annotations A°_1} and B°_2} behave in opposite ways. Indeed, A°_1}, computed on the extent, contains more specific classes when the number of entities in the extent decreases, *i.e.*, when browsing the lattice top to bottom. On the contrary, B°_2}, computed on the intent, contains more specific classes when the number of entities in the intent decreases, *i.e.*, when browsing the lattice bottom to top. Consequently, in annotated concepts at the top of the lattice, general classes of $\mathcal{O}_1$ will act as domain and specific classes of $\mathcal{O}_2$ will act as range. On the contrary, in annotated concepts at the bottom of the lattice, specific classes of $\mathcal{O}_1$ will act as domain and general classes of $\mathcal{O}_2$ will act as range. Such a behavior is of interest in an interactive setting. Indeed, an expert could interactively browse the lattice, guided by constraints on specificity of classes in domain or range. The hierarchical structure of the lattice would help to obtain more general or specific involved classes. Additional constraints such as support (*i.e.*, number of entities in extent and/or in intent) could also guide this interactive discovery process.

7.5 Discussion and perspectives

In this chapter, we proposed an extension of Formal Concept Analysis, named Concept Annotation. In this extension, an original lattice is considered as a "pivot": its original hierarchical organization is preserved and enriched with additional information from which new knowledge is discovered. This two-step process is useful when having a set of reference features that must guide the hierarchical organization. The latter is then enriched with additional elements. We illustrated how Concept Annotation in different settings with three use cases: *(i)* discovering subsumption axioms, *(ii)* suggesting alignments between classes of an ontology, and *(iii)* discovering domain $\rightarrow$ range associations for predicates.

Regarding the first use case, some of the suggested subsumption axioms between ontology classes already exist in the considered ontology. Thus, a lattice classifying entities w.r.t. their predicates has a structure similar to some parts of the existing ontology. This results means that predicates are important features in the definition of these ontology classes. However, the annotated lattice does not exactly match the ontology hierarchy. Indeed, we notice some annotated concepts with an empty annotation. This can be interpreted as the need for a new class in the ontology structure, corresponding to the entities and predicates respectively in the extent and intent of the formal concept. Some of the suggested axioms are inferable, meaning the lattice is not as fine-grained as the ontology. To increase the granularity of the lattice, we could test other attributes such as objects of RDF triples, pairs $(\texttt{predicate}, \texttt{object})$, or pairs $(\texttt{predicate}, \texttt{class})$ to take into account classes instantiated by such objects. A major future work consists in performing a quantitative (recall, precision) and qualitative evaluation of the discovered axioms. The qualitative evaluation could consist in expert evaluations or comparing two versions of the same ontology to detect if some suggested new axioms are added. Among new axioms, some seem interesting and globally valid while others are only locally valid and descriptive of the considered data set. Therefore, we could investigate some ways (*e.g.*, metrics) to automatically discriminate these two sorts of new axioms. Finally, we could perform similar experiments with an annotation defined using the formalism from Subsection 7.3.1. This would allow us to investigate the differences between the two formalisms.

Regarding the second use case, the approach needs to be validated on a real data set where alignments already exist. One main drawback lies in the definition of the annotation under the Closed World Assumption. Thus, alignments are suggested considering that all instantiations

are correct and none is missing. As many data sets are under the Open World Assumption, the suggested alignments may only be valid on the considered data set and should be validated by an expert. Therefore, as future work, we need to investigate the definition of a derivation operator "relaxing" the annotation to consider the Open World Assumption. Such an operator could also be useful in the other use cases. As previously, the features considered to build the original lattice (here, $\mathcal{O}_{\texttt{ref}}$) are of importance as they impact the original pivot structure from which equivalence relationships are then suggested. Thus, other features could be studied, such as predicates. An interesting application of this setting resides in highlighting concept drift when annotating with different versions of the same ontology. Indeed, a semantic change in classes between two (or more) versions would be indicated by annotating the same set of entities with different classes. In this case, the structure of the lattice could indicate the nature of the change. For example, finding the new version of the class higher in the lattice than the older version would indicate a generalization.

Regarding the third use case, we could also benefit from experimenting on a real data set. The main challenge in this setting lies in selecting interesting domain $\rightarrow$ range associations. Additionally to an interactive discovery process, various metrics could be considered to highlight interesting annotated concepts. For example, support and confidence based on the cardinal of extents and / or intents and depth of classes in their respective hierarchy could be of interest. Subsumption relation between annotated concepts could be used to build a hierarchical organization of domain $\rightarrow$ range associations. However, as the two annotations behave in opposite ways, this hierarchy should not be read as an order.

Finally, a major future work lies in theoretically and empirically comparing Concept Annotation with FCA and its extensions. Indeed, we showed that $(\cdot)^{\diamond\diamond}$ is a closure operator. Thus, it could be interesting to compare with triadic analysis [98], which considers three sets of elements as our annotated concepts. Since Concept Annotation is a two-step process, it is also close to Relational Concept Analysis [77] which iterates over several steps until convergence.

Chapter 8

Conclusion and perspectives

Contents

8.1 Summary of contributions 143
8.2 Perspectives 144

8.1 Summary of contributions

In this thesis, we studied the two key tasks of matching and mining in knowledge graphs. These tasks are challenging in several aspects. In our work, we coped with the inherent heterogeneity of knowledge graphs (*e.g.*, in terms of vocabularies, granularities) and the scalability issues associated with their ever-increasing number and size. We investigated how domain knowledge represented within knowledge graphs can help tackle such issues. Motivated by a real-world application in pharmacogenomics, we illustrated our work on this domain, which introduced additional challenges. Indeed, no knowledge graph focusing on the integration of pharmacogenomic knowledge was originally available. Additionally, pharmacogenomic knowledge is formed by relationships that are n-ary and whose arguments consist of sets of individuals, which affects their representation and matching.

In Chapter 3, we developed and published several resources related to the pharmacogenomic domain. Hence, we designed PGxO, a simple ontology to represent, integrate, and trace pharmacogenomic knowledge of various origins. We instantiated this ontology with pharmacogenomic relationships from three distinct sources thanks to *(i)* an automatic extraction of the reference database PharmGKB, *(ii)* the execution of a machine learning model on abstracts of biomedical articles available in PubMed, and *(iii)* the manual representation of results found in studies of Electronic Health Records of hospitals. This instantiation of PGxO constituted the PGxLOD knowledge graph, which also integrates various existing Linked Open Data sets about drugs, genomic variations, and phenotypes. We believe PGxLOD is an interesting resource for the pharmacogenomic domain as well as for Computer Science research. This knowledge graph motivated and served as the experimental setting for some of our other contributions.

Regarding the matching task, inspired by pharmacogenomic relationships, we designed and experimented two approaches. In Chapter 4, we proposed to view such n-ary relationships as n-ary tuples to match. We defined a symbolic approach consisting of five rules that perform a structure-based comparison of two such tuples and conclude on their relatedness among five levels. To tackle heterogeneity issues in the representation of tuples, this structured-based compar-

ison uses preorders that are based on domain knowledge (*e.g.*, the hierarchy of ontology classes and links between individuals). The abstract framework defined for this symbolic approach is well-suited for the matching of pharmacogenomic relationships but is flexible enough to be useful in other use cases. The obtained alignments on PGxLOD indicated some expected overlaps between the considered sources of pharmacogenomic knowledge, which seemed to validate our approach. Interestingly, the less "strict" rule resulted in the most results, which highlights the need for flexibility to cope with heterogeneous representations in knowledge graphs.

This need for flexibility motivated us to investigate graph embedding approaches in Chapter 5 since the continuous representation of embeddings is inherently more fluid [75]. In this framework, we proposed to match nodes by clustering their embeddings learned with Graph Convolutional Networks (GCNs). Inspired by recent claims and approaches combining the powerful representation of graph embedding with domain knowledge and the formal semantics of knowledge graphs, we particularly investigated the interplay between domain knowledge and GCN models. Our results showed that considering domain knowledge and associated inference rules tends to improve the matching results. Additionally, even if our GCN model was agnostic to the exact alignment relations holding between entities during training (*e.g.*, equivalence, weak similarity), distances in the embedding space were coherent with such alignment relations (*e.g.*, small distances for equivalences, large distances for weak similarities). These different distance distributions between alignment relations could somehow correspond to their rediscovery by the model. Such results advocate for a further integration of domain knowledge within embedding models.

Regarding the task of mining knowledge graphs, in Chapter 6, we proposed to mine neighbors, paths, and path patterns associated with a given set of seed nodes. These features are of interest since they can be interpreted by human agents, and thus benefit explainable approaches [164] or can be used for "semantic" embeddings [133]. To tackle the scalability issues associated with such a mining, we relied on the hierarchy of ontology classes to prune redundant patterns, and a set of constraints to reduce the number of mined neighbors, paths, and path patterns. We illustrated our approach and proposed a preliminary empirical evaluation by experimenting on PGxLOD. Our work alleviates part of the computational cost (time and memory) of mining paths and path patterns but also reveals the need for a further reduction.

Finally, we proposed in Chapter 7 an extension of Formal Concept Analysis (FCA) designed for knowledge graph refinement and mining that we named Concept Annotation. By annotating an original concept lattice with classes of ontologies, these classes are hierarchically organized according to the regularities expressed in the original lattice. Such regularities allow to mine subsumption axioms, alignment axioms, or domain $\rightarrow$ range associations between classes of ontologies. A preliminary experiment showed that some valid new subsumption axioms could be discovered with this methodology. Additional experiments are needed to confirm these results as well as a theoretical comparison of Concept Annotation with FCA and its other extensions.

8.2 Perspectives

Each chapter of this thesis discusses the perspectives regarding the contribution presented within. Here, we envision some larger or transversal perspectives about our research work.

Regarding our application in pharmacogenomics, recall that PGxLOD mainly contains pharmacogenomic relationships from the state of the art. Indeed, we only manually represented 10 relationships from studies of Electronic Health Records of hospitals as a proof of concept. Hence, one major perspective resides in developing an "observational" version of PGxLOD that inte-

grates either results of mining EHRs or EHRs themselves. Clinical practice and clinical decision support systems could then benefit from matching state-of-the-art knowledge with EHRs. However, integrating such observational knowledge raises several issues related to the research fields of text mining and data privacy (*e.g.*, anonymization, access control [90]).

Previously, we indicated that PGxLOD is an useful resource both for biomedical and Computer Science research. We illustrated in all our contributions, and particularly in Chapter 5 and Chapter 6, the various characteristics of PGxLOD: aggregation of several data sets (*i.e.*, `owl:sameAs` links), integration of several ontologies (*i.e.*, hierarchy of classes, hierarchy of predicates, inverses and symmetry of predicates), and a medium size (*i.e.*, scalability issues). Hence, we believe PGxLOD constitutes an interesting knowledge graph to experiment matching and mining approaches. That is why we could envision to propose its consideration in the Ontology Alignment Evaluation Initiative [129].

Regarding our matching approaches, their respective chapters detail some perspectives for each of them separately. However, one perspective also resides in studying their interaction. Indeed, we used the alignments resulting from our rule-based approach as positive examples in the learning process of our GCN-based approach. Reciprocally, we could envision to derive new rules from the results of the matching performed in the embedding space. Such an enrichment requires to be able to somehow "transform" clusters from the embedding space into symbolic features, which reminds of the "semantic" embeddings proposed by Paulheim [133]. Interestingly, he mentions that graph patterns or Horn clauses could be used to build these semantic embeddings. Hence, we could leverage our mining method from Chapter 6 to extract features that characterize clusters. To associate distances and clusters in the embedding spaces with such features, Formal Concept Analysis and our Concept Annotation (Chapter 7) could be used. Indeed, matched nodes and their neighbors, paths, and paths patterns could form a formal context. The resulting lattice could then be annotated with distances in the embedding space (using interval pattern structures) or cluster assignements. An analyst could then interactively traverse the lattice to find concepts of interest from which new rules could be derived. This traversal could also be guided by several metrics than can be computed on formal concepts.

We highlighted the scalability issues associated with ever-increasing knowledge graphs when mining neighbors, paths, and path patterns in Chapter 6. However, such scalability issues also arise in our matching approaches. For example, our rule-based approach (Chapter 4) needed 54 hours, 4 cores, and 15 GB of RAM to be executed. Additionally, knowledge graphs are always improving: knowledge can be added, updated, or removed. Hence, the question of updating the alignments between PGx relationships when new ones are integrated in PGxLOD arises. Regarding our graph embedding approach, it needs to be trained again with all the relationships. Consequently, integrating new relationships and matching them with the existing knowledge is computationally expensive. To avoid this computational burden, *inductive* graph embedding approaches should be considered. Our rule-based approach presents a reduced computational cost: a new relationship "only" needs to be compared with all existing ones. However, we could design strategies to reduce the amount of comparisons by only trying to match the new relationship with some others and completing results by relying on the transitivity and symmetry properties that our rules satisfy and existing alignments.

Finally, other aspects of pharmacogenomic knowledge and metadata are not currently considered but pave the way for perspectives in Computer Science approaches. For example, PGxO allows to represent negation within relationships (*e.g.*, side effects not caused by a relationship). Consequently, it could be possible to match contradictory PGx relationships. We could broaden our approaches to tackle such a matching, which would raise several interesting questions such as the geometric representation of contradiction in the embedding space. Extending the match-

ing task, we could envision the task of knowledge validation, *i.e.*, confirming or moderating the validation of a knowledge unit based on similar or contradictory units existing in other sources. However, pharmacogenomic relationships can come with heterogeneous quality metrics such as the levels of evidence of PharmGKB, odd ratios in biomedical articles, or quality metrics of mining approaches executed on Electronic Health Records. These quality metadata are already stored in the provenance model we defined in Chapter 3. Hence, a research question lies in defining a knowledge validation model that uses the alignments output by our matching approaches and is able to deal with the heterogeneity in quality metadata. To this aim, we could build upon existing works regarding provenance and probabilistic databases [148].

Appendix A

List of publications

International journal

- Pierre Monnin, Joël Legrand, Graziella Husson, Patrice Ringot, Andon Tchechmedjiev, Clément Jonquet, Amedeo Napoli, and Adrien Coulet. "PGxO and PGxLOD: a reconciliation of pharmacogenomic knowledge of various provenances, enabling further comparison". In: *BMC Bioinformatics* 20-S.4 (2019), 139:1–139:16. DOI: 10.1186/s12859-019-2693-9.

International conferences (with program committee)

- Pierre Monnin, Emmanuel Bresso, Miguel Couceiro, Malika Smaïl-Tabbone, Amedeo Napoli, and Adrien Coulet. "Tackling scalability issues in mining path patterns from knowledge graphs: a preliminary study". In: *1st international conference "Algebras, graphs and ordered sets" (ALGOS 2020)*. Ed. by Miguel Couceiro, Pierre Monnin, and Amedeo Napoli. Nancy, France, Aug. 2020. URL: https://hal.inria.fr/hal-02913224/document.

- Pierre Monnin, Miguel Couceiro, Amedeo Napoli, and Adrien Coulet. "Knowledge-Based Matching of n-ary Tuples". In: *Ontologies and Concepts in Mind and Machine - 25th International Conference on Conceptual Structures, ICCS 2020, Bolzano, Italy, September 18–20, 2020, Proceedings*. Ed. by Mehwish Alam, Tanya Braun, and Bruno Yun. Vol. 12277. Lecture Notes in Computer Science. Springer, 2020, pp. 48–56. DOI: 10.1007/978-3-030-57855-8_4. URL: https://doi.org/10.1007/978-3-030-57855-8_4.

- Pierre Monnin, Mario Lezoche, Amedeo Napoli, and Adrien Coulet. "Using Formal Concept Analysis for Checking the Structure of an Ontology in LOD: The Example of DBpedia". In: *Foundations of Intelligent Systems - 23rd International Symposium, ISMIS 2017, Warsaw, Poland, June 26-29, 2017, Proceedings*. Ed. by Marzena Kryszkiewicz, Annalisa Appice, Dominik Slezak, Henryk Rybinski, Andrzej Skowron, and Zbigniew W. Ras. Vol. 10352. Lecture Notes in Computer Science. Springer, 2017, pp. 674–683. DOI: 10.1007/978-3-319-60438-1_66. URL: https://doi.org/10.1007/978-3-319-60438-1_66.

International workshops (with program committee)

- Pierre Monnin, Chedy Raïssi, Amedeo Napoli, and Adrien Coulet. "Knowledge Reconciliation with Graph Convolutional Networks: Preliminary Results". In: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019), Portoroz, Slovenia, June 2, 2019.* Ed. by Mehwish Alam, Davide Buscaldi, Michael Cochez, Francesco Osborne, Diego Reforgiato Recupero, and Harald Sack. Vol. 2377. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 47–56. URL: http://ceur-ws.org/Vol-2377/paper_6.pdf.

- Pierre Monnin. "Discovering and Comparing Relational Knowledge, the Example of Pharmacogenomics". In: *Proceedings of the EKAW Doctoral Consortium 2018 co-located with the 21st International Conference on Knowledge Engineering and Knowledge Management (EKAW 2018), Nancy, France, November 13, 2018.* Ed. by Laura Hollink and Francesco Osborne. Vol. 2306. CEUR Workshop Proceedings. CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2306/paper5.pdf.

- Pierre Monnin, Amedeo Napoli, and Adrien Coulet. "Combining Concept Annotation and Pattern Structures for Guiding Ontology Mapping". In: *Proceedings of the 6th International Workshop "What can FCA do for Artificial Intelligence"? co-located with International Joint Conference on Artificial Intelligence and European Conference on Artificial Intelligence (IJCAI/ECAI 2018), Stockholm, Sweden, July 13, 2018.* Ed. by Sergei O. Kuznetsov, Amedeo Napoli, and Sebastian Rudolph. Vol. 2149. CEUR Workshop Proceedings. CEUR-WS.org, 2018, pp. 117–126. URL: http://ceur-ws.org/Vol-2149/paper11.pdf.

- Pierre Monnin, Clément Jonquet, Joël Legrand, Amedeo Napoli, and Adrien Coulet. "PGxO: A very lite ontology to reconcile pharmacogenomic knowledge units". In: *Methods, tools & platforms for Personalized Medicine in the Big Data Era, NETTAB 2017*. Vol. 5. PeerJ PrePrints. PeerJ, 2017, e3140. DOI: 10.7287/peerj.preprints.3140v1.

Poster in national conference (with program committee)

- Pierre Monnin, Amedeo Napoli, and Adrien Coulet. "Discovering Subsumption Axioms with Concept Annotation". In: *Gestion de Données – Principes, Technologies et Applications (BDA 2017)*. 2017. URL: https://hal.inria.fr/hal-01671454/document.

Others

- François-Élie Calvier, Pierre Monnin, Miguel Boland, Patryk Jarnot, Emmanuel Bresso, Malika Smaïl-Tabbone, Adrien Coulet, and Cedric Bousquet. "Providing Molecular Characterization for Unexplained Adverse Drug Reactions". Podium Abstract at MedInfo 2019, Lyon, France. July 2019. URL: https://hal.inria.fr/hal-02196134/document.

- Pierre Monnin, Amedeo Napoli, and Adrien Coulet. "Data-Interlinking: the Seed of Knowledge Reconciliation in Pharmacogenomics". Presented at the Workshop "Symbolic methods for data-interlinking" co-located with the 21st International Conference on Knowledge Engineering and Knowledge Management (EKAW 2018), Nancy, France. Nov. 2018. URL: https://hal.inria.fr/hal-01955262/document.

References

[1] Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, and Pierre Senellart. *Web Data Management*. Cambridge University Press, 2011. ISBN: 978-1-10-701243-1.

[2] Charu C. Aggarwal and Haixun Wang, eds. *Managing and Mining Graph Data*. Vol. 40. Advances in Database Systems. Springer, 2010. ISBN: 978-1-4419-6044-3. DOI: 10.1007/978-1-4419-6045-0. URL: https://doi.org/10.1007/978-1-4419-6045-0.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. "Mining Association Rules between Sets of Items in Large Databases". In: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993*. Ed. by Peter Buneman and Sushil Jajodia. ACM Press, 1993, pp. 207–216. DOI: 10.1145/170035.170072. URL: https://doi.org/10.1145/170035.170072.

[4] Rajendra Akerkar and Priti Sajja. *Knowledge-based systems*. Jones & Bartlett Publishers, 2010.

[5] Mehwish Alam, Aleksey Buzmakov, Víctor Codocedo, and Amedeo Napoli. "Mining Definitions from RDF Annotations Using Formal Concept Analysis". In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. Ed. by Qiang Yang and Michael J. Wooldridge. AAAI Press, 2015, pp. 823–829. URL: http://ijcai.org/Abstract/15/121.

[6] Mehwish Alam, Diego Reforgiato Recupero, Misael Mongiovì, Aldo Gangemi, and Petar Ristoski. "Reconciling Event-Based Knowledge Through RDF2VEC". In: *Joint Proceedings of the International Workshops on Hybrid Statistical Semantic Understanding and Emerging Semantics, and Semantic Statistics co-located with 16th International Semantic Web Conference, HybridSemStats@ISWC 2017, Vienna, Austria October 22nd, 2017*. Ed. by Sarven Capadisli, Franck Cotton, Xin Luna Dong, Ramanathan V. Guha, Armin Haller, Pascal Hitzler, Evangelos Kalampokis, Mayank Kejriwal, Freddy Lécué, D. Sivakumar, Pedro A. Szekely, Raphaël Troncy, and Michael J. Witbrock. CEUR Workshop Proceedings. CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-1923/article-03.pdf.

[7] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. "OPTICS: Ordering Points To Identify the Clustering Structure". In: *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*. Ed. by Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh. ACM Press, 1999, pp. 49–60. DOI: 10.1145/304182.304187. URL: https://doi.org/10.1145/304182.304187.

[8] Manuel Atencia, Jérôme David, and Jérôme Euzenat. "Data interlinking through robust linkkey extraction". In: *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*. Ed. by Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan. Vol. 263. Frontiers in Artificial Intelligence and Applications. IOS Press, 2014, pp. 15–20. DOI: 10.3233/978-1-61499-419-0-15. URL: https://doi.org/10.3233/978-1-61499-419-0-15.

[9] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. "The description logics handbook". In: *Theory Implementations and Applications, Cambridge* (2003).

[10] Franz Baader, Ralf Küsters, and Ralf Molitor. "Computing Least Common Subsumers in Description Logics with Existential Restrictions". In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*. Ed. by Thomas Dean. Morgan Kaufmann, 1999, pp. 96–103. URL: http://ijcai.org/Proceedings/99-1/Papers/015.pdf.

[11] Pablo Barceló, Leonid Libkin, and Juan L. Reutter. "Querying graph patterns". In: *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*. Ed. by Maurizio Lenzerini and Thomas Schwentick. ACM, 2011, pp. 199–210. DOI: 10.1145/1989284.1989307. URL: https://doi.org/10.1145/1989284.1989307.

[12] Rokia Bendaoud, Amedeo Napoli, and Yannick Toussaint. "Formal Concept Analysis: A Unified Framework for Building and Refining Ontologies". In: *Knowledge Engineering: Practice and Patterns, 16th International Conference, EKAW 2008, Acitrezza, Italy, September 29 - October 2, 2008. Proceedings*. Ed. by Aldo Gangemi and Jérôme Euzenat. Vol. 5268. Lecture Notes in Computer Science. Springer, 2008, pp. 156–171. DOI: 10.1007/978-3-540-87696-0_16. URL: https://doi.org/10.1007/978-3-540-87696-0_16.

[13] Michael K. Bergman. *A Common Sense View of Knowledge Graphs*. July 2019. URL: https://www.mkbergman.com/2244/a-common-sense-view-of-knowledge-graphs/ (visited on 07/28/2020).

[14] Tim Berners-Lee. *Linked Data - Design Issues*. July 2006. URL: https://www.w3.org/DesignIssues/LinkedData.html (visited on 08/10/2020).

[15] Tim Berners-Lee, James Hendler, and Ora Lassila. "The semantic web". In: *Scientific american* 284.5 (2001), pp. 28–37.

[16] Kelly A. Birdwell, Ben Grady, Leena Choi, Hua Xu, Aihua Bian, Josh C. Denny, Min Jiang, Gayle Vranic, Melissa Basford, James D. Cowan, Danielle M. Richardson, Melanie P. Robinson, Talat Alp Ikizler, Marylyn D. Ritchie, Charles Michael Stein, and David W. Haas. "The use of a DNA biobank linked to electronic medical records to characterize pharmacogenomic predictors of tacrolimus dose requirement in kidney transplant recipients". In: *Pharmacogenetics and Genomics* 22.1 (Jan. 2012), pp. 32–42. ISSN: 1744-6872.

[17] Christian Bizer, Tom Heath, and Tim Berners-Lee. "Linked Data - The Story So Far". In: *Int. J. Semantic Web Inf. Syst.* 5.3 (2009), pp. 1–22. DOI: 10.4018/jswis.2009081901. URL: https://doi.org/10.4018/jswis.2009081901.

[18] Olivier Bodenreider. “The Unified Medical Language System (UMLS): integrating biomedical terminology”. In: *Nucleic Acids Research* 32.Database-Issue (2004), pp. 267–270. DOI: 10.1093/nar/gkh061. URL: https://doi.org/10.1093/nar/gkh061.

[19] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. “Freebase: a collaboratively created graph database for structuring human knowledge”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*. Ed. by Jason Tsong-Li Wang. ACM, 2008, pp. 1247–1250. DOI: 10.1145/1376616.1376746. URL: https://doi.org/10.1145/1376616.1376746.

[20] Piero Andrea Bonatti, Stefan Decker, Axel Polleres, and Valentina Presutti. “Knowledge Graphs: New Directions for Knowledge Representation on the Semantic Web (Dagstuhl Seminar 18371)”. In: *Dagstuhl Reports* 8.9 (2019). Ed. by Piero Andrea Bonatti, Stefan Decker, Axel Polleres, and Valentina Presutti, pp. 29–111. ISSN: 2192-5283. DOI: 10.4230/DagRep.8.9.29. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10328.

[21] Guillaume Bonfante, Bruno Guillaume, and Guy Perrier. *Application of Graph Rewriting to Natural Language Processing*. Wiley Online Library, 2018.

[22] Antoine Bordes and Evgeniy Gabrilovich. “Constructing and mining web-scale knowledge graphs: KDD 2014 tutorial”. In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’14, New York, NY, USA - August 24 - 27, 2014*. Ed. by Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani. ACM, 2014, p. 1967. DOI: 10.1145/2623330.2630803. URL: https://doi.org/10.1145/2623330.2630803.

[23] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. “Translating Embeddings for Modeling Multi-relational Data”. In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.* Ed. by Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger. 2013, pp. 2787–2795. URL: http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.

[24] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. “Learning Structured Embeddings of Knowledge Bases”. In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. Ed. by Wolfram Burgard and Dan Roth. AAAI Press, 2011. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3659.

[25] Rajendra Bose and James Frew. “Lineage retrieval for scientific data processing: a survey”. In: *ACM Comput. Surv.* 37.1 (2005), pp. 1–28. DOI: 10.1145/1057977.1057978. URL: https://doi.org/10.1145/1057977.1057978.

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

[27] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. “A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications”. In: *IEEE Trans. Knowl. Data Eng.* 30.9 (2018), pp. 1616–1637. DOI: 10.1109/TKDE.2018.2807452. URL: https://doi.org/10.1109/TKDE.2018.2807452.

[28] François-Élie Calvier, Pierre Monnin, Miguel Boland, Patryk Jarnot, Emmanuel Bresso, Malika Smaïl-Tabbone, Adrien Coulet, and Cedric Bousquet. "Providing Molecular Characterization for Unexplained Adverse Drug Reactions". Podium Abstract at MedInfo 2019, Lyon, France. July 2019. URL: https://hal.inria.fr/hal-02196134/document.

[29] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. "Toward an Architecture for Never-Ending Language Learning". In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. Ed. by Maria Fox and David Poole. AAAI Press, 2010. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1879.

[30] Kelly E. Caudle, Teri E. Klein, James M. Hoffman, Daniel J. Muller, Michelle Whirl-Carrillo, Li Gong, Ellen M. McDonagh, Katrin Sangkuhl, Caroline F. Thorn, Matthias Schwab, Jose A.G. Agundez, Robert R. Freimuth, Vojtech Huser, Ming Ta Michael Lee, Otito F. Iwuchukwu, Kristine R. Crews, Stuart A. Scott, Mia Wadelius, Jesse J. Swen, Rachel F. Tyndale, C. Michael Stein, Dan Roden, Mary V. Relling, Marc S. Williams, and Samuel G. Johnson. "Incorporation of Pharmacogenomics into Routine Clinical Practice: the Clinical Pharmacogenetics Implementation Consortium (CPIC) Guideline Development Process". In: *Current Drug Metabolism* 15.2 (2014), pp. 209–217. ISSN: 1389-2002/1875-5453. DOI: 10.2174/1389200215666140130124910.

[31] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. *Machine Learning on Graphs: A Model and Comprehensive Taxonomy*. 2020. arXiv: 2005.03675. URL: https://arxiv.org/abs/2005.03675.

[32] Minjun Chen, Ayako Suzuki, Shraddha Thakkar, Ke Yu, Chuchu Hu, and Weida Tong. "DILIrank: the largest reference drug list ranked by the risk for developing drug-induced liver injury in humans". In: *Drug Discov Today* 21.4 (2016), pp. 648–653.

[33] Philipp Cimiano, Andreas Hotho, and Steffen Staab. "Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis". In: *J. Artif. Intell. Res.* 24 (2005), pp. 305–339. DOI: 10.1613/jair.1648. URL: https://doi.org/10.1613/jair.1648.

[34] Michael Cochez, Petar Ristoski, and Simone Paolo Ponzetto and. "Biased graph walks for RDF graph embeddings". In: *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, WIMS 2017, Amantea, Italy, June 19-22, 2017*. Ed. by Rajendra Akerkar, Alfredo Cuzzocrea, Jannong Cao, and Mohand-Said Hacid. ACM, 2017, 21:1–21:12. DOI: 10.1145/3102254.3102279. URL: https://doi.org/10.1145/3102254.3102279.

[35] Víctor Codocedo and Amedeo Napoli. "A Proposition for Combining Pattern Structures and Relational Concept Analysis". In: *Formal Concept Analysis - 12th International Conference, ICFCA 2014, Cluj-Napoca, Romania, June 10-13, 2014. Proceedings*. Ed. by Cynthia Vera Glodeanu, Mehdi Kaytoue, and Christian Sacarea. Vol. 8478. Lecture Notes in Computer Science. Springer, 2014, pp. 96–111. DOI: 10.1007/978-3-319-07248-7_8. URL: https://doi.org/10.1007/978-3-319-07248-7_8.

[36] Adrien Coulet, Florent Domenach, Mehdi Kaytoue, and Amedeo Napoli. "Using Pattern Structures for Analyzing Ontology-Based Annotations of Biomedical Data". In: *Formal Concept Analysis, 11th International Conference, ICFCA 2013, Dresden, Germany, May 21-24, 2013. Proceedings*. Ed. by Peggy Cellier, Felix Distel, and Bernhard Ganter. Vol. 7880. Lecture Notes in Computer Science. Springer, 2013, pp. 76–91. DOI:

10.1007/978-3-642-38317-5_5. URL: https://doi.org/10.1007/978-3-642-38317-5_5.

[37] Adrien Coulet, Yael Garten, Michel Dumontier, Russ B. Altman, Mark A. Musen, and Nigam H. Shah. "Integration and publication of heterogeneous text-mined relationships on the Semantic Web". In: *J. Biomedical Semantics* 2.S-2 (2011), S10. URL: http://www.jbiomedsem.com/content/2/S2/S10.

[38] Adrien Coulet and Malika Smaïl-Tabbone. "Mining Electronic Health Records to Validate Knowledge in Pharmacogenomics". In: *ERCIM News* 2016.104 (2016). URL: http://ercim-news.ercim.eu/en104/special/mining-electronic-health-records-to-validate-knowledge-in-pharmacogenomics.

[39] Adrien Coulet, Malika Smaïl-Tabbone, Amedeo Napoli, and Marie-Dominique Devignes. "Suggested Ontology for Pharmacogenomics (SO-Pharm): Modular Construction and Preliminary Testing". In: *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, COMINF, IS, KSinBIT, MIOS-CIAO, MONET, OnToContent, ORM, PerSys, OTM Academy Doctoral Consortium, RDDS, SWWS, and SeBGIS 2006, Montpellier, France, October 29 - November 3, 2006. Proceedings, Part I*. Ed. by Robert Meersman, Zahir Tari, and Pilar Herrero. Vol. 4277. Lecture Notes in Computer Science. Springer, 2006, pp. 648–657. DOI: 10.1007/11915034_89. URL: https://doi.org/10.1007/11915034_89.

[40] Olivier C. Curé, Henri Maurer, Nigam H. Shah, and Paea LePendu. "A formal concept analysis and semantic query expansion cooperation to refine health outcomes of interest". In: *BMC Med. Inf. & Decision Making* 15.S-1 (2015), S8. DOI: 10.1186/1472-6947-15-S1-S8. URL: https://doi.org/10.1186/1472-6947-15-S1-S8.

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

[42] Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. "Query Answering and Ontology Population: An Inductive Approach". In: *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*. Ed. by Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis. Vol. 5021. Lecture Notes in Computer Science. Springer, 2008, pp. 288–302. DOI: 10.1007/978-3-540-68234-9_23. URL: https://doi.org/10.1007/978-3-540-68234-9_23.

[43] Claudia d'Amato, Steffen Staab, and Nicola Fanizzi. "On the Influence of Description Logics Ontologies on Conceptual Similarity". In: *Knowledge Engineering: Practice and Patterns, 16th International Conference, EKAW 2008, Acitrezza, Italy, September 29 - October 2, 2008. Proceedings*. Ed. by Aldo Gangemi and Jérôme Euzenat. Vol. 5268. Lecture Notes in Computer Science. Springer, 2008, pp. 48–63. DOI: 10.1007/978-3-540-87696-0_7. URL: https://doi.org/10.1007/978-3-540-87696-0_7.

[44] Kevin Dalleau, Yassine Marzougui, Sébastien Da Silva, Patrice Ringot, Ndeye Coumba Ndiaye, and Adrien Coulet. "Learning from biomedical linked data to suggest valid pharmacogenes". In: *J. Biomedical Semantics* 8.1 (2017), 16:1–16:12. DOI: 10.1186/s13326-017-0125-1. URL: https://doi.org/10.1186/s13326-017-0125-1.

[45] John Davies, Rudi Studer, and Paul Warren. *Semantic Web technologies: trends and research in ontology-based systems*. John Wiley & Sons, 2006.

[46] Allan Peter Davis, Cynthia J. Grondin, Robin J. Johnson, Daniela Sciaky, Roy McMorran, Jolene Wiegers, Thomas C. Wiegers, and Carolyn J. Mattingly. "The Comparative Toxicogenomics Database: update 2019". In: *Nucleic Acids Research* 47.Database-Issue (2019), pp. D948–D954. DOI: 10.1093/nar/gky868. URL: https://doi.org/10.1093/nar/gky868.

[47] Gerben Klaas Dirk de Vries and Steven de Rooij. "A Fast and Simple Graph Kernel for RDF". In: *Proceedings of the International Workshop on Data Mining on Linked Data, with Linked Data Mining Challenge collocated with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2013), Prague, Czech Republic, September 23, 2013*. Ed. by Claudia d'Amato, Petr Berka, Vojtech Svátek, and Krzysztof Wecel. Vol. 1082. CEUR Workshop Proceedings. CEUR-WS.org, 2013. URL: http://ceur-ws.org/Vol-1082/paper2.pdf.

[48] Gerben Klaas Dirk de Vries and Steven de Rooij. "Substructure counting graph kernels for machine learning from RDF data". In: *J. Web Semant.* 35 (2015), pp. 71–84. DOI: 10.1016/j.websem.2015.08.002. URL: https://doi.org/10.1016/j.websem.2015.08.002.

[49] J T Delaney, A H Ramirez, E Bowton, J M Pulley, M A Basford, J S Schildcrout, Y Shi, R Zink, M Oetjens, H Xu, J H Cleator, E Jahangir, M D Ritchie, D R Masys, D M Roden, D C Crawford, and J C Denny. "Predicting Clopidogrel Response Using DNA Samples Linked to an Electronic Health Record". In: *Clinical Pharmacology & Therapeutics* 91.2 (2012), pp. 257–263. DOI: 10.1038/clpt.2011.221.

[50] Joshua C. Denny, Sara L. Van Driest, Wei-Qi Wei, and Dan M. Roden. "The Influence of Big (Clinical) Data and Genomics on Precision Medicine and Drug Development". In: *Clinical Pharmacology & Therapeutics* 103.3 (2018), pp. 409–418. DOI: 10.1002/cpt.951.

[51] Rose Dieng, Olivier Corby, Alain Giboin, and Myriam Ribière. "Methods and tools for corporate knowledge management". In: *Int. J. Hum.-Comput. Stud.* 51.3 (1999), pp. 567–598. DOI: 10.1006/ijhc.1999.0281. URL: https://doi.org/10.1006/ijhc.1999.0281.

[52] Marcos Aurélio Domingues and Solange Oliveira Rezende. *Using Taxonomies to Facilitate the Analysis of the Association Rules*. 2011. arXiv: 1112.1734. URL: http://arxiv.org/abs/1112.1734.

[53] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. "Knowledge vault: a web-scale approach to probabilistic knowledge fusion". In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. Ed. by Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani. ACM, 2014, pp. 601–610. DOI: 10.1145/2623330.2623623. URL: https://doi.org/10.1145/2623330.2623623.

[54] Michel Dumontier, Alison Callahan, Jose Cruz-Toledo, Peter Ansell, Vincent Emonet, François Belleau, and Arnaud Droit. "Bio2RDF Release 3: A larger, more connected network of Linked Data for the Life Sciences". In: *Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014*. Ed. by Matthew Horridge,

Marco Rospocher, and Jacco van Ossenbruggen. Vol. 1272. CEUR Workshop Proceedings. CEUR-WS.org, 2014, pp. 401–404. URL: http://ceur-ws.org/Vol-1272/paper_121.pdf.

[55] Michel Dumontier and Natalia Villanueva-Rosales. "Towards pharmacogenomics knowledge discovery with the semantic web". In: *Briefings in Bioinformatics* 10.2 (2009), pp. 153–163. DOI: 10.1093/bib/bbn056. URL: https://doi.org/10.1093/bib/bbn056.

[56] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. *Benchmarking Graph Neural Networks*. 2020. arXiv: 2003.00982. URL: https://arxiv.org/abs/2003.00982.

[57] Lisa Ehrlinger and Wolfram Wöß. "Towards a Definition of Knowledge Graphs". In: *Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16) co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016), Leipzig, Germany, September 12-15, 2016*. Ed. by Michael Martin, Martí Cuquet, and Erwin Folmer. Vol. 1695. CEUR Workshop Proceedings. CEUR-WS.org, 2016. URL: http://ceur-ws.org/Vol-1695/paper4.pdf.

[58] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching, Second Edition*. Springer, 2013. ISBN: 978-3-642-38720-3.

[59] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. "From Data Mining to Knowledge Discovery: An Overview". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. AAAI/MIT Press, 1996, pp. 1–34.

[60] Christina Feilmayr and Wolfram Wöß. "An analysis of ontologies and their success factors for application to business". In: *Data Knowl. Eng.* 101 (2016), pp. 1–23. DOI: 10.1016/j.datak.2015.11.003. URL: https://doi.org/10.1016/j.datak.2015.11.003.

[61] Q. Feng, W. Q. Wei, C. P. Chung, R. T. Levinson, L. Bastarache, J. C. Denny, and C. M. Stein. "The effect of genetic variation in PCSK9 on the LDL-cholesterol response to statin therapy". In: *The Pharmacogenomics Journal* 17.2 (2017), pp. 204–208. ISSN: 1473-1150. DOI: 10.1038/tpj.2016.3. URL: https://doi.org/10.1038/tpj.2016.3.

[62] William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. "Knowledge Discovery in Databases: An Overview". In: *Knowledge Discovery in Databases*. Ed. by Gregory Piatetsky-Shapiro and William J. Frawley. AAAI/MIT Press, 1991, pp. 1–30.

[63] Nicholas Frosst, Nicolas Papernot, and Geoffrey E. Hinton. "Analyzing and Improving Representations with the Soft Nearest Neighbor Loss". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2012–2020. URL: http://proceedings.mlr.press/v97/frosst19a.html.

[64] Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M. Suchanek. "Canonicalizing Open Knowledge Bases". In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. Ed. by Jianzhong Li, Xiaoyang Sean Wang, Minos N. Garofalakis, Ian Soboroff, Torsten Suel, and Min Wang. ACM, 2014, pp. 1679–1688. DOI: 10.1145/2661829.2662073. URL: https://doi.org/10.1145/2661829.2662073.

[65] Luis Antonio Galárraga, Nicoleta Preda, and Fabian M. Suchanek. "Mining rules to align knowledge bases". In: *Proceedings of the 2013 workshop on Automated knowledge base construction, AKBC@CIKM 13, San Francisco, California, USA, October 27-28, 2013*. Ed. by Fabian M. Suchanek, Sebastian Riedel, Sameer Singh, and Partha Pratim Talukdar. ACM, 2013, pp. 43–48. DOI: 10.1145/2509558.2509566. URL: https://doi.org/10.1145/2509558.2509566.

[66] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. "AMIE: association rule mining under incomplete evidence in ontological knowledge bases". In: *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*. Ed. by Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo A. Baeza-Yates, and Sue B. Moon. International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 413–422. DOI: 10.1145/2488388.2488425. URL: https://doi.org/10.1145/2488388.2488425.

[67] Aldo Gangemi. "Ontology Design Patterns for Semantic Web Content". In: *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*. Ed. by Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen. Vol. 3729. Lecture Notes in Computer Science. Springer, 2005, pp. 262–276. DOI: 10.1007/11574620_21. URL: https://doi.org/10.1007/11574620_21.

[68] Aldo Gangemi. *Ontology:DOLCE+DnS Ultralite - Odp (accessed November 10th, 2019)*. URL: http://ontologydesignpatterns.org/wiki/Ontology:DOLCE+DnS_Ultralite.

[69] Bernhard Ganter and Sergei O. Kuznetsov. "Pattern Structures and Their Projections". In: *Conceptual Structures: Broadening the Base, 9th International Conference on Conceptual Structures, ICCS 2001, Stanford, CA, USA, July 30-August 3, 2001, Proceedings*. Ed. by Harry S. Delugach and Gerd Stumme. Vol. 2120. Lecture Notes in Computer Science. Springer, 2001, pp. 129–142. DOI: 10.1007/3-540-44583-8_10. URL: https://doi.org/10.1007/3-540-44583-8_10.

[70] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999. ISBN: 978-3-540-62771-5. DOI: 10.1007/978-3-642-59830-2. URL: https://doi.org/10.1007/978-3-642-59830-2.

[71] Genet Asefa Gesese, Russa Biswas, Mehwish Alam, and Harald Sack. *A Survey on Knowledge Graph Embeddings with Literals: Which model links better Literal-ly?* 2019. arXiv: 1910.12507. URL: http://arxiv.org/abs/1910.12507.

[72] Janet Piñero González, Núria Queralt-Rosinach, Àlex Bravo, Jordi Deu-Pons, Anna Bauer-Mehren, Martin Baron, Ferran Sanz, and Laura Inés Furlong. "DisGeNET: a discovery platform for the dynamical exploration of human diseases and their genes". In: *Database* 2015 (2015). DOI: 10.1093/database/bav028. URL: https://doi.org/10.1093/database/bav028.

[73] Omri Gottesman, Helena Kuivaniemi, Gerard Tromp, W. Andrew Faucett, Rongling Li, Teri A. Manolio, Saskia C. Sanderson, Joseph Kannry, Randi Zinberg, Melissa A. Basford, Murray Brilliant, David J. Carey, Rex L. Chisholm, Christopher G. Chute, John J. Connolly, David Crosslin, Joshua C. Denny, Carlos J. Gallego, Jonathan L. Haines, Hakon Hakonarson, John Harley, Gail P. Jarvik, Isaac Kohane, Iftikhar J. Kullo, Eric B. Larson, Catherine McCarty, Marylyn D. Ritchie, Dan M. Roden, Maureen E. Smith, Erwin P. Böttinger, Marc S. Williams, Network, and The eMERGE. "The Electronic Medical

Records and Genomics (eMERGE) Network: past, present, and future". In: *Genetics in Medicine* 15.10 (2013), pp. 761–771. ISSN: 1530-0366. DOI: 10.1038/gim.2013.72.

[74] Thomas R Gruber. "A translation approach to portable ontology specifications". In: *Knowledge acquisition* 5.2 (1993), pp. 199–220.

[75] Ramanathan V. Guha. "Towards A Model Theory for Distributed Representations". In: *2015 AAAI Spring Symposia, Stanford University, Palo Alto, California, USA, March 22-25, 2015*. AAAI Press, 2015. URL: http://www.aaai.org/ocs/index.php/SSS/SSS15/paper/view/10220.

[76] Víctor Gutiérrez-Basulto and Steven Schockaert. "From Knowledge Graph Embedding to Ontology Embedding? An Analysis of the Compatibility between Vector Space Representations and Rules". In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*. Ed. by Michael Thielscher, Francesca Toni, and Frank Wolter. AAAI Press, 2018, pp. 379–388. URL: https://aaai.org/ocs/index.php/KR/KR18/paper/view/18013.

[77] Mohamed Rouane Hacene, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. "Relational concept analysis: mining concept lattices from multi-relational data". In: *Ann. Math. Artif. Intell.* 67.1 (2013), pp. 81–108. DOI: 10.1007/s10472-012-9329-3. URL: https://doi.org/10.1007/s10472-012-9329-3.

[78] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. "Embedding Logical Queries on Knowledge Graphs". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 2030–2041. URL: http://papers.nips.cc/paper/7473-embedding-logical-queries-on-knowledge-graphs.

[79] 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 F. Sequeda, Steffen Staab, and Antoine Zimmermann. *Knowledge Graphs*. 2020. arXiv: 2003.02320. URL: https://arxiv.org/abs/2003.02320.

[80] Betsy L. Humphreys, Donald A. B. Lindberg, Harold M. Schoolman, and G. Octo Barnett. "Technical Milestone: The Unified Medical Language System: An Informatics Research Collaboration". In: *JAMIA* 5.1 (1998), pp. 1–11. DOI: 10.1136/jamia.1998.0050001. URL: https://doi.org/10.1136/jamia.1998.0050001.

[81] John P.A. Ioannidis. "To Replicate or Not to Replicate: The Case of Pharmacogenetic Studies". In: *Circulation: Cardiovascular Genetics* 6.4 (2013), pp. 413–418. DOI: 10.1161/CIRCGENETICS.113.000106. URL: https://www.ahajournals.org/doi/abs/10.1161/CIRCGENETICS.113.000106.

[82] Anne-Sophie Jannot, Eric Zapletal, Paul Avillach, Marie-France Mamzer, Anita Burgun, and Patrice Degoulet. "The Georges Pompidou University Hospital Clinical Data Warehouse: A 8-years follow-up experience". In: *I. J. Medical Informatics* 102 (2017), pp. 21–28. DOI: 10.1016/j.ijmedinf.2017.02.006. URL: https://doi.org/10.1016/j.ijmedinf.2017.02.006.

[83] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. *A Survey on Knowledge Graphs: Representation, Acquisition and Applications*. 2020. arXiv: 2002.00388. URL: https://arxiv.org/abs/2002.00388.

[84] Matthias Jurisch and Bodo Igler. "Graph-Convolution-Based Classification for Ontology Alignment Change Prediction". In: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019), Portoroz, Slovenia, June 2, 2019.* Ed. by Mehwish Alam, Davide Buscaldi, Michael Cochez, Francesco Osborne, Diego Reforgiato Recupero, and Harald Sack. Vol. 2377. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 11–20. URL: http://ceur-ws.org/Vol-2377/paper_2.pdf.

[85] Maulik R. Kamdar and Mark A. Musen. "PhLeGrA: Graph Analytics in Pharmacology over the Web of Life Sciences Linked Open Data". In: *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. Ed. by Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich. ACM, 2017, pp. 321–329. DOI: 10.1145/3038912.3052692. URL: https://doi.org/10.1145/3038912.3052692.

[86] Minoru Kanehisa and Susumu Goto. "KEGG: Kyoto Encyclopedia of Genes and Genomes". In: *Nucleic Acids Research* 28.1 (2000), pp. 27–30. DOI: 10.1093/nar/28.1.27. URL: https://doi.org/10.1093/nar/28.1.27.

[87] Jason H. Karnes, Robert M. Cronin, Jerome Rollin, Alexander Teumer, Claire Pouplard, Christian M. Shaffer, Carmelo Blanquicett, Erica A. Bowton, James D. Cowan, Jonathan D. Mosley, Sara L. Van Driest, Peter E. Weeke, Quinn S. Wells, Tamam Bakchoul, Joshua C. Denny, Andreas Greinacher, Yves Gruel, and Dan M. Roden. "A genome-wide association study of heparin-induced thrombocyto - penia using an electronic medical record". In: *Thromb Haemost* 113.04 (2015). 772, pp. 772–781. ISSN: 0340-6245. DOI: 10.1160/TH14-08-0670.

[88] Vivian K. Kawai, Andrew Cunningham, Susan I. Vear, Sara L. Van Driest, Abimbola Oginni, Hua Xu, Min Jiang, Chun Li, Joshua C. Denny, Christian Shaffer, Erica Bowton, Brian F. Gage, Wayne A. Ray, Dan M. Roden, and C. Michael Stein. "Genotype and risk of major bleeding during warfarin treatment". In: *Pharmacogenomics* 15.16 (2014), pp. 1973–1983. DOI: 10.2217/pgs.14.153.

[89] Mehdi Kaytoue, Sergei O. Kuznetsov, Amedeo Napoli, and Sébastien Duplessis. "Mining gene expression data with pattern structures in formal concept analysis". In: *Inf. Sci.* 181.10 (2011), pp. 1989–2001. DOI: 10.1016/j.ins.2010.07.007. URL: https://doi.org/10.1016/j.ins.2010.07.007.

[90] Yasar Khan, Muhammad Saleem, Muntazir Mehdi, Aidan Hogan, Qaiser Mehmood, Dietrich Rebholz-Schuhmann, and Ratnesh Sahay. "SAFE: SPARQL Federation over RDF Data Cubes with Access Control". In: *J. Biomed. Semant.* 8.1 (2017), 5:1–5:22. DOI: 10.1186/s13326-017-0112-6. URL: https://doi.org/10.1186/s13326-017-0112-6.

[91] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[92] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl.

[93] Markus Krötzsch. "Ontologies for Knowledge Graphs?" In: *Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, July 18-21, 2017*. Ed. by Alessandro Artale, Birte Glimm, and Roman Kontchakov. Vol. 1879. CEUR Workshop Proceedings. CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-1879/invited2.pdf.

[94] Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. "The SIDER database of drugs and side effects". In: *Nucleic Acids Research* 44.Database-Issue (2016), pp. 1075–1079. DOI: 10.1093/nar/gkv1075. URL: https://doi.org/10.1093/nar/gkv1075.

[95] Melissa J. Landrum, Jennifer M. Lee, George R. Riley, Wonhee Jang, Wendy S. Rubinstein, Deanna M. Church, and Donna R. Maglott. "ClinVar: public archive of relationships among sequence variation and human phenotype". In: *Nucleic Acids Research* 42.Database-Issue (2014), pp. 980–985. DOI: 10.1093/nar/gkt1113. URL: https://doi.org/10.1093/nar/gkt1113.

[96] Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. *PROV-O: The PROV Ontology*. W3C Recommendation. Apr. 2013. URL: https://www.w3.org/TR/prov-o/.

[97] Joël Legrand, Romain Gogdemir, Cédric Bousquet, Kevin Dalleau, Marie-Dominique Devignes, William Digan, Chia-Ju Lee, Ndeye-Coumba Ndiaye, Nadine Petitpain, Patrice Ringot, Malika Smaïl-Tabbone, Yannick Toussaint, and Adrien Coulet. "PGxCorpus, a manually annotated corpus for pharmacogenomics". In: *Scientific Data* 7.1 (2020), p. 3. ISSN: 2052-4463. DOI: 10.1038/s41597-019-0342-9. URL: https://doi.org/10.1038/s41597-019-0342-9.

[98] Fritz Lehmann and Rudolf Wille. "A Triadic Approach to Formal Concept Analysis". In: *Conceptual Structures: Applications, Implementation and Theory, Third International Conference on Conceptual Structures, ICCS '95, Santa Cruz, California, USA, August 14-18, 1995, Proceedings*. Ed. by Gerard Ellis, Robert Levinson, William Rich, and John F. Sowa. Vol. 954. Lecture Notes in Computer Science. Springer, 1995, pp. 32–43. DOI: 10.1007/3-540-60161-9_27. URL: https://doi.org/10.1007/3-540-60161-9_27.

[99] 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". In: *Semantic Web* 6.2 (2015), pp. 167–195. DOI: 10.3233/SW-140134. URL: https://doi.org/10.3233/SW-140134.

[100] Jean Lieber, Amedeo Napoli, Laszlo Szathmary, and Yannick Toussaint. "First Elements on Knowledge Discovery Guided by Domain Knowledge (KDDK)". In: *Concept Lattices and Their Applications, Fourth International Conference, CLA 2006, Tunis, Tunisia, October 30 - November 1, 2006, Selected Papers*. Ed. by Sadok Ben Yahia, Engelbert Mephu Nguifo, and Radim Belohlávek. Vol. 4923. Lecture Notes in Computer Science. Springer, 2006, pp. 22–41. DOI: 10.1007/978-3-540-78921-5_2. URL: https://doi.org/10.1007/978-3-540-78921-5_2.

[101] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. "Learning Entity and Relation Embeddings for Knowledge Graph Completion". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. Ed. by Blai Bonet and Sven Koenig. AAAI Press, 2015, pp. 2181–2187. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571.

[102] Claudia Marinica and Fabrice Guillet. "Knowledge-Based Interactive Postmining of Association Rules Using Ontologies". In: *IEEE Trans. Knowl. Data Eng.* 22.6 (2010), pp. 784–797. DOI: 10.1109/TKDE.2010.29. URL: https://doi.org/10.1109/TKDE.2010.29.

[103] Nicolas Matentzoglu, James Malone, Chris Mungall, and Robert Stevens. "MIRO: guidelines for minimum information for the reporting of an ontology". In: *J. Biomedical Semantics* 9.1 (2018), 6:1–6:13. DOI: 10.1186/s13326-017-0172-7. URL: https://doi.org/10.1186/s13326-017-0172-7.

[104] Julie A. McMurry, Nick Juty, Niklas Blomberg, Tony Burdett, Tom Conlin, Nathalie Conte, Mélanie Courtot, John Deck, Michel Dumontier, Donal K. Fellows, Alejandra Gonzalez-Beltran, Philipp Gormanns, Jeffrey Grethe, Janna Hastings, Jean-Karim Hériché, Henning Hermjakob, Jon C. Ison, Rafael C. Jimenez, Simon Jupp, John Kunze, Camille Laibe, Nicolas Le Novère, James Malone, Maria Jesus Martin, Johanna R. McEntyre, Chris Morris, Juha Muilu, Wolfgang Müller, Philippe Rocca-Serra, Susanna-Assunta Sansone, Murat Sariyar, Jacky L. Snoep, Stian Soiland-Reyes, Natalie J. Stanford, Neil Swainston, Nicole Washington, Alan R. Williams, Sarala M. Wimalaratne, Lilly M. Winfree, Katherine Wolstencroft, Carole Goble, Christopher J. Mungall, Melissa A. Haendel, and Helen Parkinson. "Identifiers for the 21st century: How to design, provision, and reuse persistent identifiers to maximize utility and impact of life science data". In: *PLOS Biology* 15.6 (June 2017), pp. 1–18. DOI: 10.1371/journal.pbio.2001414. URL: https://doi.org/10.1371/journal.pbio.2001414.

[105] Alistair Miles and Sean Bechhofer. *RDF Schema 1.1*. W3C Recommendation. Aug. 2009. URL: https://www.w3.org/TR/skos-reference/.

[106] George A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (1995), pp. 39–41. DOI: 10.1145/219717.219748. URL: http://doi.acm.org/10.1145/219717.219748.

[107] Justin J Miller. "Graph database applications and concepts with Neo4j". In: *Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*. Vol. 2324. 36. 2013.

[108] Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil A. Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. "Never-Ending Learning". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. Ed. by Blai Bonet and Sven Koenig. AAAI Press, 2015, pp. 2302–2310. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/10049.

[109] Sameh K. Mohamed, Vít Novácek, Pierre-Yves Vandenbussche, and Emir Muñoz. "Loss Functions in Knowledge Graph Embedding Models". In: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019), Portoroz, Slovenia, June 2, 2019*. Ed. by Mehwish Alam, Davide Buscaldi, Michael Cochez, Francesco Osborne, Diego Reforgiato Recupero, and Harald Sack. Vol. 2377. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 1–10. URL: http://ceur-ws.org/Vol-2377/paper_1.pdf.

[110] Misael Mongiovì, Diego Reforgiato Recupero, Aldo Gangemi, Valentina Presutti, and Sergio Consoli. "Merging open knowledge extracted from text with MERGILO". In: *Knowl.-Based Syst.* 108 (2016), pp. 155–167. DOI: 10.1016/j.knosys.2016.05.014. URL: https://doi.org/10.1016/j.knosys.2016.05.014.

[111] Pierre Monnin. "Discovering and Comparing Relational Knowledge, the Example of Pharmacogenomics". In: *Proceedings of the EKAW Doctoral Consortium 2018 co-located with the 21st International Conference on Knowledge Engineering and Knowledge Management (EKAW 2018), Nancy, France, November 13, 2018*. Ed. by Laura Hollink and Francesco Osborne. Vol. 2306. CEUR Workshop Proceedings. CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2306/paper5.pdf.

[112] Pierre Monnin, Emmanuel Bresso, Miguel Couceiro, Malika Smaïl-Tabbone, Amedeo Napoli, and Adrien Coulet. "Tackling scalability issues in mining path patterns from knowledge graphs: a preliminary study". In: *1st international conference "Algebras, graphs and ordered sets" (ALGOS 2020)*. Ed. by Miguel Couceiro, Pierre Monnin, and Amedeo Napoli. Nancy, France, Aug. 2020. URL: https://hal.inria.fr/hal-02913224/document.

[113] Pierre Monnin, Miguel Couceiro, Amedeo Napoli, and Adrien Coulet. "Knowledge-Based Matching of n-ary Tuples". In: *Ontologies and Concepts in Mind and Machine - 25th International Conference on Conceptual Structures, ICCS 2020, Bolzano, Italy, September 18–20, 2020, Proceedings*. Ed. by Mehwish Alam, Tanya Braun, and Bruno Yun. Vol. 12277. Lecture Notes in Computer Science. Springer, 2020, pp. 48–56. DOI: 10.1007/978-3-030-57855-8_4. URL: https://doi.org/10.1007/978-3-030-57855-8_4.

[114] Pierre Monnin, Clément Jonquet, Joël Legrand, Amedeo Napoli, and Adrien Coulet. "PGxO: A very lite ontology to reconcile pharmacogenomic knowledge units". In: *Methods, tools & platforms for Personalized Medicine in the Big Data Era, NETTAB 2017*. Vol. 5. PeerJ PrePrints. PeerJ, 2017, e3140. DOI: 10.7287/peerj.preprints.3140v1.

[115] Pierre Monnin, Joël Legrand, Graziella Husson, Patrice Ringot, Andon Tchechmedjiev, Clément Jonquet, Amedeo Napoli, and Adrien Coulet. "PGxO and PGxLOD: a reconciliation of pharmacogenomic knowledge of various provenances, enabling further comparison". In: *BMC Bioinformatics* 20-S.4 (2019), 139:1–139:16. DOI: 10.1186/s12859-019-2693-9.

[116] Pierre Monnin, Mario Lezoche, Amedeo Napoli, and Adrien Coulet. "Using Formal Concept Analysis for Checking the Structure of an Ontology in LOD: The Example of DBpedia". In: *Foundations of Intelligent Systems - 23rd International Symposium, ISMIS 2017, Warsaw, Poland, June 26-29, 2017, Proceedings*. Ed. by Marzena Kryszkiewicz, Annalisa Appice, Dominik Slezak, Henryk Rybinski, Andrzej Skowron, and Zbigniew W. Ras. Vol. 10352. Lecture Notes in Computer Science. Springer, 2017, pp. 674–683. DOI: 10.1007/978-3-319-60438-1_66. URL: https://doi.org/10.1007/978-3-319-60438-1_66.

[117] Pierre Monnin, Amedeo Napoli, and Adrien Coulet. "Discovering Subsumption Axioms with Concept Annotation". In: *Gestion de Données – Principes, Technologies et Applications (BDA 2017)*. 2017. URL: https://hal.inria.fr/hal-01671454/document.

[118] Pierre Monnin, Amedeo Napoli, and Adrien Coulet. "Combining Concept Annotation and Pattern Structures for Guiding Ontology Mapping". In: *Proceedings of the 6th International Workshop "What can FCA do for Artificial Intelligence"? co-located with International Joint Conference on Artificial Intelligence and European Conference on Artificial Intelligence (IJCAI/ECAI 2018), Stockholm, Sweden, July 13, 2018*. Ed. by Sergei O. Kuznetsov, Amedeo Napoli, and Sebastian Rudolph. Vol. 2149. CEUR Workshop Proceedings. CEUR-WS.org, 2018, pp. 117–126. URL: http://ceur-ws.org/Vol-2149/paper11.pdf.

[119] Pierre Monnin, Amedeo Napoli, and Adrien Coulet. "Data-Interlinking: the Seed of Knowledge Reconciliation in Pharmacogenomics". Presented at the Workshop "Symbolic methods for data-interlinking" co-located with the 21st International Conference on Knowledge Engineering and Knowledge Management (EKAW 2018), Nancy, France. Nov. 2018. URL: https://hal.inria.fr/hal-01955262/document.

[120] Pierre Monnin, Chedy Raïssi, Amedeo Napoli, and Adrien Coulet. "Knowledge Reconciliation with Graph Convolutional Networks: Preliminary Results". In: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019), Portoroz, Slovenia, June 2, 2019*. Ed. by Mehwish Alam, Davide Buscaldi, Michael Cochez, Francesco Osborne, Diego Reforgiato Recupero, and Harald Sack. Vol. 2377. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 47–56. URL: http://ceur-ws.org/Vol-2377/paper_6.pdf.

[121] J. D. Mosley, C. M. Shaffer, S. L. Van Driest, P. E. Weeke, Q. S. Wells, J. H. Karnes, D. R. Velez Edwards, W.-Q. Wei, P. L. Teixeira, L. Bastarache, D. C. Crawford, R. Li, T. A. Manolio, E. P. Bottinger, C. A. McCarty, J. G. Linneman, M. H. Brilliant, J. A. Pacheco, W. Thompson, R. L. Chisholm, G. P. Jarvik, D. R. Crosslin, D. S. Carrell, E. Baldwin, J. Ralston, E. B. Larson, J. Grafton, A. Scrol, H. Jouni, I. J. Kullo, G. Tromp, K. M. Borthwick, H. Kuivaniemi, D. J. Carey, M. D. Ritchie, Y. Bradford, S. S. Verma, C. G. Chute, A. Veluchamy, M. K. Siddiqui, C. N. A. Palmer, A. Doney, S. H. MahmoudPour, A. H. Maitland-van der Zee, A. D. Morris, J. C. Denny, and D. M. Roden. "A genome-wide association study identifies variants in KCNIP4 associated with ACE inhibitor-induced cough". In: *The Pharmacogenomics Journal* 16.3 (2016), pp. 231–237. ISSN: 1473-1150. DOI: 10.1038/tpj.2015.51.

[122] Mark A. Musen. "The protégé project: a look back and a look forward". In: *AI Matters* 1.4 (2015), pp. 4–12. DOI: 10.1145/2757001.2757003. URL: https://doi.org/10.1145/2757001.2757003.

[123] Antoine Neuraz, Laurent Chouchana, Georgia Malamut, Christine Le Beller, Denis Roche, Philippe Beaune, Patrice Degoulet, Anita Burgun, Marie-Anne Loriot, and Paul Avillach. "Phenome-Wide Association Studies on a Quantitative Trait: Application to TPMT Enzyme Activity and Thiopurine Therapy in Pharmacogenomics". In: *PLoS Computational Biology* 9.12 (2013). DOI: 10.1371/journal.pcbi.1003405. URL: https://doi.org/10.1371/journal.pcbi.1003405.

[124] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. "A Review of Relational Machine Learning for Knowledge Graphs". In: *Proceedings of the IEEE* 104.1 (2016), pp. 11–33. DOI: 10.1109/JPROC.2015.2483592. URL: https://doi.org/10.1109/JPROC.2015.2483592.

[125] Natalya F. Noy and Deborah L. McGuinness. *Ontology development 101: A guide to creating your first ontology.* Mar. 2001.

[126] Natalya Fridman Noy, Nigam H. Shah, Patricia L. Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clément Jonquet, Daniel L. Rubin, Margaret-Anne D. Storey, Christopher G. Chute, and Mark A. Musen. "BioPortal: ontologies and integrated data resources at the click of a mouse". In: *Nucleic Acids Research* 37.Web-Server-Issue (2009), pp. 170–173. DOI: 10.1093/nar/gkp440. URL: https://doi.org/10.1093/nar/gkp440.

[127] Natasha Noy, Alan Rector, Pat Hayes, and Chris Welty. "Defining N-ary Relations on the Semantic Web". In: *W3C working group note* 12.4 (2006).

[128] David J Odgers and Michel Dumontier. "Mining electronic health records using linked data". In: *AMIA Summits on Translational Science Proceedings* 2015 (2015), p. 217.

[129] *Ontology Alignment Evaluation Initiative.* URL: http://oaei.ontologymatching.org/ (visited on 07/27/2020).

[130] Ning Pang, Weixin Zeng, Jiuyang Tang, Zhen Tan, and Xiang Zhao. "Iterative Entity Alignment with Improved Neural Attribute Embedding". In: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2019) Co-located with the 16th Extended Semantic Web Conference 2019 (ESWC 2019), Portoroz, Slovenia, June 2, 2019.* Ed. by Mehwish Alam, Davide Buscaldi, Michael Cochez, Francesco Osborne, Diego Reforgiato Recupero, and Harald Sack. Vol. 2377. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 41–46. URL: http://ceur-ws.org/Vol-2377/paper_5.pdf.

[131] Heiko Paulheim. "Generating Possible Interpretations for Statistics from Linked Open Data". In: *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings.* Ed. by Elena Simperl, Philipp Cimiano, Axel Polleres, Óscar Corcho, and Valentina Presutti. Vol. 7295. Lecture Notes in Computer Science. Springer, 2012, pp. 560–574. DOI: 10.1007/978-3-642-30284-8_44. URL: https://doi.org/10.1007/978-3-642-30284-8_44.

[132] Heiko Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods". In: *Semantic Web* 8.3 (2017), pp. 489–508. DOI: 10.3233/SW-160218. URL: https://doi.org/10.3233/SW-160218.

[133] Heiko Paulheim. "Make Embeddings Semantic Again!" In: *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018.* Ed. by Marieke van Erp, Medha Atre, Vanessa López, Kavitha Srinivas, and Carolina Fortuna. Vol. 2180. CEUR Workshop Proceedings. CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2180/ISWC_2018_Outrageous_Ideas_paper_4.pdf.

[134] Heiko Paulheim and Johannes Fürnkranz. "Unsupervised generation of data mining features from linked open data". In: *2nd International Conference on Web Intelligence, Mining and Semantics, WIMS '12, Craiova, Romania, June 6-8, 2012.* Ed. by Dumitru Dan Burdescu, Rajendra Akerkar, and Costin Badica. ACM, 2012, 31:1–31:12. DOI: 10.1145/2254129.2254168. URL: https://doi.org/10.1145/2254129.2254168.

[135] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. "Scikit-learn: Machine Learning in Python". In: *J. Mach. Learn. Res.* 12 (2011), pp. 2825–2830. URL: http://dl.acm.org/citation.cfm?id=2078195.

[136] Núria Queralt-Rosinach, Janet Piñero González, Àlex Bravo, Ferran Sanz, and Laura I. Furlong. "DisGeNET-RDF: harnessing the innovative power of the Semantic Web to explore the genetic basis of diseases". In: *Bioinformatics* 32.14 (2016), pp. 2236–2238. DOI: 10.1093/bioinformatics/btw214. URL: https://doi.org/10.1093/bioinformatics/btw214.

[137] Andrea H Ramirez, Yaping Shi, Jonathan S Schildcrout, Jessica T Delaney, Hua Xu, Matthew T Oetjens, Rebecca L Zuvich, Melissa A Basford, Erica Bowton, Min Jiang, Peter Speltz, Raquel Zink, James Cowan, Jill M Pulley, Marylyn D Ritchie, Daniel R Masys, Dan M Roden, Dana C Crawford, and Joshua C Denny. "Predicting warfarin dosage in European–Americans and African–Americans using DNA samples linked to an electronic health record". In: *Pharmacogenomics* 13.4 (2012), pp. 407–418. DOI: 10.2217/pgs.11.164.

[138] M V Relling, E E Gardner, W J Sandborn, K Schmiegelow, C-H Pui, S W Yee, C M Stein, M Carrillo, W E Evans, J K Hicks, M Schwab, and T E Klein. "Clinical Pharmacogenetics Implementation Consortium Guidelines for Thiopurine Methyltransferase Genotype and Thiopurine Dosing: 2013 Update". In: *Clinical Pharmacology & Therapeutics* 93.4 (2013), pp. 324–325. DOI: 10.1038/clpt.2013.4.

[139] Thomas C. Rindflesch, Halil Kilicoglu, Marcelo Fiszman, Graciela Rosemblat, and Dongwook Shin. "Semantic MEDLINE: An advanced information management application for biomedicine". In: *Inf. Services and Use* 31.1-2 (2011), pp. 15–21. DOI: 10.3233/ISU-2011-0627. URL: https://doi.org/10.3233/ISU-2011-0627.

[140] Petar Ristoski and Heiko Paulheim. "A Comparison of Propositionalization Strategies for Creating Features from Linked Open Data". In: *Proceedings of the 1st Workshop on Linked Data for Knowledge Discovery co-located with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2014), Nancy, France, September 19th, 2014.* Ed. by Ilaria Tiddi, Mathieu d'Aquin, and Nicolas Jay. Vol. 1232. CEUR Workshop Proceedings. CEUR-WS.org, 2014. URL: http://ceur-ws.org/Vol-1232/paper1.pdf.

[141] Petar Ristoski and Heiko Paulheim. "Feature Selection in Hierarchical Feature Spaces". In: *Discovery Science - 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings.* Ed. by Saso Dzeroski, Pance Panov, Dragi Kocev, and Ljupco Todorovski. Vol. 8777. Lecture Notes in Computer Science. Springer, 2014, pp. 288–300. DOI: 10.1007/978-3-319-11812-3_25. URL: https://doi.org/10.1007/978-3-319-11812-3_25.

[142] Petar Ristoski and Heiko Paulheim. "RDF2Vec: RDF Graph Embeddings for Data Mining". In: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I.* Ed. by Paul T. Groth, Elena Simperl, Alasdair J. G. Gray, Marta Sabou, Markus Krötzsch, Freddy Lécué, Fabian Flöck, and Yolanda Gil. Vol. 9981. Lecture Notes in Computer Science. 2016, pp. 498–514. DOI: 10.1007/978-3-319-46523-4_30. URL: https://doi.org/10.1007/978-3-319-46523-4_30.

[143] Petar Ristoski and Heiko Paulheim. “Semantic Web in data mining and knowledge discovery: A comprehensive survey”. In: *J. Web Semant.* 36 (2016), pp. 1–22. DOI: 10.1016/j.websem.2016.01.001. URL: https://doi.org/10.1016/j.websem.2016.01.001.

[144] DM Roden, JM Pulley, MA Basford, GR Bernard, EW Clayton, JR Balser, and DR Masys. “Development of a Large-Scale De-Identified DNA Biobank to Enable Personalized Medicine”. In: *Clinical Pharmacology & Therapeutics* 84.3 (2008), pp. 362–369. DOI: 10.1038/clpt.2008.89.

[145] Matthias Samwald, José Antonio Miñarro-Giménez, Richard D. Boyce, Robert R. Freimuth, Klaus-Peter Adlassnig, and Michel Dumontier. “Pharmacogenomic knowledge representation, reasoning and genome-based clinical decision support based on OWL 2 DL ontologies”. In: *BMC Med. Inf. & Decision Making* 15 (2015), p. 12. DOI: 10.1186/s12911-015-0130-1. URL: https://doi.org/10.1186/s12911-015-0130-1.

[146] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. “Modeling Relational Data with Graph Convolutional Networks”. In: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*. Ed. by Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 593–607. DOI: 10.1007/978-3-319-93417-4_38. URL: https://doi.org/10.1007/978-3-319-93417-4_38.

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

[148] Pierre Senellart. “Provenance and Probabilities in Relational Databases”. In: *SIGMOD Rec.* 46.4 (2017), pp. 5–15. DOI: 10.1145/3186549.3186551. URL: https://doi.org/10.1145/3186549.3186551.

[149] Luciano Serafini and Artur S. d’Avila Garcez. “Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge”. In: *Proceedings of the 11th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy’16) co-located with the Joint Multi-Conference on Human-Level Artificial Intelligence (HLAI 2016), New York City, NY, USA, July 16-17, 2016.* Ed. by Tarek R. Besold, Luís C. Lamb, Luciano Serafini, and Whitney Tabor. Vol. 1768. CEUR Workshop Proceedings. CEUR-WS.org, 2016. URL: http://ceur-ws.org/Vol-1768/NESY16_paper3.pdf.

[150] Baoxu Shi and Tim Weninger. “Discriminative predicate path mining for fact checking in knowledge graphs”. In: *Knowl.-Based Syst.* 104 (2016), pp. 123–133. DOI: 10.1016/j.knosys.2016.04.015. URL: https://doi.org/10.1016/j.knosys.2016.04.015.

[151] Lian Shi, Yannick Toussaint, Amedeo Napoli, and Alexandre Blansché. “Mining for Reengineering: An Application to Semantic Wikis Using Formal and Relational Concept Analysis”. In: *The Semanic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29 - June 2, 2011, Proceedings, Part II*. Ed. by Grigoris Antoniou, Marko Grobelnik, Elena Paslaru Bontas Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Z. Pan. Vol. 6644. Lecture Notes in Computer Science. Springer, 2011, pp. 421–435. DOI: 10.1007/978-3-642-21064-8_29. URL: https://doi.org/10.1007/978-3-642-21064-8_29.

[152] Amit Singhal. *Introducing the Knowledge Graph: things, not strings.* May 2012. URL: https://www.blog.google/products/search/introducing-knowledge-graph-things-not/ (visited on 07/28/2020).

[153] Barry Smith, Werner Ceusters, Bert Klagges, Jacob Köhler, Anand Kumar, Jane Lomax, Chris Mungall, Fabian Neuhaus, Alan L. Rector, and Cornelius Rosse. "Relations in biomedical ontologies". In: *Genome biology* 6.5 (2005), R46.

[154] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. "Reasoning With Neural Tensor Networks for Knowledge Base Completion". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.* Ed. by Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger. 2013, pp. 926–934. URL: http://papers.nips.cc/paper/5028-reasoning-with-neural-tensor-networks-for-knowledge-base-completion.

[155] Ramakrishnan Srikant and Rakesh Agrawal. "Mining Generalized Association Rules". In: *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland.* Ed. by Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio. Morgan Kaufmann, 1995, pp. 407–419. URL: http://www.vldb.org/conf/1995/P407.PDF.

[156] Josua Stadelmaier and Sebastian Padó. "Modeling Paths for Explainable Knowledge Base Completion". In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP.* 2019, pp. 147–157.

[157] Pontus Stenetorp, Sampo Pyysalo, Goran Topic, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. "brat: a Web-based Tool for NLP-Assisted Text Annotation". In: *EACL 2012, 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012.* Ed. by Walter Daelemans, Mirella Lapata, and Lluís Màrquez. The Association for Computer Linguistics, 2012, pp. 102–107. URL: https://www.aclweb.org/anthology/E12-2021/.

[158] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. "PARIS: Probabilistic Alignment of Relations, Instances, and Schema". In: *PVLDB* 5.3 (2011), pp. 157–168. DOI: 10.14778/2078331.2078332. URL: http://www.vldb.org/pvldb/vol5/p157_fabianmsuchanek_vldb2012.pdf.

[159] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: a core of semantic knowledge". In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007.* Ed. by Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy. ACM, 2007, pp. 697–706. DOI: 10.1145/1242572.1242667. URL: https://doi.org/10.1145/1242572.1242667.

[160] The W3C SPARQL Working Group. *SPARQL 1.1 Overview.* W3C Recommendation. Mar. 2013. URL: https://www.w3.org/TR/sparql11-overview/.

[161] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. "Complex Embeddings for Simple Link Prediction". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016.* Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 2071–2080. URL: http://proceedings.mlr.press/v48/trouillon16.html.

[162] Yoshimasa Tsuruoka, Makoto Miwa, Kaisei Hamamoto, Jun'ichi Tsujii, and Sophia Ananiadou. "Discovering and visualizing indirect associations between biomedical concepts". In: *Bioinformatics [ISMB/ECCB]* 27.13 (2011), pp. 111–119. DOI: 10.1093/bioinformatics/btr214. URL: https://doi.org/10.1093/bioinformatics/btr214.

[163] Sara L. Van Driest, Tracy L. McGregor, Digna R. Velez Edwards, Ben R. Saville, Terrie E. Kitchner, Scott J. Hebbring, Murray Brilliant, Hayan Jouni, Iftikhar J. Kullo, C. Buddy Creech, Prince J. Kannankeril, Susan I. Vear, Kyle B. Brothers, Erica A. Bowton, Christian M. Shaffer, Neelam Patel, Jessica T. Delaney, Yuki Bradford, Sarah Wilson, Lana M. Olson, Dana C. Crawford, Amy L. Potts, Richard H. Ho, Dan M. Roden, and Josh C. Denny. "Genome-Wide Association Study of Serum Creatinine Levels during Vancomycin Therapy". In: *PLOS ONE* 10.6 (June 2015), pp. 1–14. DOI: 10.1371/journal.pone.0127791.

[164] Gilles Vandewiele, Bram Steenwinckel, Femke Ongenae, and Filip De Turck. "Inducing a Decision Tree with Discriminative Paths to Classify Entities in a Knowledge Graph". In: *Proceedings of the 4th International Workshop on Semantics-Powered Data Mining and Analytics co-located with the 18th International Semantic Web Conference (ISWC 2019), Aukland, New Zealand, October 27, 2019.* Ed. by Zhe He, Jiang Bian, Cui Tao, and Rui Zhang. Vol. 2427. CEUR Workshop Proceedings. CEUR-WS.org, 2019. URL: http://ceur-ws.org/Vol-2427/SEPDA_2019_paper_3.pdf.

[165] Denny Vrandecic and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase". In: *Commun. ACM* 57.10 (2014), pp. 78–85. DOI: 10.1145/2629489. URL: https://doi.org/10.1145/2629489.

[166] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition).* W3C Recommendation. Dec. 2012. URL: https://www.w3.org/TR/owl-overview/.

[167] PeiFeng Wang, Jialong Han, Chenliang Li, and Rong Pan. "Logic Attention Based Neighborhood Aggregation for Inductive Knowledge Graph Embedding". In: *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.* AAAI Press, 2019, pp. 7152–7159. URL: https://aaai.org/ojs/index.php/AAAI/article/view/4698.

[168] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. "Knowledge Graph Embedding: A Survey of Approaches and Applications". In: *IEEE Trans. Knowl. Data Eng.* 29.12 (2017), pp. 2724–2743. DOI: 10.1109/TKDE.2017.2754499. URL: https://doi.org/10.1109/TKDE.2017.2754499.

[169] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. "Knowledge Graph Embedding by Translating on Hyperplanes". In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.* Ed. by Carla E. Brodley and Peter Stone. AAAI Press, 2014, pp. 1112–1119. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531.

[170] 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, Brussels, Belgium, October 31 - November 4, 2018.* Ed. by Ellen Riloff, David Chiang, Julia Hockenmaier,

and Jun'ichi Tsujii. Association for Computational Linguistics, 2018, pp. 349–357. DOI: 10.18653/v1/d18-1032. URL: https://doi.org/10.18653/v1/d18-1032.

[171] Chih-Hsuan Wei, Hung-Yu Kao, and Zhiyong Lu. "PubTator: a web-based text mining tool for assisting biocuration". In: *Nucleic Acids Research* 41.Webserver-Issue (2013), pp. 518–522. DOI: 10.1093/nar/gkt441. URL: https://doi.org/10.1093/nar/gkt441.

[172] Quinn S. Wells, Olivia J. Veatch, Joshua P. Fessel, Aron Y. Joon, Rebecca T. Levinson, Jonathan D. Mosley, Elizabeth P. Held, Chase S. Lindsay, Christian M. Shaffer, Peter E. Weeke, Andrew M. Glazer, Kevin R. Bersell, Sara L. Van Driest, Jason H. Karnes, Marcia A. Blair, Lore W. Lagrone, Yan R. Su, Erica A. Bowton, Ziding Feng, Bonnie Ky, Daniel J. Lenihan, Michael J. Fisch, Joshua C. Denny, and Dan M. Roden. "Genome-wide association and pathway analysis of left ventricular function after anthracycline exposure in adults". In: *Pharmacogenetics and Genomics* 27.7 (2017). ISSN: 1744-6872.

[173] Michelle Whirl-Carrillo, Ellen M. McDonagh, J.M. Hebert, Li Gong, K. Sangkuhl, C.F. Thorn, Russ B. Altman, and Teri E. Klein. "Pharmacogenomics knowledge for personalized medicine". In: *Clinical pharmacology and therapeutics* 92.4 (2012), p. 414.

[174] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J. G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A. C. 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. "The FAIR Guiding Principles for scientific data management and stewardship". In: *Scientific Data* 3.1 (2016), p. 160018. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.18. URL: https://doi.org/10.1038/sdata.2016.18.

[175] David S. Wishart, Craig Knox, Anchi Guo, Dean Cheng, Savita Shrivastava, Dan Tzur, Bijaya Gautam, and Murtaza Hassanali. "DrugBank: a knowledgebase for drugs, drug actions and drug targets". In: *Nucleic Acids Research* 36.Database-Issue (2008), pp. 901–906. DOI: 10.1093/nar/gkm958. URL: https://doi.org/10.1093/nar/gkm958.

[176] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. "Quality assessment for Linked Data: A Survey". In: *Semantic Web* 7.1 (2016), pp. 63–93. DOI: 10.3233/SW-150175. URL: https://doi.org/10.3233/SW-150175.

[177] Xuan Zhou and James Geller. "Raising, to enhance rule mining in web marketing with the use of an ontology". In: *Data Mining with Ontologies: Implementations, Findings, and Frameworks*. IGI Global, 2008, pp. 18–36.

Résumé

Dans le Web des données, des graphes de connaissances de plus en plus nombreux sont simultanément publiés, édités, et utilisés par des agents humains et logiciels. Cette large adoption rend essentielles les tâches d'*appariement* et de *fouille*. L'*appariement* identifie des unités de connaissances équivalentes, plus spécifiques ou similaires au sein et entre graphes de connaissances. Cette tâche est cruciale car la publication et l'édition parallèles peuvent mener à des graphes de connaissances co-existants et complémentaires. Cependant, l'hétérogénéité inhérente aux graphes de connaissances (*e.g.*, granularité, vocabulaires, ou complétude) rend cette tâche difficile. Motivés par une application en pharmacogénomique, nous proposons deux approches pour apparier des relations n-aires représentées au sein de graphes de connaissances : une méthode symbolique à base de règles et une méthode numérique basée sur le plongement de graphe. Nous les expérimentons sur PGxLOD, un graphe de connaissances que nous avons construit de manière semi-automatique en intégrant des relations pharmacogénomiques de trois sources du domaine. La tâche de *fouille* permet quant à elle de découvrir de nouvelles unités de connaissances à partir des graphes de connaissances. Leur taille croissante et leur nature combinatoire entraînent des problèmes de passage à l'échelle que nous étudions dans le cadre de la fouille de patrons de chemins. Nous proposons également l'annotation de concepts, une méthode d'amélioration des graphes de connaissances qui étend l'Analyse Formelle de Concepts, un cadre mathématique groupant des entités en fonction de leurs attributs communs. Au cours de tous nos travaux, nous nous sommes particulièrement intéressés à tirer parti des connaissances de domaines formalisées au sein d'ontologies qui peuvent être associées aux graphes de connaissances. Nous montrons notamment que, lorsqu'elles sont prises en compte, ces connaissances permettent de réduire l'impact des problèmes d'hétérogénéité et de passage à l'échelle dans les tâches d'appariement et de fouille.

Mots-clés : Ontologie, Tuple n-aire, Préordre, Plongement de graphe, Patron de chemin, Analyse Formelle de Concepts.

Abstract

In the Web of data, an increasing number of knowledge graphs are concurrently published, edited, and accessed by human and software agents. Their wide adoption makes key the two tasks of *matching* and *mining*. First, *matching* consists in identifying equivalent, more specific, or somewhat similar units within and across knowledge graphs. This task is crucial since concurrent publication and edition may result in coexisting and complementary knowledge graphs. However, this task is challenging because of the inherent heterogeneity of knowledge graphs, *e.g.*, in terms of granularities, vocabularies, and completeness. Motivated by an application in pharmacogenomics, we propose two approaches to match n-ary relationships represented in knowledge graphs: a symbolic rule-based approach and a numeric approach using graph embedding. We experiment on PGxLOD, a knowledge graph that we semi-automatically built by integrating pharmacogenomic relationships from three distinct sources of this domain. Second, *mining* consists in discovering new and useful knowledge units from knowledge graphs. Their increasing size and combinatorial nature entail scalability issues, which we address in the mining of path patterns. We also propose Concept Annotation, a refinement approach extending Formal Concept Analysis, a mathematical framework that groups entities based on their common attributes. Throughout all our works, we particularly focus on taking advantage of domain knowledge in the form of ontologies that can be associated with knowledge graphs. We show that, when considered, such domain knowledge alleviates heterogeneity and scalability issues in matching and mining approaches.

Keywords: Ontology, n-ary Tuple, Preorder, Graph Embedding, Path Pattern, Formal Concept Analysis.