Identity in RDF knowledge graphs : propagation of properties between contextually identical entities

Pierre-Henri Paris

Sorbonne Université

École doctorale Informatique, Télécommunications et Électronique (Paris)

Cédric / ISID

Identity in RDF knowledge graphs

Propagation of properties between contextually identical entities

Par Pierre-Henri Paris

Thèse de doctorat d'Informatique

Présentée et soutenue publiquement le 17 juin 2020

THÈSE dirigée par

Mme Si-said Cherfi Samira	*Professeure des Universités, CNAM*
M. Hamdi Fayçal	*Maître de Conférences, CNAM*

RAPPORTEURS

Mme Pernelle Nathalie	*Professeure des Universités, Université Sorbonne Paris Nord*
M. d'Aquin Mathieu	*Professeur des Universités, NUI Galway*

EXAMINATEURS

Mme Faron-Zucker Catherine	*Maître de Conférences (HDR), Université Nice Sophia Antipolis*
Mme Zanni-Merk Cecilia	*Professeure des Universités, INSA Rouen*
M. Amann Bernd	*Professeur des Universités, Sorbonne Université*
M. Kotzinos Dimitris	*Professeur des Universités, ENSEA*

Abstract

Due to its strict semantics, the *owl:sameAs* property is often misused. Nowadays, with the large number of knowledge graphs published on the Web of Data and, more importantly, their numerous interconnections using the *owl:sameAs* property, this situation has become worrying. Indeed, two entities linked by the *owl:sameAs* property must be identical in all possible and imaginable contexts, but this is not always the case. Since identity is context-dependent, all property-value pairs of the first entity could be wrongly transferred (propagated) to the other entity, leading to a deterioration of data quality. Therefore, in this thesis we are particularly interested in how some properties may or may not be propagated between identical entities in a given context. As a first step, we conducted a large-scale study on the presence of semantics in knowledge graphs since specific semantic characteristics allow us to deduce identity links. This study naturally led us to build an ontology to describe the semantic content of a knowledge graph. Moreover, we observed that some properties are more important than others to determine the (contextual) identity between two entities. For this, we proposed an interlinking approach based on the logic of semantic definitions, and on the predominance of certain properties to characterize the identity relationship between two entities. Then, we proposed an approach based on sentence embedding to compute the properties that can be propagated in a context. This propagation approach allows the expansion of SPARQL queries and, ultimately, the increasing of the completeness of query results. Finally, we proposed a tool to measure the impact of the propagation on entity completeness at a schema level. The different approaches proposed in this thesis have been implemented and evaluated through experiments on real-world data.

Keywords: Semantic Web, Contextual Identity, Property Propagation, Knowledge Graph, RDF, OWL, Ontology, Instance Matching, Linked Data, Sentence Embedding, Completeness, Conceptual Schema Mining.

Résumé

En raison de sa sémantique stricte, la propriété *owl:sameAs* est souvent mal utilisée. Aujourd'hui, avec le grand nombre de graphes de connaissances publiés sur le Web des Données et, surtout, leurs nombreuses interconnexions utilisant la propriété *owl:sameAs*, cette situation est devenue préoccupante. En effet, deux entités liées par la propriété *owl:sameAs* doivent être identiques dans tous les contextes possibles et imaginables, mais ce n'est pas toujours le cas. L'identité étant dépendante du contexte, toutes les paires propriété-valeur de la première entité pourraient être transférées (propagées) à tort à l'autre entité, ce qui entraînerait une détérioration de la qualité des données. C'est pourquoi, dans cette thèse, nous nous intéressons particulièrement à la manière dont certaines propriétés peuvent ou non être propagées entre des entités identiques dans un contexte donné. Dans un premier temps, nous avons mené une étude à grande échelle sur la présence de la sémantique dans les graphes de connaissance puisque des caractéristiques sémantiques spécifiques nous permettent de déduire des liens d'identité. Cette étude nous a naturellement conduit à construire une ontologie pour décrire le contenu sémantique d'un graphe de connaissance. De plus, nous avons observé que certaines propriétés sont plus importantes que d'autres pour déterminer l'identité (contextuelle) entre deux entités. Pour cela, nous avons proposé une approche d'interconnexion basée sur la logique des définitions sémantiques, et sur la prédominance de certaines propriétés pour caractériser la relation d'identité entre deux entités. Ensuite, nous avons proposé une approche basée sur les plongements de phrases pour calculer les propriétés qui peuvent être propagées dans un contexte. Cette approche de propagation permet l'expansion des requêtes SPARQL et, à terme, l'augmentation de la complétude des résultats des requêtes. Enfin, nous avons proposé un outil permettant de mesurer l'impact de la propagation sur la complétude des entités au niveau du schéma.

Les différentes approches proposées dans cette thèse ont été mises en œuvre et évaluées par des expériences sur des données réelles.

Mots clés : Web Sémantique, Identité Contextuelle, Propagation de Propriétés, Graphe de Connaissances, RDF, OWL, Ontologie, Appariement d'Instances, Données Liées, Plongement de Phrases, Complétude, Extraction de Schéma Conceptuel.

Acknowledgements

Cette thèse, je ne l'ai pas réalisée tout seul, loin de là. C'est un travail d'équipe et de nombreuses personnes m'ont aidé avant, pendant et, qui sait, m'aideront peut-être après.

Je tiens donc à commencer mes remerciements par ceux que je dois à mes deux guides pendant ces presque quatre années, ma directrice de thèse, la Pr Samira Si-said Cherfi, et mon co-encadrant, le Dr Fayçal Hamdi. Ils ont su aiguiser en moi mes compétences de chercheurs, m'orienter lorsque j'étais perdu et, je l'espère, faire naître un chercheur compétent. Je suis certain que nos chemins se croiseront à nouveau parmi les nombreux projets qui jalonneront encore nos chemins de chercheurs. Samira et Fayçal, merci.

Je tiens à remercier le Dr Nobal Niraula, au contact de qui j'ai beaucoup appris lors d'un projet bref, mais intense. Mes plus sincères remerciements aussi vont aux Drs Nathalie Abadie et Carmen Brando. Elles ont été les premières à me guider sur la voie de la recherche et j'ai pu grâce à elles confirmer mon envie de commencer un doctorat. Mes sincères remerciements vont aussi à Nadira Lammari qui a su me conseiller et m'a permis de découvrir mon appétence pour l'enseignement.

J'adresse aussi mes remerciements aux Prs Nathalie Pernelle et Mathieu d'Aquin qui ont accepté d'être les rapporteurs des fruits de mon travail. Je remercie en outre les Pr Philippe Rigaux et une seconde fois Nathalie Pernelle pour leurs participation à mes comités de suivis et leurs remarques constructives. Je suis reconnaissant aux Prs Catherine Faron-Zucker, Cecilia Zanni-Merk, Bernd Amann et Dimitris Kotzinos d'avoir accepté d'être membres de mon jury de thèse. C'est un grand honneur que vous me faites.

Mes remerciements vont aussi à toute l'équipe ISID du laboratoire CEDRIC. Non seulement, car j'ai pu reprendre un master proposé par l'équipe qui m'a amené là où j'en

suis aujourd'hui, mais aussi pour les discussions que nous avons eues et les conseils qu'ils ont su me prodiguer durant ces années. Je n'oublie pas non plus toute l'équipe administrative, et notamment Meryem Hara, Sandra Bosse et Alexandre Lescault, qui m'ont apporté leur soutien indéfectible et amical.

Cela a été une immense joie et un immense honneur aussi de partager le même bureau que mes loyaux camarades : Dr Noura Herradi, Dr Odette Sangupamba Mwilu, Dr Quentin Grossetti, Dr Abdelbadie Belmouhcine, Dr Fatma Hannou et Dr Subhi Issa. Merci Noura pour nos conversations qui m'ont beaucoup manqué après ton départ. Merci Odette pour ta sagesse, ta douceur et ta bonne humeur. Merci Quentin pour ton ouverture, tes conseils scientifiques et sur les restaurants, et les quelques morceaux de musique que tu m'auras fait découvrir. Merci Badie pour ces débats qui n'en étaient pas mais qui en étaient quand même. Merci Fatma pour toute cette expérience que tu as su partager avec moi en si peu de temps. J'espère sincèrement que nos chemins professionnels se recroiseront. Enfin, merci Subhi. Merci à toi pour ton amitié inébranlable, pour avoir supporté mes états d'âme, pour l'exemple que tu es. Sois toujours certain de ta valeur.

Mille mercis aussi à vous, Ayoub et Khalil, mes très chers padawans. C'est en grande partie votre influence qui m'a donné le courage de reprendre des études. Nous nous sommes élevés tous les trois, les uns les autres, grâce à notre rencontre. Merci.
Ces remerciements n'en seraient pas si je ne manifestais pas non plus ma gratitude envers mes amis fidèles, ma famille de cœur : Anaël, Anne-Flore, Aurel, Aurélie, Brice, Gaëlle, Hélène, Louise, Lucile, Max, Oliv (ne cherchez pas, vous êtes classés par ordre alphabétique). Vous avez su me supporter, voire me porter, dans les moments difficiles tout au long de ma vie. Et nous avons ri, et nous avons partagé, et nous avons vécu et grandi ensemble. Nos péripéties au collège, au lycée, à l'université, au boulot, dans la cave, en vacances et ailleurs, resteront à jamais gravées dans ma mémoire. Vous avez été avec moi. Vous êtes avec moi. Vous serez avec moi.

Ma reconnaissance est aussi immense pour ma famille. Ma petite sœur chérie, Fanny, regarde le chemin que nous avons parcouru. Je suis immensément content de t'avoir dans ma vie et si tu es moitié aussi fière de moi que je le suis de toi, alors je suis heureux. Je remercie mon père pour son soutien et la confiance qu'il a toujours eue en mes capacités.

Je suis heureux que nous nous soyons retrouvés et que tu fasses partie de ma vie. Viviane, tu le sais, tu es comme une deuxième mère pour moi. Je sais ce que je te dois et j'espère t'avoir rendue fière. Mameu, même si tu ne peux plus lire ces lignes, je pense à toi tous les jours et je ne te remercierai jamais assez pour tout l'amour et la force que tu m'as apportée. Philippe et Annie, je vous remercie d'abord d'avoir donné au monde, et à moi par la même occasion, la chance de connaître Camille. Je vous remercie aussi de m'avoir accueilli dans votre famille comme vous l'avez fait. C'est pour moi un honneur et un plaisir de vous considérer réciproquement comme ma famille. Je tiens aussi à te remercier Marie-Pierre, déjà de m'avoir embauché en des temps reculés, à une époque durant laquelle je me sentais perdu. Mais aussi pour tout ce que tu apportes au quotidien à notre petite équipe. Tu es la meilleure des belles-sœurs et une amie très chère.

Je souhaite aussi remercier celles et ceux qui ne sont plus là. Maman, Papeu, Claude. Vous avez été chacun à votre façon des piliers de ma vie, des moteurs et des exemples à suivre. Cette thèse n'aurait pas été possible sans vous et, quoi que j'aie fait ou que je fasse, vous m'accompagnez. Ma vie, dans toutes ses composantes, est à jamais marquée par votre empreinte.

Enfin, pour finir sur une note plus joyeuse ces longs remerciements, je termine en beauté par les deux (pour l'instant ?) joyaux de ma vie, Camille et Sélène. Sélène, tu es la petite fille la plus formidable de l'univers connu et inconnu. Être ton père est l'expérience la plus bouleversante et la plus importante de ma vie. J'ai bien peur de ne pouvoir utiliser que des lieux communs pour décrire ce que je ressens pour toi. Mais, ces lieux communs prennent tout leur sens à l'instant où l'on devient parent, et tu les comprendras un jour, lorsque tu seras toi-même passée de l'autre côté du miroir. Nous danserons ensemble pour l'éternité sur "Vier Stücke für Xylophon" de Carl Orff. Nous chanterons ensemble pour l'éternité nos chansons connues ou inventées. Je serai toujours là, je serai toujours ton papa qui t'aime. Camille, j'espère que tu sais, que tu sens, que tu ressens, que tu comprends tout ce que tu m'as apporté, tout ce que tu m'as donné. Bien sûr, il y a ton soutien pendant cette thèse, avant aussi, mais il y a surtout tellement plus. Tu as amené la lumière dans ma vie. Toutes ces épreuves, ces moments de doutes, ces larmes, ces sourires, ces rires, cette complicité, ces moments de félicités, je les chérirai à jamais comme mon bien le plus précieux. Tu es

l'épine dorsale du chercheur, du père et de l'homme que je suis. Notre équipe est invincible et immortelle, et donc, cette thèse, c'est à toi que je la dédie.

Contents

List of Tables

List of Figures

Résumé long en français

La littérature scientifique croît depuis des siècles et elle croît de plus en plus vite. Face à ce constat, dès le milieu des années 40, Vannevar Bush a soulevé les problèmes de l'accès à ces connaissances et à leur utilisation par les scientifiques. Il a proposé une machine conceptuelle appelée Memex ([Bush 1945]), qui est une sorte de prothèse mémorielle permettant le stockage et la consultation de documents et de créer des liens entre eux afin d'aider l'utilisateur à reprendre sa consultation là où il en était. Ensuite, en 1965, Ted Nelson ([Nelson 1965]) a complété cette idée en créant les liens hypertextes entre documents, fichiers, etc. Il s'agit d'un des éléments fondateurs du Web encore à venir. À la fin des années 80, Tim Berners-Lee [Berners-Lee 1989] propose d'aller plus loin en distribuant les documents sur des machines différentes, tout en liant ces documents à l'aide de liens hypertextes. C'est la naissance du Web. Ce dernier repose sur trois briques fondamentales, à savoir *(i)* le protocole HTTP pour la communication entre machines ou clients et serveurs, *(ii)* le langage HTML pour exposer les données aux clients et enfin *(iii)* les URL pour l'identification et l'adressage. Le W3C est l'organisme responsable des différents standards liés au Web. De nos jours, de très nombreuses données sont publiées chaque jour sur le Web sous différentes formes. La majorité de ces données est publiée sous forme de pages HTML et est donc très peu structurée. Des formats tels que XML, JSON, PDF ou CSV sont aussi régulièrement utilisés, notamment par les organisations, permettant ainsi aux données d'être structurées ou semi structurées.

Le Web sémantique [Berners-Lee, Hendler, Lassila, et al. 2001] est une extension du Web proposée par Tim Berners-Lee qui permet d'aller un cran au-delà dans la structuration des données. Cette extension repose sur une pile technologique gérée par le W3C et qui permet d'exprimer les données et leur schéma sous forme de graphe. C'est ce qu'on appelle

le modèle de graphe RDF. Il est composé de trois éléments de bases que sont *(i)* les ressources (IRI), *(ii)* les littéraux et *(iii)* les noeuds anonymes (quantification existentielle). Les ressources en sont l'élément principal puisqu'elles permettent d'identifier tout objet ou concept du monde réel. Elles permettent de décrire tout et n'importe quoi sous forme de triplets dont le sujet, c'est-à-dire le premier élément, est une entité que l'on souhaite décrire. Cette entité est identifiée par son IRI. Le second élément du triplet est lui-même une ressource correspondant à une caractéristique de la chose décrite, comme, par exemple, une adresse, un nom ou une latitude. C'est la propriété du triplet. Enfin, le troisième élément du triplet est l'objet, qui représente la valeur de la propriété. Cet objet est au choix une ressource, permettant ainsi de lier deux ressources entre elles, ou un littéral, c'est-à-dire une forme lexicale ayant un type tel qu'un entier ou une date exprimée à l'aide d'un IRI. De plus, par l'intermédiaire des langages RDFS et OWL 2, il est possible de décrire le schéma de ces données, c'est-à-dire les propriétés utilisées dans les triplets, ainsi que les classes (les types) que peuvent avoir les ressources. Certaines classes et propriétés peuvent avoir une sémantique permettant l'inférence de nouvelles données.

Le Web Sémantique favorise donc la réutilisation et le partage de ces données, ainsi que leur traitement automatique par des agents informatiques. Les données représentées de la sorte ont un sens et permettent, en théorie, qu'elles soient interprétées consensuellement par tous les acteurs (producteurs et consommateurs). En 2012, Google a communiqué sur son "Knowledge Graph"[1], ainsi les graphes RDF sont depuis parfois appelés graphes de connaissances.

Les données contenues dans ces graphes sont liées entre elles, telles que l'a proposé Tim Berners-Lee avec ses quatre principes des données liées ouvertes :

1. Utiliser des URI pour nommer les choses.
2. Utiliser les URI HTTP pour que l'on puisse déréférencer ces noms.
3. Fournir des informations utiles lorsque l'URI est déréférencé.
4. Inclure des liens vers d'autres URI afin de permettre la découverte de nouvelles

[1] `https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html` - Accédé le: 2019-10-31

choses.

Une manière de lier deux entités est d'utiliser un lien *owl:sameAs*, c'est-à-dire de spécifier de manière explicite que ces deux entités représentent en réalité les deux mêmes objets du monde réel, quelles sont identiques. La notion d'identité est une notion étudiée depuis l'antiquité avec, par exemple, le principe d'identité qui stipule que toute chose est identique à elle même. Le sujet a été longuement débattu, puisque l'on peut se poser, par exemple, la question de l'identité si le sujet évolue au cours du temps, ou si certaines de ses parties sont remplacées. Leibniz a proposé une définition fondée sur les mathématiques. Ce sont les principes d'identité des indiscernables (si deux entités ont les mêmes propriétés, alors elles sont identiques) et sa réciproque, le principe d'indiscernabilité des identiques (si deux entités sont identiques alors elles ont les mêmes propriétés). La propriété *owl:sameAs* a été définie en utilisant la proposition de Leibniz concernant l'identité. Ainsi, la sémantique de *owl:sameAs* est très stricte, puisque, pour être identiques, deux entités doivent avoir les mêmes couples de propriété-valeur dans tous les contextes possibles et imaginables. Cette conception de l'identité a été mainte fois attaquée ou débattue et n'est donc pas sans poser de problèmes dans notre cadre du Web sémantique. En effet, de nombreux cas d'utilisations de cette propriété sont discutables, puisque la relation d'identité est extrêmement dépendante du contexte.

owl:sameAs peut aussi être utilisée pour lier les entités identiques au sein d'un même graphe ou pour lier plusieurs graphes de connaissances, favorisant ainsi la réutilisation de ces entités et de leurs descriptions et permettant de découvrir de nouvelles informations. En effet, un couple de propriété-valeur décrivant une entité peut être réutilisé par n'importe quelle autre entité qui lui est identique. Ceci est l'un des points forts de cette propriété, puisque ***owl:sameAs* permet d'augmenter la complétude d'une entité**, c'est-à-dire d'augmenter les connaissances que l'on a à son sujet. La propriété est par définition transitive, il est ainsi possible d'avoir plusieurs entités identiques reliées par des propriétés *owl:sameAs* et formant une chaîne d'entités identiques. Ainsi, sur une chaîne d'entités liées par *owl:sameAs*, tout couple propriété-valeur peut être utilisé sur n'importe quelle entité de cette chaîne. **C'est ce que l'on nomme la propagation de propriétés.** Mais, de ce fait, un lien qui peut être parfois vrai et parfois faux risque de propager des informations

fausses. Cette propagation de données fausses le long d'une chaîne de propriétés *owl:sameAs* diminue la qualité des données et peut avoir des conséquences dramatiques selon l'utilisation qui est faite de ces données. Par exemple, dans le domaine médical, une mauvaise prise de décision fondée sur des données qui ne sont pas justes peut entraîner un préjudice important pour les patients.

Afin de pouvoir identifier les différents contextes dans lesquels un lien d'identité peut être explicité entre deux entités, nous avons observé les différents travaux déjà effectués sur ce sujet et avons réalisé que la propagation des propriétés n'était pas ou peu traitées, malgré les propositions de plusieurs définitions d'identité contextuelles. En première approximation avant d'étudier plus en profondeur sa définition, nous pouvons considérer qu'un contexte d'identité est un ensemble de propriétés. Toutes les entités identiques dans ce contexte ont les mêmes valeurs pour les propriétés de cet ensemble.

Dans cette thèse, nous nous intéressons donc particulièrement à la manière dont certaines propriétés peuvent être propagées ou pas entre des entités identiques dans un contexte donné. En effet, tout comme la propriété *owl:sameAs*, pour être utile, **un lien d'identité contextuelle doit permettre la propagation de certains couples de propriété-valeur au sein des entités identiques dans le contexte donné**. Par exemple, si l'on considère deux médicaments dont le principe actif est le même, mais étant produits par deux entreprises différentes, alors ces médicaments sont identiques dans le contexte médical, mais différents dans le contexte commercial. Dans le contexte médical, on peut s'attendre à ce que les propriétés décrivant les effets indésirables soient les mêmes pour les deux, ou que l'on puisse compléter les effets de l'un avec ceux de l'autre.

Nos **questions de recherches** principales sont donc :

- Comment définir une identité contextuelle entre entités appartenant à un graphe de connaissances RDF ?
- Comment trouver les propriétés qui peuvent être propagées entre les entités identiques dans un contexte donné ?

Dans un premier temps, nous avons exploré plusieurs pistes pour permettre à un agent (humain ou machine) de connaître (semi-)automatiquement quelles propriétés sont

propageables dans un contexte d'identité donné. La première piste consiste à utiliser la sémantique fournie par les ontologies, exprimée en OWL 2 ou RDFS, qui décrivent les propriétés et les classes d'un graphe de connaissance. En effet, en raison du manque de sémantique observé sur quelques graphes, nous avons, dans un premier temps, mené une étude à large échelle sur la présence et la qualité de l'utilisation d'OWL 2 dans les graphes de connaissances RDF. Nous avons, dans un second temps, proposé une ontologie permettant aux fournisseurs de données de préciser quelles sont les parties d'OWL 2 utilisées dans leurs graphes. Ceci a pour but de faciliter l'exploration des graphes et surtout de permettre de choisir l'outil approprié pour une tâche donnée en fonction des fonctionnalités OWL 2 qui sont présentes dans le graphe donné. En effet, certains outils peuvent nécessiter la présence de certaines fonctionnalités OWL 2, telles que des propriétés fonctionnelles, pour pouvoir produire les résultats attendus par l'utilisateur. Dans le cas qui nous intéresse, une instance de cette ontologie peut nous aider à déterminer si l'utilisation de la sémantique peut être envisagée pour aider à déterminer les propriétés propageables.

Nous avons ensuite étudié une piste fondée sur l'importance des propriétés et de leurs valeurs dans un graphe donné. Il nous est apparu que certaines propriétés peuvent avoir plus ou moins d'importances pour déterminer si deux entités sont identiques ou pas. Ainsi, en fonction de l'utilisation des propriétés dans un graphe de connaissances, il est possible de déterminer le poids que peut avoir une propriété ou encore le pouvoir discriminant d'un couple propriété-valeur. Ces éléments aident à décider si deux entités sont les mêmes ou pas. Cette approche utilise aussi la sémantique en premier ressort pour tenter de trouver les liens d'identités. En effet, la présence de propriétés fonctionnelles, par exemple, peut aider à trouver de tels liens entre entités.

Pour finir, nous avons proposé une approche qui permet de trouver semi-automatiquement les propriétés propageables entre entités identiques dans un contexte donné. Nous reprenons la définition d'un contexte d'identité proposée par [Idrissou, Hoekstra, van Harmelen, Khalili, and van den Besselaar 2017] puisqu'elle offre un cadre pour la propagation des propriétés pour un ensemble de propriétés indiscernables donné. Notre approche utilise une technique de plongement de phrases. Chaque propriété ayant une description longue en langage naturel peut être représentée par un vecteur. Il est ainsi possible de calculer un vecteur

représentant un ensemble de propriétés indiscernables d'une part, et, d'autre part, de calculer la distance entre ce vecteur et ceux des propriétés candidates à la propagation. Nous postulons en effet que plus la description d'une propriété est proche des descriptions de l'ensemble d'indiscernabilité et plus cette propriété a de chances d'être propageable. Pour trouver les contextes d'une entité, nous proposons un algorithme qui calcule le treillis qui représente l'ensemble des contextes d'identité de cette entité. De plus, notre approche de propagation utilise notre ontologie sur les statistiques sémantiques et notre approche sur l'importance des propriétés lors de plusieurs étapes de l'algorithme.

Afin de pouvoir mesurer les effets sur la complétude, au niveau du schéma, de la propagation des propriétés dans un contexte d'identité donné, nous avons besoin d'un outil permettant de calculer cette complétude. En effet, un des objectifs de l'établissement de l'identité entre deux entités est de pouvoir réutiliser de l'information, c'est-à-dire d'augmenter la complétude d'une entité à l'aide d'une ou plusieurs autres entités. Cette complétude peut avoir deux formes : *(i)* au niveau du schéma de l'entité, c'est-à-dire du nombre de propriétés différentes qu'elle utilise, et *(ii)* au niveau de ses données, c'est-à-dire du nombre de valeurs que peut avoir une propriété. La mesure de la complétude des données est très difficile puisqu'il est presque impossible d'établir un étalon d'or pour les données dans le domaine du Web sémantique étant donné qu'il est régi par l'hypothèse du monde ouvert [Darari, Nutt, Pirrò, and Razniewski 2013]. Par conséquent, dans le cas présent, nous nous intéressons uniquement à la complétude du schéma. Afin de calculer cette complétude, il est nécessaire d'avoir un schéma de référence correspondant à la nature de l'entité, c'est-à-dire sa classe (l'objet de la propriété *rdf:type*). À l'aide de ce schéma, il est possible de mesurer la complétude d'une entité avant et après liage avec d'autres entités (contextuellement) identiques. Ainsi, nous présentons une approche et un démonstrateur (LOD-CM) permettant de trouver le schéma conceptuel d'une classe donnée. Bien entendu, le contexte et les propriétés propageables doivent être un sous-ensemble des propriétés utilisées dans le schéma. En effet, si les propriétés propageables n'apparaissent pas dans le schéma conceptuel, il ne sera pas possible de mesurer l'évolution de la complétude sur ces propriétés.

Chapter 1

Introduction

1.1 Context and objectives

The scientific literature has been growing for centuries, and it is growing faster and faster. Faced with this observation, as early as the mid-1940s, Vannevar Bush raised the problems of access to this knowledge and its use by scientists. He proposed a conceptual machine called Memex ([Bush 1945]), which is a kind of memory prosthesis allowing the storage and consultation of documents and creating links between them to help the user to resume her reading where she was. Then, in 1965, Ted Nelson ([Nelson 1965]) complemented this idea by creating hyperlinks between documents, files, or media. Nelson's idea is one of the founding elements of the Web yet to come. At the end of the 1980s, Tim Berners-Lee ([Berners-Lee 1989]) proposed to go further by distributing documents on different machines while linking them using hyperlinks. It is the birth of the Web. The latter is based on three fundamental building parts:

1. the HTTP protocol for communication between machines or clients and servers,
2. the HTML language for exposing data to clients,
3. the URLs for identification and addressing.

The W3C is the organization responsible for the various standards related to the Web. Nowadays, a great deal of data is published daily on the Web in different forms. The majority of this data is published in the form of HTML pages and is, therefore, very

unstructured. Formats such as XML, JSON, PDF, or CSV are also commonly used, especially by organizations, allowing data to be structured or semi-structured.

The Semantic Web ([Berners-Lee, Hendler, Lassila, et al. 2001]) is an extension of the Web proposed by Tim Berners-Lee that takes data structuring a step further. This extension is based on a technology stack managed by the W3C. The stack allows the data and its schema to be expressed in the form of a graph called the RDF graph model. It is composed of three fundamental elements, which are *(i)* the resources (IRI), *(ii)* the literals, and *(iii)* the anonymous nodes (existential quantification). Resources are the main element since they enable the identification of any real-world object or concept. One can describe anything and everything in the form of triples whose subject, i.e., the first element, is an entity identified by an IRI. The second element of the triple is a resource that characterizes the thing being described, such as an address, a name, or a latitude. It is called the property of the triple. Finally, the third element of the triple is the object, which represents the value of the property. This object is either a resource, allowing two resources to be linked, or a literal, i.e., a lexical form having a type such as an integer or a date expressed using an IRI. Moreover, through the RDFS and OWL 2 languages, it is possible to describe the schema of this data, i.e., the properties used in the triples, as well as the classes (types) that the resources can have. Some classes and properties may have semantics allowing the inference of new data.

The Semantic Web promotes the reuse and sharing of this data, as well as its automatic processing by computer agents. The data represented in this way make sense and allow, in theory, for consensual interpretation by all actors (producers and consumers). Since 2012, and Google's communication on this subject[1], RDF graphs are sometimes called knowledge graphs.

By their nature, the data contained in these graphs are intended to be linked, as proposed by Tim Berners-Lee with his four principles of Linked Open Data:

1. Use URIs as names for things.

2. Use HTTP URIs so that people can look up those names.

[1] `https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html` - Accessed: 2019-10-31

3. When someone looks up a URI, provide useful information.

4. Include links to other URIs so that they can discover more things.

One way to link two entities is to use a *owl:sameAs* link, i.e., to explicitly specify that these two entities represent the same two real-world objects, which are identical. The notion of identity has been studied since antiquity with, for example, the principle of identity, which states that everything is identical to itself. There has been much discussion on the subject. For example, what happens if the subject evolves, or if some of its parts are replaced. Leibniz proposed a definition based on mathematics called the principle of identity of indiscernibles (if two entities have the same properties, then they are identical) and its reciprocal, the principle of indiscernibility of identicals (if two entities are identical, then they have the same properties). The *owl:sameAs* property was defined using Leibniz's proposal for identity. Thus, the semantics of *owl:sameAs* is strict, since, to be identical, two entities must have the same property-value pairs in all possible and imaginable contexts. This conception of identity has been attacked or debated many times and is therefore not without problems in the Semantic Web framework. Indeed, many cases of uses of this property are questionable, since the identity relationship is extremely context-dependent.

owl:sameAs is used to link identical entities within the same graph or to link several knowledge graphs, thus promoting the reuse of these entities and their descriptions and allowing the discovery of new information. Indeed, a property-value pair describing an entity can be reused by any other entity that is identical to it. This reuse is one of the strong points of this property since ***owl:sameAs* makes it possible to increase the completeness of an entity**, i.e., to increase the knowledge that one has about it. The property is, by definition, transitive. It is thus possible to have several identical entities linked by *owl:sameAs* properties and forming a chain of identical entities. Thus, on a chain of entities linked by *owl:sameAs*, any property-value pair can be used on any entity of this chain. **This mechanism is called property propagation.** However, as a result, a link that may sometimes be true and sometimes false risks spreading false information. The propagation of false data along a chain of *owl:sameAs* properties decreases the quality of the data. The quality drop can have dramatic consequences depending on data usage. For

example, in the medical field, poor decision-making based on incorrect data can result in significant harm to patients.

To be able to identify the different contexts in which an identity link can be explained between two entities, we analyzed the different works already done on this subject. We realized that the propagation of properties was not or hardly dealt with, despite the proposals of several contextual identity definitions. As a first approximation before studying its definition in more depth, we can consider that an identity context is a set of properties. All identical entities in this context have the same values for the properties of this set.

In this thesis, we are therefore particularly interested in how some properties may or may not be propagated between identical entities in a given context. Indeed, just like the property *owl:sameAs*, to be useful, **a contextual identity link must allow the propagation of certain property-value pairs within identical entities in the given context**. For example, if one considers two drugs with the same active ingredient, but produced by two different companies, then these drugs will be identical in the medical context, but different in the commercial context. In the medical context, it can be expected that the properties describing the adverse effects will be the same for both, or that the effects of one can be complemented with those of the other.

Our primary **research questions** are as follows:

- How to define a contextual identity between entities belonging to an RDF knowledge graph?
- How to find the properties that can be propagated between identical entities in a given context?

1.2 Contributions

In a first step, we explored several ways to allow an agent (human or machine) to know (semi-)automatically which properties are propagable in a given identity context. The first approach consists in using semantics provided by ontologies, expressed in OWL 2 or RDFS, which describe the properties and classes of a knowledge graph. Indeed, due to the

lack of semantics observed on some graphs, we first conducted a large-scale study on the presence and quality of the use of OWL 2 in RDF knowledge graphs. We then proposed an ontology allowing data providers to specify which parts of OWL 2 are used in their graphs. The ontology is intended to facilitate the exploration of graphs and especially to allow the choice of the appropriate tool for a given task according to the OWL 2 functionalities present in the given graph. Indeed, some tools may require the presence of certain OWL 2 functionalities, such as functional properties, to produce the results expected by the user. In the case at hand, an instance of this ontology can help to determine if the use of semantics can be considered to provide propagable properties.

We then investigated an idea based on the importance of properties and their values in a given graph. It appeared to us that some properties might be more or less significant in determining whether or not two entities are identical. Thus, depending on the use of properties in a knowledge graph, it is possible to determine the weight that a property can have or the discriminating power of a property-value pair. These elements help to decide if two entities are the same or not. This approach also uses semantics as a first step to try to find identity links. Indeed, the presence of functional properties, for example, can help to find such links between entities.

Afterwards, we proposed an approach that allows finding semi-automatically the properties that can be propagated between identical entities in a given context. We use the definition of an identity context proposed by [Idrissou, Hoekstra, van Harmelen, Khalili, and van den Besselaar 2017] since it provides a framework for property propagation for a given set of indiscernible properties. Our approach uses a sentence embedding technique where a vector represents each property having a long description in natural language. It is thus possible to compute a vector representing a set of indiscernible properties, on the one hand, and, on the other hand, to compute the distance between this vector and those of the properties that are candidates for propagation. We postulate that the closer the description of a property is to the descriptions of the indiscernible set, the more likely it is propagable. To find the contexts of an entity, we propose an algorithm that computes the lattice that represents the set of identity contexts of this entity. The proposed approach relies on techniques of our previous work on using semantics as a first technique and the

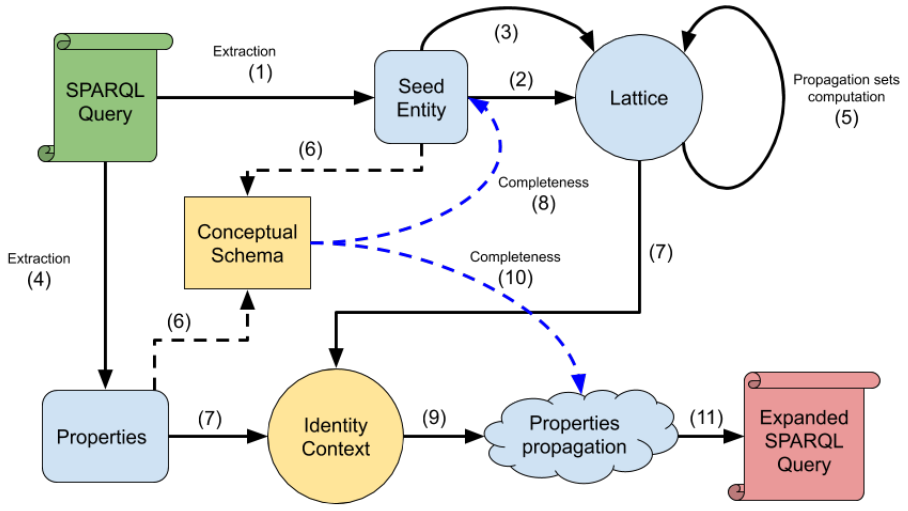

Figure 1.1: Full framework. The element in green is the input of the approach, the element in red is the output. Elements in yellow are those that may require user intervention. The dotted arrows correspond to the part where completeness measurements can be performed. The blue dotted arrows correspond to the moments when completeness measurements are performed in the flow.

importance of properties to weight vector calculations (what we called the weight of a property).

Finally, one of the objectives of establishing the identity between two entities is to be able to reuse information, i.e., to increase the completeness of an entity using one or more other entities. This completeness can take two forms: *(i)* at the schema level of the entity, i.e., the number of different properties it uses, and *(ii)* at the data level, i.e., the number of values a property can have. Measuring data completeness is very difficult since it is almost impossible to establish a gold standard for data in the Semantic Web domain since it is governed by the open-world assumption ([Darari, Nutt, Pirrò, and Razniewski 2013]). Hence, we measure the effects on the completeness, at the schema level, of the propagation of properties in a given identity context. In this case, it is necessary to have a reference schema corresponding to the nature of the entity to compute its completeness, i.e., its class (the object of the property *rdf:type*). With an entity schema, it is possible to measure the completeness of the entity before and after binding with other (contextually) identical entities. Thus, we present an approach and a demonstrator (LOD-CM), allowing us to find the conceptual schema of a given class. Of course, the context and the propagable

properties must be a subset of the properties used in the schema. Indeed, if the propagable properties do not appear in the conceptual schema, it will not be possible to measure the evolution of completeness on these properties.

The final objective of this thesis is to propose and implement the framework illustrated in Figure 1.1 and described below.

1. The user provides a SPARQL query that must be executed on the knowledge graph $\mathcal{KG}$, and it must contain at least one instantiated variable. The following steps are repeated for each instantiated variable in the query. For simplicity, we will take the case where the query has only one instantiated variable called e.

2. The entity e will serve as the seed for the lattice of indiscernible properties, i.e., that each element of the lattice is a set of indiscernible properties Pi_i.

3. Computation of entities identical to the seed e from a logical (or semantic) point of view to enrich the similar entities of each indiscernibility set.

4. The set of $PropQuery_e$ properties related to the seed in the query is extracted. These are the properties that will allow one to select the appropriate context later on.

5. For each Π_i, we calculate its corresponding set of propagable properties.

6. The user chooses, if necessary, a conceptual diagram among those proposed to be able to calculate the completeness of the entity. This schema must include a maximum of $PropQuery_e$ properties.

7. The user chooses, if necessary, an identity context whose propagable properties contain a maximum of $PropQuery_e$ properties.

8. Calculating the completeness C_1 of e.

9. Propagation of properties on e.

10. Calculating the completeness C_2 of e.

11. Expansion of the query.

1.3 Thesis outlines

The thesis is structured as follows:

Chapter 2 introduces the necessary background knowledge and discusses related works. In the first section of this chapter, we explain how knowledge graphs have emerged as a mature way to expose structured data and the vocabulary we use in this thesis. In Section 2.2, we expose the historical and philosophical roots of the problem we tackle in this work. Then, we describe, in Section 2.3, traditional ways to handle identity with knowledge graphs. Finally, before concluding, we relate propositions to handle identity in a contextual way.

Chapter 3 proposes a large-scale study about the presence of semantics in knowledge graphs. The purpose of the study is to get a clear picture of the current semantic landscape since semantics is a cornerstone of our work. Then, we describe an ontology, called OntoSemStats, that captures semantics in a given knowledge graph to facilitate knowledge discovery.

Chapter 4 describes an approach to find identical entities based on semantics and predominance of certain properties. The works described in this chapter are a key element in the main framework we present in Chapter 5.

Chapter 5 describes our main contribution to the propagation of properties for a given indiscernibility set.

Chapter 6 presents our approach to compute conceptual schemas of a class in a knowledge graph. The work is based on a property extraction technique to discover, with a given class, the properties and related classes that can make up a schema. Moreover, this conceptual schema allows computing entity completeness.

Chapter 7 summarizes our contributions and gives some directions for future work.

Chapter 2

Background and State of the art

Firstly, in this chapter, we will clarify the different terms used in this thesis and provide some background information. Secondly, in Section 2.2, we will present the subject of identity from classical points of view, to provide some broader perspectives to our work. In Section 2.3, we present traditional identity management in the Semantic Web community, i.e., instance matching approaches. In Section 2.4, we will present some work that has spotlighted the problems that identity, based on Leibniz's definition, brings to the Semantic Web community. Finally, in Section 2.5, we will present solutions that have been proposed so far.

2.1 From documents to knowledge graphs

In 2012, Google[1] popularized the "knowledge graph" term. While many definitions have been proposed, so far, none of them has been unanimously adopted. [Ehrlinger and Wöß 2016] propose a survey of the most common definitions. Nevertheless, all definitions agreed on several parts: *(i)* it must be a graph *(ii)* that describe entities from the real world or not, and *(iii)* the relations between those entities. While Google has used the "knowledge graph" term since 2012, those graphs have a longer history and are mostly the results of Tim Berners-Lee's Semantic Web.

Indeed, when Tim Berners-Lee invented the Web in 1989-90 (see [Berners-Lee 1989]),

[1] `https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html` - Accessed: 2019-10-31

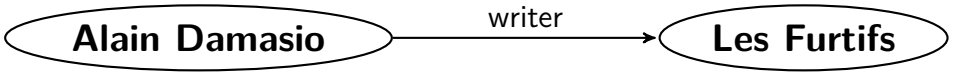

Figure 2.1: Triple for the sentence "Alain Damasio wrote *Les Furtifs*". "Alain Damasio" is the subject, "writer" is the property and "Les Furtifs" is the object (or value).

he kept in mind that the Web could be more than simple documents. He envisioned a Web that could be interpreted by an automated agent because of several key ideas. In [Berners-Lee, Hendler, Lassila, et al. 2001], he proposed the Semantic Web.

The very basis of this construct is the IRI (Internationalized Resource Identifier) that extend URIs (Uniform Resource Identifier) by expanding permitted characters beyond ASCII characters. While this is not mandatory, most of the time IRIs are HTTP IRIs, i.e., they start by "http://" and are dereferenceable. A dereferenceable IRI might be used to perform an HTTP request to obtain a representation of whatever the IRI identifies. As a result, this first atom of the Semantic Web uses the HTTP protocol and is used to identify a resource, i.e., an entity. For example, to identify an entity such as my car, your cat, the Trojan War, or solipsism, one can use an (HTTP) IRI. Despite some small differences, in the remainder of this work, a resource, an entity, or an instance are considered as synonymous terms.

The second piece of the Semantic Web is the RDF data model. It is a specification of the W3C that allows making statements called triples since they are composed of three parts (see Figure 2.1): *(i)* the subject, *(ii)* the predicate, and *(iii)* the object.

The subject must be either an IRI or a blank node (an anonymous resource). A predicate must be an IRI resource. It is a property or a characteristic of the subject. The object (the value of the triple) is either a resource, a blank node, or a literal, i.e., a lexical form having a type such as an integer or a date expressed using an IRI. If the object is an IRI or a blank node, it can be the subject of another statement, which is why several triples form a graph.

RDFS and OWL are two languages allowing the definition of classes (or types or concepts) and properties that are grouped in ontologies. Indeed, this is one of the most important aspects of the Semantic Web, since, unlike traditional databases, one can define

very fine-grained semantics based on description logics (see [Horrocks, Kutz, and Sattler 2006]). Because of semantics, one can infer new knowledge from the original triples.

Finally, SPARQL, for SPARQL Protocol and RDF Query Language, is a query language allowing retrieving and manipulating RDF data. Listing 2.1 is a straightforward example of a SPARQL query to retrieve all theaters from Paris, France. The syntax of SPARQL queries is based on Turtle which is one of the possible serialization of an RDF graph.

```
SELECT DISTINCT ?theater WHERE
{
  ?theater :locatedIn :CotyOfParis .
}
```

Listing 2.1: SPARQL query retrieving all theaters located in Paris, France.

Hence, in the present work, we mean RDF-based knowledge graph when writing using the "knowledge graph", i.e., a labeled oriented multi-graph with resources that are IRIs.

Linked (Open) Data is another important part of the Semantic Web. Indeed, Linked Data is the realization of the Semantic Web, as depicted in Figure 2.2, where each node represents a knowledge graph. The LOD Cloud[2] service exposes more than a thousand graphs.

Tim Berners-Lee defined four principles to expose how Linked Data should be published:

1. Use Uniform Resource Identifiers (URIs) as names for things.
2. Use HTTP URIs so that people can look up those names on the Web.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL [see in Section 3 below]).
4. Include links to other URIs, so that they can discover more things.

He also proposed a five-tiered deployment program to assess the degree of compliance:

1. Available on the Web (whatever format) under an open license.
2. Published as structured data (e.g., Excel instead of an image scan of a table).

[2] https://lod-cloud.net/

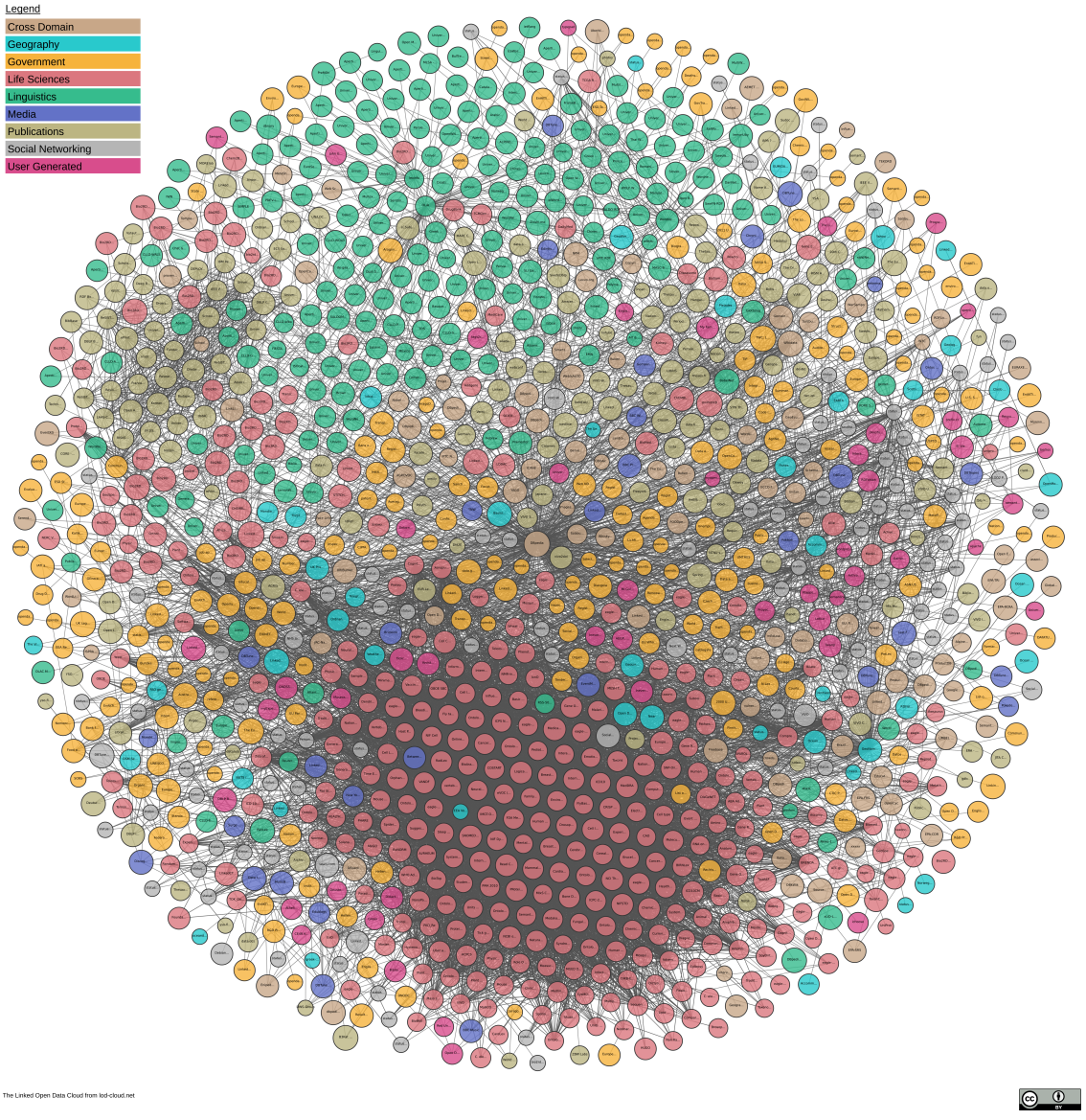

Figure 2.2: The LOD cloud.

3. Use of a non-proprietary open format (e.g., CSV instead of Excel).

4. Use of URIs to denote things, so that people can point at them.

5. Linking of your data to other data to provide context.

In the following sections, we will clarify the context of this work and delve deeper into the problematic notion of identity.

2.2 Identity from historical and philosophical points of view

Philosophers were interested in identity since at least classical antiquity, e.g., with the Theseus paradox. In this thought experiment, one can imagine a ship composed of parts that are replaced over time. Is the ship still the same when all its parts have been changed? There is no absolute answer to this question, but several possible candidates. For example, one possibility is to radically consider that there is no identity over time, which is not ideal in a database context. Another possibility is to consider a gradual loss of identity.
In the same period, the law of identity, one of the three laws of thought, states that everything is identical to itself ($A = A$).

Later, Leibniz proposed the identity of indiscernibles:

$$\forall x, \forall y(\forall p, \forall o, (\langle x, p, o\rangle \ and \ \langle y, p, o\rangle) \rightarrow x = y) \tag{2.1}$$

And its converse, the indiscernibility of identicals:

$$\forall x, \forall y(x = y \rightarrow \forall p, \forall o, (\langle x, p, o\rangle \rightarrow \langle y, p, o\rangle)) \tag{2.2}$$

Whereas there is almost no discussion about the indiscernibility of identicals, the identity of indiscernibles is more controversial, as illustrated in the previous paragraph with several paradoxes that questions its truthfulness.

Nevertheless, the World Wide Web Consortium (W3C[3]), the organization responsible for the Web standards, has based on Leibniz's laws the definition of the property *owl:sameAs*. Nowadays, the identity problem remains one of the most important for industry working with knowledge graphs ([Noy, Gao, Jain, Narayanan, Patterson, and Taylor 2019]).

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

2.3 Traditional instance matching

There are several ways to handle the identity problem. The first one is the traditional way, which goal is to retrieve (create) a maximum of right links. A second way is to find wrong identity links among existing ones. By using either semantics or statistics, it is possible to find links that may be erroneous.

The term *instance matching* refers to the problem of finding equivalent resources. The goal is to produce links between a source dataset and a target dataset. For each couple of entities, a similarity score is produced. If the score is above some (user-defined) threshold, the link is validated. Frameworks like Silk ([Volz, Bizer, Gaedke, and Kobilarov 2009]) or KnoFuss ([Nikolov, Uren, Motta, and Roeck 2008]) allow creating links between datasets after a configuration step. The Ontology Alignment Evaluation Initiative[4] (OAEI) proposes each year a contest to interlink knowledge graphs. We now present some of this work. [Khiat and Mackeprang 2017] proposed I-Match that computes the similarity between normalized strings thanks to NLP. [Achichi, Bellahsene, and Todorov 2017] proposed Legato, a multistage instance matching system that first creates vectors from entities by using NLP techniques and then compute the correlation between vectors, and finally use a clustering algorithm to eliminate some false positives candidates. In [Jiménez-Ruiz and Grau 2011], the authors proposed an instance matching system that is looping between link discovery and repairing, thus allowing reducing the number of wrong candidates. They used an external lexicon (like WordNet) to increase the matching capabilities. [Ferrara, Nikolov, and Scharffe 2011] have published a comprehensive survey and, more recently, [Achichi, Bellahsene, and Todorov 2016] and [Nentwig, Hartung, Ngomo, and Rahm 2017] have proposed further surveys.

Identity link assessment approaches consist of checking if an *existing* link is true or false. Methods from those approaches may also be used to find links. [Guéret, Groth, Stadler, and Lehmann 2012] proposed to use standard network measures to assess existing links. [de Melo 2013] proposed to use the unique name assumption within datasets (i.e., an instance has one name within a dataset) to spot sets of entities linked by *owl:sameAs* where

[4] `http://oaei.ontologymatching.org/`

at least one link may be wrong. Next, a linear programming algorithm is used to check if the link is wrong. In [Papaleo, Pernelle, Saïs, and Dumont 2014], the authors proposed a logical approach that tries to detect logical conflicts by using semantics features like functional properties in small sub-graphs containing the two involved entities to assess. Hence, this approach strongly relies on semantics. [Paulheim 2014] proposed to use data mining methods. Links are represented in an embedded space. Then an outlier detection algorithm is used to detect links that may be wrong. [Valdestilhas, Soru, and Ngomo 2017] use a combination of semantics and graph partitioning algorithms to detect erroneous transitive properties. [Raad, Beek, van Harmelen, Pernelle, and Saïs 2018] proposed to use community detection on identity links network to detect erroneous links. Also, using network structure, [Idrissou, van Harmelen, and van den Besselaar 2018] proposed to combine several network metrics to find wrong links across multiple datasets.

2.4 Identity crisis

As described in [Horrocks, Kutz, and Sattler 2006], *owl:sameAs* purpose is to link two entities that are strictly the same, i.e., both entities are identical in every possible context. *owl:sameAs* has strict semantics allowing us to infer new information. Many existing tools produce such *owl:sameAs* links, as explained in the previous section. However, none of these approaches consider contextual identity links. Their purpose is to discover identity links that allegedly always hold. This is, from a philosophical point of view, hard to obtain, as explained in Section 2.2.

As early as 2002, [Guarino and Welty 2002] raised the issue of identity for ontologies. Especially when time is involved, stating that two things are identical became a philosophical problem. The authors proposed to involve only essential properties, i.e., a property that cannot change in identity. Indeed, as stated in [Halpin, Hayes, McCusker, McGuinness, and Thompson 2010] or [Ding, Shinavier, Finin, and McGuinness 2010], because of the strict semantic of *owl:sameAs*, the burden of data publishers might be too heavy. *owl:sameAs* links are not often adequately used. Some might be simply wrong, and, more insidiously, some might be context-dependent, i.e., the *owl:sameAs* link does not hold in every possible context because it is hard to obtain a consensus on the validity of a statement. The

meaning that a data modeler gives to the data may not correspond to what the end-user expects. The misuse of *owl:sameAs* is often referred to as the "identity crisis" ([Halpin, Hayes, McCusker, McGuinness, and Thompson 2010]).

2.5 Contextual Identity

[Beek, Schlobach, and van Harmelen 2016] addressed this issue by constructing a lattice of identity contexts where contexts are defined as sets of properties. All entities belonging to a context share the same values for each property of this context. Hence, a context is a set of indiscernible properties for an entity. However, the authors do not guide the use of properties outside the context. [Raad, Pernelle, and Saïs 2017] proposed an algorithm named DECIDE to compute contexts, where identity contexts are defined as sub-ontologies. Still, as in the first work, properties of entities that are not in the sub-ontology are ignored. Thus, in the two previous works, there is a limitation regarding properties that do not belong to a context. This limitation cripples the interest in using such approaches. Indeed, one of the goals of an identity context is to define an identity relation between two entities to use information from one to the other. The solution by [Idrissou, Hoekstra, van Harmelen, Khalili, and van den Besselaar 2017] involves such propagation of properties, and thus, increases the completeness of an entity according to a context. However, this proposition requires the user to provide as input the propagable and indiscernible properties. Hence, it leaves the burden to the user to identify and provide context and properties. The user must provide the two sets of indiscernible and propagable properties.

2.6 Conclusion

The formalism proposed by [Idrissou, Hoekstra, van Harmelen, Khalili, and van den Besselaar 2017] seems to be the most appropriate for representing an identity context. Both the indiscernibility and propagation parts can be modeled between several entities with two distinct sets of properties. Nevertheless, the user must provide everything, which might be a complicated and time-consuming task. Hence, our goal is to propose an approach to

compute indiscernibility sets and their corresponding propagation sets automatically. Our approach to handle propagation is composed of several parts.

It appeared to us that some properties might be more or less significant in determining whether or not two entities are identical. Thus, depending on the use of properties in a knowledge graph, it is possible to determine the weight that a property can have or the discriminating power of a property-value pair. These elements help to decide if two entities are contextually the same or not. Moreover, semantics is also used to try to find identity links that are logically grounded. Indeed, the presence of functional properties, for example, can help to find such links between entities.

Hence, we proposed an approach that allows finding semi-automatically the properties that can be propagated between identical entities in a given context. For the definition of an identity context, we use the one proposed by [Idrissou, Hoekstra, van Harmelen, Khalili, and van den Besselaar 2017], since it provides a framework for property propagation for a given set of indiscernible properties. Our approach uses a sentence embedding technique where a vector represents each property having a long description in natural language. This vector is weighted by the importance of the property as previously described. It is thus possible to compute a vector representing a set of indiscernible properties, on the one hand, and, on the other hand, to compute the distance between this vector and those of the properties that are candidates for propagation. We postulate that the closer the description of a property is to the descriptions of the indiscernible set, the more likely it is to be propagable. To find the contexts of an entity, we propose an algorithm that computes the lattice that represents the set of identity contexts of this entity. The proposed approach relies on techniques of our previous work on using semantics as a first technique and the importance of properties to weight vector calculations (what we called the weight of a property).

Chapter 3

Knowledge graphs and OWL 2

This chapter is based on the following publication:

- Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-said Cherfi. État des lieux de l'utilisation de OWL 2 : Analyse et proposition pour capturer les utilisations de la sémantique OWL 2 dans les graphes de connaissances RDF. *Revue des Nouvelles Technologies de l'Information*, Extraction et Gestion des Connaissances , RNTI-E-36: 145–156, 2020c

- Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Ontosemstats: An ontology to express the use of semantics in rdf-based knowledge graphs. In Mária Bieliková, Tommi Mikkonen, and Cesare Pautasso, editors, *Web Engineering - 20th International Conference, ICWE 2020, Helsinki, Finland, June 9-12, 2020, Proceedings*, volume 12128 of *Lecture Notes in Computer Science*, pages 561–565. Springer, 2020a. doi: 10.1007/978-3-030-50578-3_45. URL `https://doi.org/10.1007/978-3-030-50578-3_45` (To be published)

- Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. A study about the use of OWL 2 semantics in RDF-based knowledge graphs. In *The Semantic Web: ESWC 2020 Satellite Events - ESWC 2020 Satellite Events*, 2019b (To be published)

For many users or automated agents, working with knowledge graphs may be a compli-

cated task. Indeed, multiple tools using knowledge graphs rely on semantics to perform at their best. For example, in the context of data integration, some instance matching tools use semantic features such as functional and inverse functional properties or disjoint classes to discover entities that are the same (or not). Hence, in many cases, conducting an exploratory study is required to discover which semantic features are used or defined in a knowledge graph. In this chapter, we first propose a large-scale study of the current state of the Web of data regarding its semantics use. Then, we propose an ontology and large-scale ontology-based Web service that provides statistics about the OWL 2 and RDFS semantic features (e.g., functional properties or subclasses) for a given knowledge graph. The ontology will allow a human or automatic agent to choose the most appropriate tool or data for a given task. It also gives the data publishers a clear picture of semantics they provide to data consumers. These statistics are represented in the form of an RDF graph. The graph makes it easy to use and share.

In Section 3.1, we outline the reasons that led us to conduct this large-scale study on the Web of Data, and motivated us to create an ontology to encapsulate information and usage of semantics in knowledge graphs. In Section 3.2, we give an overview of ontologies that are used to describe knowledge graphs. Section 3.3 is dedicated to present our study on the use of OWL 2 on the Web of Data. In Section 3.4 and 3.5, we provide the details about our ontology and the tools to instantiate it. Finally, in Section 3.6, we conclude both the study and the proposed ontology.

3.1 Introduction

As the number and size of RDF knowledge graphs increase, the difficulty of querying or using this data grows. For a given task, several types of approaches can be considered. Some approaches rely mainly on semantics available in the graphs, others, on the contrary, make little or no use of it. Of course, in between these two extremes, approaches can take advantage of semantics, without relying entirely on it. For example, if the task is to interconnect several knowledge graphs, approaches may use a combination of techniques such as statistics, other external knowledge graphs, semantics, or data partitioning algorithms. Besides, approaches relying mainly on semantics can outperform other types of approaches

if semantics is very present in the knowledge graph. Nevertheless, if semantics is lacking, the results may not be what the user expects. Therefore, it is often necessary to conduct a first exploratory study of the knowledge graph to know which tool will be best suited for a given task. Such a study helps to understand what the data may have to offer. Unfortunately, this exploratory step is time-consuming, especially if the documentation accompanying the knowledge graph is missing or not very informative. Several vocabularies or ontologies have been proposed to provide the user with an overview of the data contained in the knowledge graph. For example, Dublin Core[1] [Weibel, Kunze, Lagoze, and Wolf 1998], Creative Commons Rights Expression Language[2], Data Catalog Vocabulary[3], or VoID[4] [Alexander, Cyganiak, Hausenblas, and Zhao 2009] allow knowledge graphs to be described. However, they do not give the possibility to express which elements of OWL 2 or RDFS are used.

While investigating how to use semantics in knowledge graphs to discover (contextual-)identity links, we were confronted by a recurring problem: knowledge graphs we used for our experiments were lacking semantics. We dug into the literature about semantic studies of knowledge graphs and discovered that all works were relatively old (more than ten years) and were about tiny parts of the Semantic Web.

The **research questions** of this chapter are as follows:

- What is the current state of Linked Open Data knowledge graphs regarding the presence and usage of semantics?
- How to facilitate access to information on the semantic structure of a knowledge graph?

Hence, in this chapter, we propose a large-scale study of the current state of the Web of Data concerning the semantics. Then, this leads us to propose an ontology to express, for a given knowledge graph, which OWL 2 and RDFS features (e.g., functional properties or subclasses) are used and in what proportions. Indeed, this ontology allows the necessary

[1] `http://www.dublincore.org/specifications/dublin-core/`
[2] `https://creativecommons.org/ns`
[3] `https://www.w3.org/TR/vocab-dcat/`
[4] `https://www.w3.org/TR/void/`

information to be brought directly to the data consumer to select, in full knowledge of the facts, the appropriate tool for the realization of his or her task. Besides, we provide applications to instantiate the ontology for a given knowledge graph, to its SPARQL endpoint. The objective is to enable data consumers to know precisely how and to what extent OWL 2 and RDFS are used in the knowledge graph. This will be achieved by aggregating statistics about all vocabularies or ontologies of the knowledge graph described with OWL 2 and RDFS. All the present work is available on GitHub.

3.2 Related work

In this section, we present some works that focus, in one way or another, on the study of the use of semantics in knowledge graphs of Linked Open Data.

In [d'Aquin, Baldassarre, Gridinoc, Angeletou, Sabou, and Motta 2007], the authors analyzed 25500 knowledge graphs in terms of expressivity. Although compelling, this study is old and deals with a tiny number of knowledge graphs.

[Jain, Hitzler, Yeh, Verma, and Sheth 2010] denounces the lack of expressiveness of knowledge graphs, i.e., that many knowledge graphs do not use all the different features of OWL 2, far from it. As a result, many approaches based on the most advanced features of OWL 2 are unusable as they stand on these graphs using non-expressive ontologies. For example, without properties that could be declared as transitive but are not so described, it is more complicated to navigate between related data, which is supposed to be one of the strengths of this type of knowledge graph. Moreover, this paper analyzes only 70 graphs of the *LOD Cloud* [5], which is little, would have strayed to its current size (1239 graphs).

In [Hitzler and van Harmelen 2010], the authors emphasize that some data publishers focus solely on publishing data (i.e., triples) without annotating them with shared ontologies. The lack of semantics reduces the possibility of resonating with this data. They conclude that, apart from the *owl:sameAs* property, the features of OWL 2 are little used. However, this study is more of an empirical finding than a systematic study.

[Hogan, Harth, Passant, Decker, and Polleres 2010] and [Polleres, Hogan, Harth,

[5] `https://lod-cloud.net/`

and Decker 2010] state that the quality of open Linked Data can be problematic due to a lack of property and class definitions, i.e., when no OWL 2 or RDFS description is available. Whereas, for example, defining disjoint classes can help detect inconsistencies in a knowledge graph. [Hogan, Harth, Passant, Decker, and Polleres 2010] covers 12.5 million triples and aims to raise the various issues facing the Semantic Web. For example, the authors note the number of dereferencing problems, syntax problems at the RDF level or inconsistency problems. However, the small sample size and the age of the study this study does not provide answers to our questions. Moreover, the study lacks relevant metrics on the use of semantics.

[Glimm, Hogan, Krötzsch, and Polleres 2012] proposed the biggest and deepest evaluation of OWL 2 usage so far. They evaluated more than 2 billion triples and found a wide disparity in usage between the features of OWL 2. Our study covers more recent and more numerous data (more than 30 billion triples).

[Färber, Bartscherer, Menne, and Rettinger 2018] proposes to investigate the quality of some of the best-known knowledge graphs. The authors provide basic statistics on DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. Although not a large-scale study of the use of semantics, some statistics are interesting (like the number of triples, number of classes, number of relations), but do not sufficiently address semantics expressed by ontologies based on OWL 2.

None of the cited works proposes a complete study on the use of OWL 2 semantics in RDF knowledge graphs with precise figures and at such a scale. In this chapter, we propose to collect information on a large scale about the usage of OWL 2 features. We also provide an ontology describing these features.

3.3 Current state of linked open data

In this section, we present the sources we used to produce results on a larger scale. Next, we will describe the methodology we used to carry out this study and present the different results obtained. **The objective of this study is to have an overview, on a large scale and the most recent data possible, of the use of semantics expressed in**

OWL 2 in RDF knowledge graphs.

Before continuing, we propose the following definition, which will be useful for both the ontology and the study we have conducted.

Definition 3.3.1 *(**Semantic feature**) A semantic feature is any element of OWL 2, i.e., all its properties and classes such as owl:Restriction or owl:SymmetricProperty.*

For example, $\langle$*:age a owl:FunctionalProperty*$\rangle$ and all the triples using *:age* as a predicate. We will consider both the triples defining *:age* and the use of this property as a predicate in triples. Or, $\langle$*:someClasses a owl:AllDisjointClasses*$\rangle$ and all triples of form $\langle$*?x rdf:type :someClasses*$\rangle$ are also considered triples using a semantic feature of OWL 2.

The following definition will be useful to observe our results from a different perspective in Sect. 3.3.4.

Definition 3.3.2 *(**Topic of a knowledge graph**)*
The topic of a knowledge graph is the main subject of the data it contains. Our topic selection is based on LOD Cloud (see Sect. 3.3.1). If a graph has no topic, then it has the default topic: "unknown". Some graphs may have more than one topic, in which case each topic will be used in our study.

3.3.1 Sources

In this section, we will present the different data sources we have considered using for our study on the use of semantics expressed in the form of OWL 2 ontologies. We will explain why we have retained some of them and discarded others. Our goal is to gather information on the use of OWL 2 semantics in RDF knowledge graphs.

The *LOD Cloud* provides a visual overview of the extent and growth of RDF knowledge graphs in recent years. Although presenting only a limited number of graphs (a little more than 1000), the metadata associated with these graphs, especially the topics (see Def. 3.3.2) in which each graph is focused (e.g., linguistics or social networks), make this tool a valuable source for our experience. Indeed, this allowed us to compare graphs relating to very different topics and thus to observe whether differences in modeling and thus in

the use of OWL 2 semantics appear between these topics.

LOD Laundromat[6] ([Beek, Rietveld, Bazoobandi, Wielemaker, and Schlobach 2014]) provides access to more than 650 thousand knowledge graphs in HDT format ([Fernández, Martínez-Prieto, Gutiérrez, Polleres, and Arias 2013] and [Martínez-Prieto, Gallego, and Fernández 2012]). HDT is a compressed format containing RDF triples and allowing search operations to be performed on these triples. The data on these graphs have been cleaned up. Syntax errors, duplicate data, and anonymous nodes, for example, have been removed or addressed. Each graph is contained in a single file with a unique identifier, and metadata is also available (like provenance, number of triples, date of processing). Some of these graphs refer to different versions of the same dataset, e.g., DBpedia-en, DBpedia-fr or DBpedia 3.8. Some of the graphs in *LOD Cloud* are contained in *LOD Laundromat*.

LOD-a-lot[7] is a service providing access to the same data as *LOD Laundromat*. However, data are a single graph in which all the *LOD Laundromat* graphs have been merged into a single knowledge graph. Although very interesting, merging all the graphs into one no longer allows us to distinguish between the different original graphs, and therefore the resulting analysis would have been lessened. That is why we did not select *LOD-a-lot* as the data source for our study.

Therefore, we chose to use *LOD Laundromat* for its large number of knowledge graphs and their serialization in HDT format, as well as the *LOD Cloud* for its metadata on the topics of the different graphs. Wherever possible, we have tried to link the graphs from the two given sources, as explained in the next section.

3.3.2 Information collecting

In this section, we describe how, from the sources seen in the previous section, we collected the information. For a given graph, we analyzed the different classes and properties, on the one hand, i.e., their definitions in OWL 2, and, on the other hand, we analyzed their uses within this knowledge graph. For the first analysis, if ever the definition of a

[6] `http://lodlaundromat.org/`
[7] `http://lod-a-lot.lod.labs.vu.nl/`

Name	Note
# of triples	Knowledge graph size
# of distinct subjects	Graph coverage
# of subjects w/o explicit type	The type of a subject is a very basic piece of information
# of distinct predicates	Necessary to put the following figures into perspective
# of predicates w/o explicit domain	Completeness of property definition
# of predicates w/o explicit range	Completeness of property definition
for each OWL 2 feature f	
# of triples using f	
# of distinct subjects using f	
Topic	The subject of the knowledge graph, if applicable

Table 3.1: Information collected for each knowledge graph.

class (respectively of a property) is not explicitly present in the graph, then we tried to reach this definition by dereferencing, i.e., by going to see if the URL refers to an RDF document containing the searched definition. This step allows us to know, for example, whether a particular property is defined as a functional property and how often it is used. Indeed, sometimes a class or property is defined but is not used in the data at all. Also, for each property, we have searched for all its possible super properties: for example, if p_1 is defined as a sub-property of p_2 and the latter is defined as a functional property, then p_1 will also be functional. This step ensures that we do not forget properties defined with implicit OWL 2 features.

Hence, we downloaded metadata from *LOD Cloud* and *LOD Laundromat* to bring the graphs (the HDT files) closer to their topics when the graph is described in both sources. For each graph, we calculated the numbers presented in Table 3.1.

The software developed for this study was developed in Java and is available for replication on GitHub[8].

[8] `https://github.com/PHParis/sem_web_stats`

Name	Description
ALL	All knowledge graphs.
w/ topic	All graphs with a topic that is not “unknown”.
w/o topic	All graphs with the topic “unknown”.
w/ semantics	All graphs with at least one OWL 2 feature.
w/ sem & topic	All graphs with at least one OWL 2 feature and a topic that is not “unknown”.
TOP 100	The first 100 graphs in terms of the number of triples.
Topic	All graphs with this topic.

Table 3.2: Selector definitions.

3.3.3 Overall Results

Thanks to *LOD Laundromat*, 647 858 knowledge graphs have been analyzed (an HDT file represents a graph). Thus, in this document, an RDF knowledge graph is a serialization of a graph expressed using the RDF graph model, i.e., composed of subject-predicate-value triples. It contains data (A-Box) or ontology (T-Box). We will first present the overall results, then according to the topics of the graphs to know if differences appear in the use of OWL 2 semantics according to the topics. Next, we will take a detailed look at the results for the different features of OWL 2.

The first view of these results is presented in Table 3.3 and 3.4. The first column corresponds to the selector, i.e., the filter we applied to select a subset of the nearly 650 thousand graphs. This selector filters all graphs or graphs with at least one OWL 2 feature, or the 100 largest in terms of the number of triples (see Table 3.2).

Table 3.3 shows a summary of the graphs according to different selectors. The second column indicates the number of graphs selected by the selector (the maximum being, of course, 647,858 graphs). In the next three columns, we indicate the first, second, and third quartiles according to the number of distinct subjects to visualize the distribution of the graphs. Finally, the last column shows the percentage of graphs containing at least one class or property using one of the features of OWL 2. We observe in Table 3.3 that if we consider all graphs and graphs without topics, then the results are very close. This similarity is to be expected since the vast majority of graphs do not have a topic (either the graph was not found in *LOD Cloud* or the topic was not filled in). They are generally very

Selector	# of graphs	Q1	Q2	Q3	% of graphs w/ at least one OWL 2 feature
All	647,858	547	3146.5	38,580	1.53
w/ topic	706	8	29.5	425.25	37.54
w/o topic	647,152	562	3301	39,126.5	1.49
w/ semantics	10,600	184	3952	93,891	100
w/ sem & topic	271	8.25	62.5	1496.25	100
TOP 100 (# triples)	100	5,611,982	8,951,766	12,635,325	34

Table 3.3: Basic statistics by selector in terms of number of subjects (first quartile, median and third quartile) and percentage of datasets using OWL 2

Selector	% of subjects w/o types	% of predicates w/o domain	% of predicates w/o range
ALL	41.08	99.68	99.67
w/ topic	6.49	98.84	96.29
w/o topic	41.49	99.68	99.67
w/ semantics	22.36	98.84	98.79
w/ sem & topic	4.45	98	93.6
TOP 100	12.45	99.69	99.69

Table 3.4: Percentages of subjects without types, and predicates without domains and/or ranges

poor in semantics since, surprisingly, only about 1.5% of them contain at least one OWL 2 feature. Conversely, graphs with a topic use much more semantics than large graphs (those in the top 100). Graphs using OWL 2 semantics vary a lot in terms of size since the smallest 25% use far fewer subjects than in the general case.

Besides, Figure 3.1 shows the number of graphs by OWL 2 features. The horizontal axis is the number of distinct OWL 2 features contained in a graph, e.g., *owl:sameAs*, *owl:FunctionalProperty*. The vertical axis shows the number of graphs on a logarithmic scale. We can see that the vast majority of Linked Open Data knowledge graphs contain very few OWL 2 features. Only 719 knowledge graphs use more than five distinct features of OWL 2, and 9881 use exactly one.

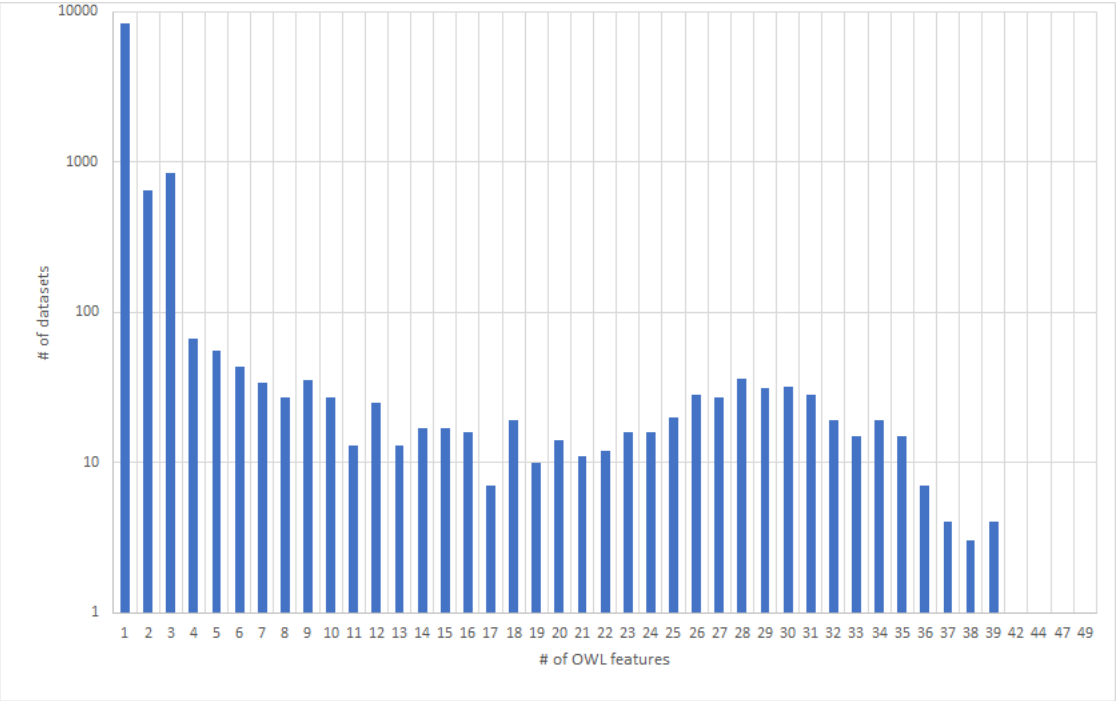

Figure 3.1: Number of knowledge graphs by OWL features 2.

3.3.4 Results by topic

The metadata of *LOD Cloud* allows us to obtain the topic of several knowledge graphs: *government*, *life sciences*, *publications*, *user generated*, *cross domain*, *geography*, *media*, *linguistics*, and *social networking*. We hypothesize that graphs with different subjects are not structured in the same way. For example, we expect user-generated graphs to be less expressive than those on *life sciences*. Table 3.5 shows the results by topic (each line concerns a group of graphs with the topic in the first column). The columns are the same as those in the 3.3 table. We were able to link 706 graphs (out of 1205 in the *LOD Cloud*) to a graph in *LOD Laundromat* (an HDT file). We used strict string matching between the URIs of *LOD Cloud* and *LOD Laundromat* to avoid assigning the wrong topic to a graph.

As expected, graphs whose topic is *user generated* or *social networking* use very little semantics (a little over 12%). Even so, they still use more of them than topicless graphs. More surprisingly, *cross domain* graphs are the biggest users of OWL 2, since 64.62% of them use at least one OWL 2 feature. Additionally, as expected, *life sciences* are among

Selector	# of graphs	Q1	Q2	Q3	% of graphs w/ at least one OWL 2 feature
publications	217	8.5	36	252.5	40.55
government	137	4.25	49.5	473.75	30.66
user_generated	91	8	18	39.5	12.09
life_sciences	68	2	21	39	35.29
cross_domain	65	16.75	48.5	208.5	64.62
geography	48	2	8	443	43.75
linguistics	43	1	20	657	48.84
media	29	2	8	110	51.72
social_networking	8	3	13	37	12.50

Table 3.5: Basic statistics by topic in terms of number of subjects (1st quartile, median and 3rd quartile) and percentage of knowledge graphs using OWL 2. At least one OWL 2 feature must be used at least once.

the largest users of OWL 2. Good news also concerning graphs on *publications*, the latter concerning scientists in the first place, who use much semantics.

3.3.5 Results by feature OWL 2

In this section, we focus on each OWL 2 feature from three different perspectives. We will first look at the property types (e.g., functional). We will then see the different types of classes offered by OWL 2, and finally, we will observe the properties of OWL 2 (like *owl:sameAs*, *owl:inverseOf*). For these perspectives, we will need a definition of the use of OWL 2 classes and properties.

Definition 3.3.3 ***(Class and property usage of OWL 2)*** *Let C be an OWL 2 class (resp. p an OWL 2 property). Let $\Omega = \{KG_i | i \in [1, N]\}$ be a set of knowledge graphs.*

*The **class usage** (resp. the **property usage**) of C (resp. of p), called $CU_\Omega(C)$ (resp. $PU_\Omega(p)$), is the weighted mean of the number of subjects with C as RDF type (resp. using p as a predicate). The weights are the inverse of the total number of subjects in the knowledge graph.*

$$CU_\Omega(C) = \frac{\sum_{i=1}^{N} |Sub(KG_i, C)| \times \frac{1}{|Sub(KG_i)|}}{\sum_{i=1}^{N} \frac{1}{|Sub(KG_i)|}} \tag{3.1}$$

$$PU_\Omega(p) = \frac{\sum_{i=1}^{N} |Sub(KG_i, p)| \times \frac{1}{|Sub(KG_i)|}}{\sum_{i=1}^{N} \frac{1}{|Sub(KG_i)|}} \tag{3.2}$$

Type	# of graphs	weighted mean of triples	weighted mean of subjects	weighted mean of predicates
ObjectProperty	747	27.5	16.57	7.53
DatatypeProperty	685	31	19.43	5.79
FunctionalProperty	434	9	5.76	3.06
InverseFunctionalProperty	310	22.7	20.6	2.54
TransitiveProperty	396	2.84	2.63	2.4
SymmetricProperty	320	7	4.77	2.87
AsymmetricProperty	15	4.7	4.66	4.66
IrreflexiveProperty	21	1.66	1.65	1.65
ReflexiveProperty	16	1.32	1.32	1.32

Table 3.6: Analysis by types of property.

where $Sub(KG_i, C)$ *gives all distinct subjects of type* C *in* KG_i, $Sub(KG_i, p)$ *gives all subjects using* p *in* KG_i *and* $Sub(KG_i)$ *gives all distinct subjects in* KG_i.

Definition 3.3.3 prevents large graphs from weighing too much on average. For example, suppose C = *owl:AllDisjointClasses* is an OWL 2 class. Let us also assume that $\Omega = \{KG_1, KG_2\}$, where KG_1 has 158 distinct subjects with 2 of them having *owl:AllDisjointClasses* as type, and KG_2 has 357 distinct subjects, with 9 of them having *owl:AllDisjointClasses* as type. Then $CU_{\{KG_1,KG_2\}}($ *owl:AllDisjointClasses)* $= \frac{2\times\frac{1}{158}+9\times\frac{1}{357}}{\frac{1}{158}+\frac{1}{357}} = 2.77$.

Table 3.6 concerns the types of properties (for example, a property that would be defined as functional). In our study, a predicate that is a sub-property of a functional property is also functional. The second column shows the number of graphs using a property of the considered type, and the third column their weighted average regarding the number of triples. The last two columns are similar, but for subjects and predicates. For example, inverse functional properties are found in 310 graphs. Among these 310 graphs, we can expect to find an average of 2.54 definitions of such properties that are used in 22.7 triples with 20.6 different subjects. As we can see, some predicates are used very little, such as the *owl:ReflexiveProperty*, which is only used in 16 graphs. In these 16 graphs, very few reflexive properties are defined (1.28) and used.

Table 3.7 concerns the definition of classes using OWL 2. The second column shows

Class	# of graphs	Class usage
Class	1905	1.36
Restriction	520	10.3
DataRange	225	1.71
AllDifferent	213	2.35
NamedIndividual	62	10.8
AllDisjointClasses	50	2.09
NegativePropertyAssertion	27	279.76
Axiom	13	14.44
AllDisjointProperties	5	4.96

Table 3.7: Analysis by class type (see Def. 3.3.3).

the number of graphs using the given class. The third column shows the usage of the class (see Def. 3.3.3). For example, class *owl:Restriction* is used in only 520 graphs. These 520 graphs use it an average of 10.3 times. As expected, the type *owl:Class* is the most used type as opposed to *owl:AllDisjointProperties*, which is practically not used (5 graphs only).

Table 3.8 shows the use of OWL 2 properties in different graphs. The second column shows the number of graphs in which the property exists. The last column shows the usage of this property (see Def. 3.3.3). For example, if a graph uses OWL 2 features, the user can expect to find 5.41 *inverseOf* in that graph. As we can see, the *owl:sameAs* property is by far the most used property of OWL 2, since it is found in 6 times more graphs than the second most used property (*owl:unionOf*). Besides, *owl:sameAs*, when used, is used intensively. On average, 10.3 triples use it. These results correspond to the discoveries of earlier work, such as [Hitzler and van Harmelen 2010]. They showed at the same time the importance of *owl:sameAs* and the inertia regarding the change that can be encountered on this type of graph.

In conclusion, a vast number of knowledge graphs in LOD use little or no semantics in the form of OWL 2, and a few use a lot. Also, many OWL 2 features are only rarely used. However, the larger the graph, the more likely it is to use OWL 2.

Property	# of graphs	Property usage
sameAs	7708	10.30
unionOf	1256	1.02
inverseOf	548	5.41
onProperty	522	10.19
equivalentClass	492	5.11
disjointWith	470	2.49
allValuesFrom	425	7.92
someValuesFrom	413	10.07
cardinality	412	3.1
minCardinality	398	4.26
intersectionOf	394	5.26
maxCardinality	388	4.88
oneOf	364	1.99
hasValue	348	5.58
equivalentProperty	324	4.61
complementOf	315	1.82
distinctMembers	207	2.26
differentFrom	144	30.1
onClass	91	2.99
members	55	2.13
propertyChainAxiom	38	1.76
onDataRange	32	7.2
qualifiedCardinality	32	2.63
assertionProperty	27	279.76
sourceIndividual	27	279.76
targetIndividual	27	279.76
minQualifiedCardinality	25	4.24
disjointUnionOf	24	2.13
withRestrictions	13	1.04
annotatedTarget	12	14.83
maxQualifiedCardinality	12	4.06
annotatedProperty	10	12.71
annotatedSource	10	12.71
onDatatype	10	1.16
hasKey	8	1
propertyDisjointWith	5	1.08
hasSelf	2	1.88
datatypeComplementOf	1	1

Table 3.8: Analysis of OWL 2 properties (see Def. 3.3.3).

3.4 Ontology

The ontology we propose (available online [9] and in Annex A) aims to explain the use of classes and properties defined with OWL 2 and RDFS elements in a knowledge graph. For instance, an objective for a user could be to know the number of properties that are transitive and their number of uses in the graph.

VoID[10] [Alexander, Cyganiak, Hausenblas, and Zhao 2009] is a vocabulary that can be used to describe a knowledge graph. This description facilitates knowledge graph discovery and use. Besides, VoID offers elementary statistics such as the number of classes or triples. Our ontology extends this vocabulary by providing more detailed statistics on the use of OWL 2 and RDFS elements. We represent a knowledge graph as an instance of the class *void:Dataset* that can have as many *:Stat*[11] instances as it uses OWL 2 and RDFS properties or classes. Each instance of *:Stat* has one and only one *:SemanticFeature* instance. The *:hasSemanticFeature* property (see Listing 3.1) allows an instance of *:Stat* to be linked to its *:SemanticFeature*. The different types of range of *:hasSemanticFeature* are disjointed two by two, thus making it possible to detect any error in the instantiation of this ontology.

```
:hasSemanticFeature rdf:type owl:ObjectProperty ,
    owl:FunctionalProperty , owl:AsymmetricProperty ,
    owl:IrreflexiveProperty ;  rdfs:domain :Stat ;
    rdfs:range :SemanticFeature ;
    rdfs:comment "Specify which OWL 2 or RDFS semantic
    feature is the target of the given stat."@en ;
    rdfs:label "has semantic feature"@en .
```

Listing 3.1: Definition of the *hasSemanticFeature* property.

For each feature of OWL 2 and RDFS, we created its own interpretation for two reasons. First, if one has an OWL 1 KG and wants to integrate the stats, then to keep the OWL profile unchanged, we must represent the semantic features with our own IRI. For example, the triple ⟨ *:stat :hasSemanticFeature owl:FunctionalProperty* ⟩ would lead to OWL 1 Full and undecidability problems[12][13] since *owl:FunctionalProperty* is a class. Therefore, for

[9] `http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl`
[10] `https://www.w3.org/TR/void/`
[11] *Classes and properties represented without a prefix belong to our ontology.*
[12] `https://www.w3.org/2007/OWL/wiki/Profile_Explanations`
[13] `https://www.w3.org/TR/owl2-profiles/`

every OWL 2 and RDFS feature, we created a subclass of *:SemanticFeature*. For example, *:FunctionalProperty* represents the statistics of the functional properties. Moreover, the different axioms of OWL 2 and RDFS can impact properties, classes, or entities. For this, we have chosen to ensure that the design of our ontology reflects these possibilities. Depending on its purpose, an axiom will be "put" in a particular class. For example, Listing 3.2 shows the definition of *:PropertyType* (a subclass of *:SemanticFeature*) used to represent the different types that a property can have (e.g., symmetrical, reflexive). Another example is the "PropertyRelation" class, which gathers, among others, statistics concerning *owl:propertyChainAxiom* or *owl:inverseOf*, which are axioms allowing the description of the nature of the relation between properties.

```
:PropertyType rdf:type owl:Class ; rdfs:subClassOf
    :PropertyAxiom ;
    owl:disjointUnionOf ( :OwlAsymmetricProperty
    :OwlFunctionalProperty :OwlInverseFunctionalProperty
    :OwlIrreflexiveProperty :OwlReflexiveProperty
    :OwlSymmetricProperty :OwlTransitiveProperty) .
```

Listing 3.2: Definition of the *Properties* class which represents the different types used to define a property.

To provide statistics for each feature of OWL 2, we have created two properties: *:definitionCount* and *:usageCount*. The first one is to state how many times the axiom is used in a definition (e.g., the number of functional properties) and the second one how many times the definitions using the axiom are used (e.g., how many triples use a functional property). Listing 3.3 shows the definition of the *:usageCount* property, which allows us to state, for example, that 3 000 triples use a functional property.

```
:usageCount rdf:type owl:DatatypeProperty ,
  owl:FunctionalProperty ; rdfs:domain :Stat ;
  rdfs:range xsd:integer ;
  rdfs:comment "Number of usage of a semantic
      feature."@en ; rdfs:label "usage count"@en .
```

Listing 3.3: Definition of the property allowing specifying how many times a feature is used.

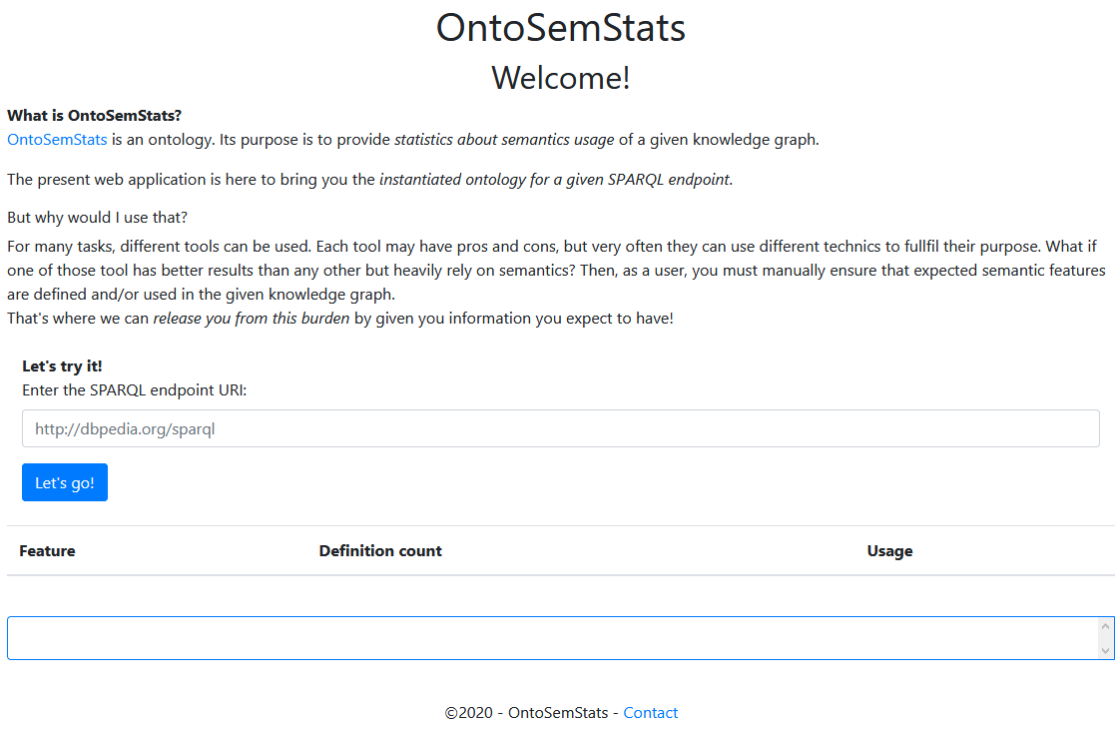

Figure 3.2: Web application interface.

3.5 Web application

Our application, OntoSemStatsWeb[14], is an open-source software (under the GPL open-source license) written in C# (using dotnetRDF[15]) and JavaScript (using Comunica[16] [Taelman, Herwegen, Sande, and Verborgh 2018]). The application has three different forms: *(i)* a Web page that is our live demonstrator[17], *(ii)* a Web API to operate seamlessly with an automated agent, and *(iii)* a command-line application. All the tools that we developed are available as Docker images (one for the command-line application and one for the Web application and the Web API), to promote ease of use and adoption. Figure 3.2 shows the interface of the Web demonstrator. Here, a user can provide a SPARQL endpoint URL, and then the stats are automatically computed. The output is an instantiation of our ontology.

[14] `https://github.com/PHParis/OntoSemStatsWeb`
[15] `https://github.com/dotnetrdf/dotnetrdf`
[16] `https://comunica.linkeddatafragments.org/`
[17] `https://ontosemstats.herokuapp.com/`

Feature	Definition count	Usage
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#OwlDifferentFrom	57623	NA
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#RdfsDomain	5884	NA
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#OwlEquivalentProperty	728	NA
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#OwlEquivalentClass	570058	NA
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#OwlInverseOf	5	NA
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#OwlDisjointWith	173	NA
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#RdfsRange	6053	NA
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#RdfsSubClassOf	1191946	NA
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#OwlInverseFunctionalProperty	3	0
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#OwlSameAs	33624480	NA
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#OwlFunctionalProperty	160	3461431
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#OwlSymmetricProperty	38	0
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#OwlTransitiveProperty	51	0
http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#RdfType	286405045	NA

Figure 3.3: Results as a table for DBpedia endpoint.

Figure 3.3 shows the results for human readings, and Figure 3.4 shows the results for an automated agent. The format chosen is N-Triples to facilitate the processing by the automated agent.

```
_:b9 a <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#Stat> .
_:b9 <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#hasSemanticFeature> <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#
_:b9 <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#definitionCount> 5884 .

_:b11 a <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#Stat> .
_:b11 <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#hasSemanticFeature> <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.ow
_:b11 <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#definitionCount> 728 .

_:b10 a <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#Stat> .
_:b10 <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#hasSemanticFeature> <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.ow
_:b10 <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#definitionCount> 570058 .

_:b12 a <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#Stat> .
_:b12 <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#hasSemanticFeature> <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.ow
:b12 <http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#definitionCount> 5 .
```

Figure 3.4: Results as N-Triples for DBpedia endpoint.

Figure 3.5 shows the workflow of the OntoSemStatsWeb application. The agent (human

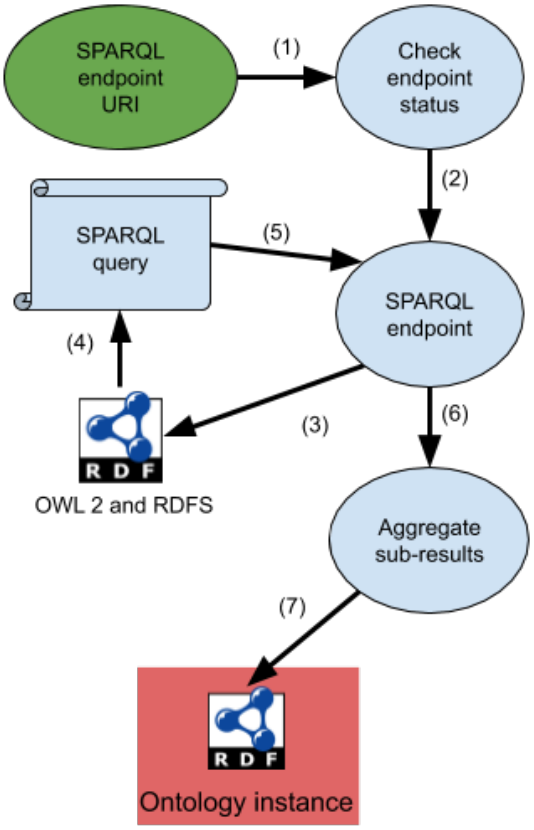

Figure 3.5: OntoSemStatsWeb workflow.

or automated) must provide a SPARQL endpoint URL (1) as input. The application checks the status of the endpoint (2), i.e., if it is up or down. Then, for each OWL 2 or RDFS feature, the application executes the corresponding query on the endpoint (loop (3) to (5)). Next, all sub-results are aggregated in one serialized RDF graph, i.e., the output is the graph that is an instance of the ontology regarding the given endpoint.

Depending on the used tool (i.e., Web page, API, or command-line), the graph is presented in various fashions. The Web page summarizes the results through a user-friendly table and a visual representation and provides a link to download the graph. On the other side, the Web API and the command-line applications allow the graph serialization to be chosen between RDF/XML, Turtle, N-Triples, Notation3, and JSON-LD.

3.6 Conclusion

In this chapter, we conducted a large-scale study that provides an up-to-date overview of the semantic usages in Linked Open Data. We observed a real lack of semantics at several levels. First, many knowledge graphs do not use any OWL 2 features, and many use only a little semantics. On the contrary, some knowledge graphs use complex structures. Second, only some features are heavily used. Indeed, many features are almost not used. These statements are not a judgment, but only an observation about the current state of the Web of Data: maybe most users do not need semantics. Nevertheless, we argue that some users could benefit from having access to more semantic data. We also proposed an ontology that described the OWL 2 and RDFS features defined and used in a given knowledge graph. Moreover, we provided tools that automatically instantiate this ontology for a given SPARQL endpoint. A human agent can use these tools through a Web page and a command-line program or an automated agent through a Web API. By offering easy access to the statistics about semantic usages, we help data consumers in choosing the right tool or knowledge graph that best suited his or her objectives. Moreover, this facilitated access may increase knowledge graph consumption and improve user experience.

As far as our thesis framework is concerned, ontology will help us to decide whether it is worth using semantics or not for a given knowledge graph.

Chapter 4

Semantics and predominance of properties

This chapter is based on the following publication:

- Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Interlinking rdf-based datasets: A structure-based approach. In Imre J. Rudas, János Csirik, Carlos Toro, János Botzheim, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES-2019, Budapest, Hungary, 4-6 September 2019*, volume 159 of *Procedia Computer Science*, pages 162–171. Elsevier, 2019a. doi: 10.1016/j.procs.2019.09.171. URL `https://doi.org/10.1016/j.procs.2019.09.171` **(best student paper)**

While considering (contextual-)identity link discovery, we had the intuition that some properties might be more relevant than others to govern the choice to create such a link. Indeed, with an increasing number of Linked Open Data knowledge graphs, insufficient interlinking quality can lead to a decrease in overall data quality. Therefore, it is necessary to keep the interlinking quality as high as possible. One of the main ways to link knowledge graphs is to use *owl:sameAs* links, i.e., to indicate that two things are the same. However, with its strict semantics, there is a lot of misuse of *owl:sameAs* in the wild. Indeed,

identity is often relative and depends on the context of use. We, therefore, propose an approach that enables considering the characteristics of involved knowledge graphs to interlink them thanks to *owl:sameAs* statements. These characteristics will then be used to weight properties in our propagation framework (see Chapter 5). The experimental results performed on real-world knowledge graphs show that the proposed approach is promising.

The remainder of this chapter is structured as follows: in Sect. 4.1, we present our motivations to take in consideration the predominance of some properties while creating identity links. In Sect. 4.3, we detail our proposition to improve instance matching quality. The Sect. 4.4 is about the implementation and results of our approach. Finally, in Sect. 4.5 we present the future works and our conclusions about our approach.

4.1 Introduction

Linked Data[1] knowledge graphs use ontologies backed by Description logics and *OWL* (see [Horrocks, Kutz, and Sattler 2006]) to define their schema. *owl:sameAs*[2] links are the most commonly used links to link knowledge graphs together and discover new knowledge. The property *owl:sameAs* from the OWL 2 ontology language states that two things (two individuals, entities, or resources) are the same and that all statements about one entity are also true for the other (the indiscernibility of identicals as explained in Chapter 2). In the Semantic Web field, retrieving all interchangeable knowledge between two things is still a significant challenge.

There are many ways to find such linksets (see Definition 4.2.2) of pairwise identical things between knowledge graphs. The most obvious is by using OWL semantics itself, e.g., by using properties such as *owl:hasKey*, *owl:FunctionalProperty*, *owl:InverseFunctionalProperty*, etc. Another way to find such links is to use frameworks based on similarity measures between entities.

However, *owl:sameAs* has a strict semantics[3] (see Table 4.1) and is proved to be often misused in the wild (see [Halpin, Hayes, McCusker, McGuinness, and Thompson 2010]).

[1] http://lod-cloud.net/
[2] owl: http://www.w3.org/2002/07/owl#
[3] https://www.w3.org/TR/owl2-profiles/

A lot of *owl:sameAs* links are inappropriate, whether they are simply wrong, or because they are context-dependent.

The **research questions** of this chapter are as follows:

- How semantics can help to determine the existence or the absence of an identity link between two entities?

- Is the structure of a knowledge graph in terms of property usage is any help to discover identity links?

In this chapter, we propose an approach to detect identity links between entities of two knowledge graphs. This approach takes into account the structure of each knowledge graph, which consists of the use of explicit (ontology axioms) and implicit (statistics about properties) characteristics of properties.

The objective of our approach is to find identity links by considering the involved datasets' structures. In other words, we use explicit characteristics (from the ontology) like functional properties, disjoint class axiom, i.e., all available semantics that can help us in our task. We use *OntoSemStats* (see Chapter 3) to first detect if searched properties are used within the knowledge graphs, to avoid unnecessary work if they are lacking. We also use implicit properties, i.e., how properties are used in datasets. A property is explicitly defined in the ontology by its domain, its range, the fact that it can have a literal value or an object value (i.e., another IRI) and is implicitly defined within a dataset by the way it is used (e.g., how many entities use a given property?). Those explicit and implicit features may be of great help to discover new identity links. Concerning the implicit features, we propose to combine similarities function in conjunction with weights according to the use of each property within datasets. For entities of a given class, we use two types of weights. The first one is representing the importance of a property, and the second one is the discriminating power of this property.

4.2 Background and Notation

In this section, we give the preliminary background knowledge and introduce the notation. For a more exhaustive background, we refer the reader to [Baader and Sattler 2001].

In all the remainder of this document, $\mathcal{KG}$, $\mathcal{KG}_i$ denote knowledge graphs, C denotes a class (or an entity type), and R denotes a property (or predicate). $R(a, b)$ denotes that an entity a has the property R with the value b. $C(a)$ denotes that a is an entity of class C. $sameAs(a, b)$ denotes that the entity a is the *same as* [4] the entity b. We also note $\mathcal{S}(\mathcal{KG})$ (resp. $\mathcal{O}(\mathcal{KG})$ and $\mathcal{R}(\mathcal{KG})$) the set of subjects (resp. objects and properties) belonging to triples from $\mathcal{KG}$. Definition 4.2.1 corresponds to the *input* of our approach:

Definition 4.2.1 (Source and target knowledge graphs) *Let $\mathcal{KG}_s$ and $\mathcal{KG}_t$ be two RDF knowledge graphs such that $\mathcal{KG}_s = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ and $\mathcal{KG}_t = \langle \mathcal{T}_2, \mathcal{A}_2 \rangle$. $\mathcal{T}_1$ and $\mathcal{T}_2$ are T-Boxes, and $\mathcal{A}_1$ and $\mathcal{A}_2$ are A-Boxes (see [Baader and Sattler 2001]) such that $\mathcal{T}_1 \sqsubseteq \mathcal{T}_2$*

$\mathcal{KG}_s$ and $\mathcal{KG}_t$ are called respectively the source *and the* target *knowledge graphs.*

Therefore, for this approach, we require that the T-Box of the source knowledge graph be included in the T-Box of the target knowledge graph. We need for entities in both knowledge graphs to share the same properties and have the same classes. Indeed, in this work, we do not deal with the ontology alignment problem for now. Moreover, if there is a need for alignment, it is possible to use multiple tools that exist and gives good results.

Definition 4.2.2 corresponds to the *output* of our approach:

Definition 4.2.2 (Linkset) *Let $L_{\mathcal{KG}_s,\mathcal{KG}_t}(R) = \{R(a, b) | a \in \mathcal{KG}_s \wedge b \in \mathcal{KG}_t\}$ be a linkset between $\mathcal{KG}_s$ and $\mathcal{KG}_t$ for the property R. $L_{\mathcal{KG}_s,\mathcal{KG}_t}(R)$ is thus the set of triples such that the subject comes from the source knowledge graph, the property is R and object is in the target knowledge graph.*

Example 1 *If $R = owl{:}sameAs$, $\{s{:}London, s{:}Paris, s{:}New\_York\} \subset \mathcal{S}(\mathcal{KG}_s)$ and $\{t{:}London, t{:}Dublin, t{:}Paris\} \subset \mathcal{S}(\mathcal{KG}_t)$ then $L_{\mathcal{KG}_s,\mathcal{KG}_t}(R) = \{\langle s{:}London, t{:}London \rangle,$*

[4] https://www.w3.org/TR/owl-ref/#sameAs-def

$\langle s{:}Paris, t{:}Paris\rangle\}$.

In the next section, we will detail our approach.

4.3 Approach

First, we will see an overview of the main algorithm in Sect. 4.3.1, and then a detailed description of the different phases of our approach in Sect. 4.3.2.

4.3.1 Approach summary

Our approach to determining whether an *owl:sameAs* link can be created between two x_1 and x_2 entities can be summarized as follows:

1. If there is semantics (see OntoSemStats in Chapter 3), find any semantic proof of identity. If there is one, stop there.

2. Otherwise, for each common property R between x_1 and x_2 (such that $R(x_1, o_1)$ and $R(x_2, o_2)$):

   (a) Compute similarity between objects of current common property (e.g., between o_1 and o_2) (see Algorithm 2)

   (b) Compute the *weight of the property* R (see Definition 4.3.3)

   (c) Compute the *discriminating power* of the property-value pair $\langle R, o_1\rangle$ (see Definition 4.3.5)

   (d) Aggregate in *SubAggregation* the similarity, the *weight of the property* R and the *discriminating power* of the property-value pair $\langle R, o_1\rangle$

3. Compute the *weight of clues* between x_1 and x_2 in *clue* (see Definition 4.3.6)

4. Aggregate all *SubAggregation* variables and the *clue* weight

In the following section, we will see more details about each step.

4.3.2 In-depth approach

There are two distinct phases when two entities are compared to know if they are the same and if an *owl:sameAs* link must be created between them. The first one relies on *owl:sameAs* semantics. The second one is the heart of the approach.

4.3.2.1 Direct semantic proof

The first step of our approach is to check if some OWL 2 features of interest are present in the knowledge graphs. Hence, we check if the graphs contain instances of our ontology OntoSemStats (see Chapter 3). Otherwise, we build them with one of our tools (see Section 3.5). If no feature allowing discovering identity links is present, then we skip this part. Otherwise, we look for direct semantic proof of equality (or inequality) between the two inspected resources. Table 4.1 and 4.2 shows the semantics of *owl:sameAs* that have been used in our approach. We search for any pattern corresponding to one of the cases contained in Tables 4.1 or 4.2. If the conditions in the *If* column holds (are true), then the column *Then* can be applied, i.e., we can explicitly add the triples it contained in the knowledge graph. The first table is used to find logically identical entities, and the second one is used to find logically different entities.

Example 2 *(Rule prp-fp from Table 4.1) Let us suppose we are assessing if the entities x_1 and x_2 are the same. If we find that the property P is a functional property* [5] *($FunctionalProperty(P)$) and $P(s, x_1)$ and $P(s, x_2)$ (either directly in $\mathcal{KG}_1$ and $\mathcal{KG}_2$ or by inference), then we know for sure that $sameAs(x_1, x_2)$ holds. In that case the main similarity algorithm stops and return 1.*

If there is a proof that x_1 and x_2 are different, the main similarity algorithm stops and return 0. If no semantic clue has been found, the main similarity algorithm continues to the next step.

Algorithm 1 illustrate our approach.

The $IsSemProof$ function (line 2 in Algorithm 1) returns a boolean. If a semantic

[5] https://www.w3.org/TR/owl-ref/#FunctionalProperty-def

	If	**Then**
eq-sym	$T(?x, owl{:}sameAs, ?y)$	$T(?y, owl{:}sameAs, ?x)$
eq-trans	$T(?x, owl{:}sameAs, ?y)$ $T(?y, owl{:}sameAs, ?z)$	$T(?x, owl{:}sameAs, ?z)$
prp-fp	$T(?p, rdf{:}type,$ $owl{:}FunctionalProperty)$ $T(?x, ?p, ?y_1)$ $T(?x, ?p, ?y_2)$	$T(?y_1, owl{:}sameAs, ?y_2)$
prp-ifp	$T(?p, rdf{:}type,$ $owl{:}InverseFunctionalProperty)$ $T(?x_1, ?p, ?y)$ $T(?x_2, ?p, ?y)$	$T(?x_1, owl{:}sameAs, ?x_2)$
prp-key	$T(?c, owl{:}hasKey, ?u)$ LIST$[?u, ?p_1, \ldots, ?p_n]$ $T(?x, rdf{:}type, ?c)$ $T(?x, ?p_1, ?z_1)$ $\ldots$ $T(?x, ?p_n, ?z_n)$ $T(?y, rdf{:}type, ?c)$ $T(?y, ?p_1, ?z_1)$ $\ldots$ $T(?y, ?p_n, ?z_n)$	$T(?x, owl{:}sameAs, ?y)$
cls-maxc2	$T(?x, owl{:}maxCardinality,$ "1"^^$xsd{:}nonNegativeInteger)$ $T(?x, owl{:}onProperty, ?p)$ $T(?u, rdf{:}type, ?x)$ $T(?u, ?p, ?y_1)$ $T(?u, ?p, ?y_2)$	$T(?y_1, owl{:}sameAs, ?y_2)$
cls-maxqc3	$T(?x, owl{:}maxQualifiedCardinality,$ "1"^^$xsd{:}nonNegativeInteger)$ $T(?x, owl{:}onProperty, ?p)$ $T(?x, owl{:}onClass, ?c)$ $T(?u, rdf{:}type, ?x)$ $T(?u, ?p, ?y_1)$ $T(?y_1, rdf{:}type, ?c)$ $T(?u, ?p, ?y_2)$ $T(?y_2, rdf{:}type, ?c)$	$T(?y_1, owl{:}sameAs, ?y_2)$
cls-maxqc4	$T(?x, owl{:}maxQualifiedCardinality,$ "1"^^$xsd{:}nonNegativeInteger)$ $T(?x, owl{:}onProperty, ?p)$ $T(?x, owl{:}onClass, owl{:}Thing)$ $T(?u, rdf{:}type, ?x)$ $T(?u, ?p, ?y_1)$ $T(?u, ?p, ?y_2)$	$T(?y_1, owl{:}sameAs, ?y_2)$

Table 4.1: Semantics used to find identical entities.

	If	**Then**
prp-asyp	$T(?p, rdf{:}type, owl{:}AsymmetricProperty)$ $T(?x, ?p, ?y)$ $T(?y, ?p, ?z)$	$T(?x, owl{:}differentFrom, ?z)$
prp-pdw	$T(?p_1, owl{:}propertyDisjointWith, ?p_2))$ $T(?x, ?p_1, ?y)$ $T(?z, ?p_2, ?y)$	$T(?x, owl{:}differentFrom, ?z)$
prp-adp	$T(?x, rdf{:}type, owl{:}AllDisjointProperties)$ $T(T(?x, owl:members, ?y))$ $LIST[?y, ?p_1, \ldots, ?p_n]$ $T(?u, ?p_i, ?v)$ $T(?w, ?p_j, ?v)$ for each $1 \leq i < j \leq n$	$T(?u, owl{:}differentFrom, ?w)$
cls-com	$T(?c_1, owl{:}complementOf, ?c_2)$ $T(?x, rdf{:}type, ?c_1)$ $T(?y, rdf{:}type, ?c_2)$	$T(?x, owl{:}differentFrom, ?y)$
cls-maxc1	$T(?x, owl{:}maxCardinality,$ "0"$^{\wedge}xsd{:}nonNegativeInteger)$ $T(?x, owl{:}onProperty, ?p)$ $T(?u, rdf{:}type, ?x)$ $T(?v, ?p, ?y)$	$T(?u, owl{:}differentFrom, ?v)$
cls-maxqc1	$T(?x, owl{:}maxQualifiedCardinality,$ "0"$^{\wedge}xsd{:}nonNegativeInteger)$ $T(?x, owl{:}onProperty, ?p)$ $T(?x, owl{:}onClass, ?c)$ $T(?u, rdf{:}type, ?x$ $T(?v, ?p, ?y)$ $T(?y, rdf{:}type, ?c)$	$T(?u, owl{:}differentFrom, ?v)$
cax-dw	$T(?c_1, owl{:}disjointWith, ?c_2)$ $T(?x, rdf{:}type, ?c_1)$ $T(?y, rdf{:}type, ?c_2)$	$T(?x, owl{:}differentFrom, ?y)$
cax-adc	$T(?x, rdf{:}type, owl{:}AllDisjointClasses)$ $T(?x, owl{:}members, ?y)$ $LIST[?y, ?c_1, \ldots, ?_cn]$ $T(?z, rdf{:}type, ?c_i)$ $T(?w, rdf{:}type, ?c_j)$ for each $1 \leq i < j \leq n$	$T(?z, owl{:}differentFrom, ?w)$

Table 4.2: Semantics used to find distinct entities.

input : $x_1 \in \mathcal{KG}_s$ and $x_2 \in \mathcal{KG}_t$ (x_1 and x_2 are entities), hasSem (boolean)
output : the similarity score

1 /* The value of *hasSem* depends on whether the instance of OntoSemStats has features from Table 4.1. If nothing found, then it is *false*, *true* otherwise. */
2 **if** $hasSem \wedge IsSemProof(x_1, x_2)$ **then return** $SemProofValue(x_1, x_2)$;
3 $scores \leftarrow []$;
4 $C \leftarrow max_C\{depth_{\mathcal{KG}}(C) \mid C(x_1) \in \mathcal{KG}_s \wedge C(x_2) \in \mathcal{KG}_t\}$;
5 /* At worst, $C = owl{:}Thing$ */
6 **foreach** $R \in \mathcal{R}_{x_1} \cap \mathcal{R}_{x_2}$ **do**
7 /* see Algorithm 2 for the sim function */
8 $(maxSim, o) \leftarrow max\{sim(o_1, o_2) \mid (o_1, o_2) \in \{o : R(x_1, o)\} \times \{o : R(x_2, o)\}\}$;
9 // the *max()* function returns a tuple composed of o_1 or o_2 depending on which one has the highest similarity score, and this score
10 $subscore \leftarrow Aggregation_1(maxSim, (1 - W_{\mathcal{KG}_s}(R, C)), (1 - D_{\mathcal{KG}_s}(C, R, o)))$;
11 $scores.Add(subscore)$;
12 **end**
13 /* The more information (triples) there is, the stronger the decision must be. w represents this force. */
14 $clue \leftarrow \frac{|R_{x_1} \cap R_{x_2}|}{|R_{x_1}| + |R_{x_2}| - |R_{x_1} \cap R_{x_2}|}$;
15 **return** $Aggregation_2(clue, SubAggregation(scores))$;

Algorithm 1: Calculate the similarity score between two entities

proof of equality or inequality is found, it return true. In the same line, $SemProofValue$ return either 1 or 0 (one if the clue is in favor of
$owl{:}sameAs$, zero otherwise).

4.3.2.2 The use of properties

In this second step of our approach, there are two main ideas. The first one is that rarely occurring property (among entities of a given class) may be stronger to evaluate identity between two entities than an omnipresent property (see Example 3). The second one is that a property-value couple that occurred less is more helpful to find an identity link (see Example 4). We have been inspired by the fact that when looking for a clue, we seek for a specificity since peculiarities narrow down the possibilities.

Example 3 *If 90% of the* People*'s entities use the property* name *but only 8% of those*

input : $o_1 \in \mathcal{KG}_s$ and $o_2 \in \mathcal{KG}_t$
output : the similarity score between two objects

$score \leftarrow 0$;
if $termType(o_1) \neq termType(o_2)$ **then return** $score$;
// The *termType()* function returns a value between *resource* or *literal*, whether the parameter is an RDF resource or a literal value. Blank nodes are ignored for simplicity
if $termType(o_1) = resource$ **then**
  if *semantic proof that* $o_1 = o_2$ **then**
    $score \leftarrow 1$;
  else
    $score \leftarrow stringSim(fragment(o_1), fragment(o_2))$;
else if $dataType(o_1) = dataType(o_2)$ **then**
  // The *dataType()* function returns the data type of a literal, e.g., string or date.
  $score \leftarrow sim_{dataType(o_1)}(o_1, o_2)$;
else
  $score \leftarrow stringSim(o_1, o_2)$;
return $score$;

Algorithm 2: the sim function

entities use the property ownerOf*, then* ownerOf *might help more to determine (the absence of) an identity relation between two entities.*

Example 4 *Suppose that we have the following triples:* $town(a, t_1)$ *and* $town(b, t_2)$*. The discriminating power of the values is important. Suppose we have 100 entities with the property-value* $\langle town, t_1 \rangle$ *but only 3 entities with the property-value* $\langle town, t_2 \rangle$*, then clues having the property-value* $\langle town, t_2 \rangle$ *are stronger than clues having* $\langle town, t_1 \rangle$*, since* $\langle town, t_2 \rangle$ *allows discriminating more entities. We name this the discriminating power of a property-value pair.*

Hence, for each common property between the two entities x_1 and x_2, a similarity score is computed between objects. For example, if we have $country(x_1, o_1)$ and $country(x_2, o_2)$, we compute the similarity $sim(o_1, o_2)(\in [0, 1])$. This similarity depends on the nature of o_1 and o_2 (IRIs, typed literals, etc.). We will see more details later. Each of those similarities will be weighted based on intuitions explain in Example 3 and 4. More formally, before defining the *weight of a property*, we need two preliminary definitions:

Definition 4.3.1 *Let* $NS_{\mathcal{KG}}(C) = |\{s : \exists(R, o) \in \mathcal{R}(\mathcal{KG}) \times \mathcal{O}(\mathcal{KG}), R(s, o) \in \mathcal{KG} \wedge C(s) \in \mathcal{KG}\}|$ *the number of subjects from* $\mathcal{KG}$ *that are of the class* C*.*

Example 5 *In the following examples, we will use this* $\mathcal{KG} = \{$*Woman(ada), Man(lennon), Woman(kahlo), Man(obama), Man(einstein), age(lennon, 26), age(obama, 47), age(einstein, 26), age(kahlo, 37), nationality(ada, brithish)*$\}$*.*

If $C = Woman$*, then* $NS_{\mathcal{KG}}(Woman) = 2$ *(*kahlo *and* ada*).*

Definition 4.3.2 *Let* $NS_{\mathcal{KG}}(C, R) = |\{s : \exists o \in \mathcal{O}(\mathcal{KG}), R(s, o) \in \mathcal{KG} \wedge C(s) \in \mathcal{KG}\}|$ *be the number of subjects of* C *participating in triples having the property* R *in* $\mathcal{KG}$*.*

Example 6 *With the knowledge graph from Example 5,* $C = Woman$ *and* $R = Age$*, then* $NS_{\mathcal{KG}}(Woman, Age) = 1$ *(only* kahlo *has an age provided).*

Now, we can define the *weight of a property* (as explained in Example 3):

Definition 4.3.3 (weight of a property) *The* weight of the property R *on the class* C *in* $\mathcal{KG}$ *is defined by* $W_{\mathcal{KG}}(R, C) = \frac{NS_{\mathcal{KG}}(C,R)}{NS_{\mathcal{KG}}(C)}$ *and* $W_{\mathcal{KG}}(R, C) \in [0, 1]$*.*

This weight $W_{\mathcal{KG}}(R, C)$ represents the percentage of entities of a class (or class) C having the property R in their description.

Example 7 *With knowledge graph from Example 5, then* $W_{\mathcal{KG}}(Age, Woman) = \frac{NS_{\mathcal{KG}}(Woman,Age)}{NS_{\mathcal{KG}}(Woman)} = \frac{1}{2} = 50\%$*.*

We will next define the second intuition seen in Example 4, which is the *discriminating power* of property-value pair:

Definition 4.3.4 *Let* $NS_{\mathcal{KG}}(C, R, o) = |\{s : \exists s, R(s, o) \in \mathcal{KG} \wedge C(s) \in \mathcal{KG}\}|$ *be the number of subjects of class* C *participating in triples having the property* R *and the object* o *in* $\mathcal{KG}$*.*

Example 8 *With the knowledge graph from Example 5,* $C = Man$*,* $R = Age$ *and* $o = 26$*, then* $NS_{\mathcal{KG}}(Man, Age, 26) = 2$ *(*einstein *and* lennon*).*

Now, we can define the *discriminating power*:

Definition 4.3.5 (discriminating power) *The* discriminating power *of a property-value pair* $\langle R, o\rangle$ *on the class* C *in* $\mathcal{KG}$ *is defined by* $D_{\mathcal{KG}}(C, R, o) = \frac{NS_{\mathcal{KG}}(C,R,o)}{NS_{\mathcal{KG}}(C,R)}$ *and* $D_{\mathcal{KG}}(C, R, o) \in [0, 1]$.

The lower the number $D_{\mathcal{KG}}(C, R, o)$, the more the property-value $\langle R, o\rangle$ makes it possible to differentiate between two entities.

Example 9 *With the knowledge graph from Example 5, then* $D_{\mathcal{KG}}(Man, Age, 26) = \frac{NS_{\mathcal{KG}}(Man,Age,26)}{NS_{\mathcal{KG}}(Man,Age)} = \frac{|\{lennon,einstein\}|}{|\{lennon,einstein,obama\}|} = \frac{2}{3} = 66\%$ *and* $D_{\mathcal{KG}}(Man, Age, 47) = \frac{NS_{\mathcal{KG}}(Man,Age,47)}{NS_{\mathcal{KG}}Man,Age} = \frac{|\{obama\}|}{|\{lennon,einstein,obama\}|} = \frac{1}{3} = 33\%$. *In combination with the property Age and for Man entities, the object "47" selects only one entity (*obama*) with its discriminating power of 33%, against two (*lennon, einstein*) for the object "26" with a discriminating power of 66%.*

To sum up, for each common property between two entities, we compute a similarity score weighted by the *weight of the property* and the *discriminating power* of the property-value pair.

Finally, we need to aggregate those weighted similarity scores. In the aggregation process, we take into account the fact that the more clue there is, the more trustworthy the result will be (see line 14 in Algorithm 1).

Example 10 *Suppose that we have three entities a, b and c where a is from* $\mathcal{KG}_s$ *and b and c are from* $\mathcal{KG}_t$. *On the one hand, if we have four clues between a and b, and on the other hand, eight clues between a and c then we strengthen the second result. We give a bonus to the comparison with the more clues to present.*

Therefore, after aggregating all weighted similarity scores, we use another weight to either penalize or reward the result according to the number of common properties between the compared entities. To do this, we compute the following weight:

Definition 4.3.6 (weight of clues) *Let* x_1 *and* x_2 *be two entities. A clue is the usage of a property by the two entities, whatever the values of the triples are. We define* $clue_{x_1,x_2} \in [0, 1]$

as the weight of clues between x_1 and x_2 with the following formula:

$$clue_{x_1,x_2} = \frac{\mid R_{x_1} \cap R_{x_2} \mid}{\mid R_{x_1} \mid + \mid R_{x_2} \mid - \mid R_{x_1} \cap R_{x_2} \mid} \tag{4.1}$$

Example 11 *If* $R_{x_1} = \{rdfs{:}label, foaf{:}name, dbo{:}birthPlace, dbo{:}birthDate\}$, $R_{x_2} = \{rdfs{:}label, dbo{:}birthDate, dbo{:}deathDate\}$ *and* $R_{x_3} = \{rdfs{:}label, foaf{:}name, dbo{:}birthPlace, dbo{:}deathDate\}$, *then* $R_{x_1} \cap R_{x_2} = \{rdfs{:}label, dbo{:}birthDate\}$, $clue_{x_1,x_2} = \frac{2}{4+3-2} = \frac{2}{5} = 0.4$ *and* $R_{x_1} \cap R_{x_3} = \{rdfs{:}label, foaf{:}name, dbo{:}birthPlace\}$ *and* $clue_{x_1,x_3} = \frac{3}{4+4-3} = \frac{3}{5} = 0.6$. *We can see in the second case that there is more information to support the approach. x_1 and x_3 have more information to be compared than x_1 and x_2.*

Some additional points need explanation. Line 2 from Algorithm 1 corresponds to the stage where we search for direct semantic features. Furthermore, in line 4, we use the $depth_{\mathcal{KG}}(C)$ function that is defined by the following:

Definition 4.3.7 (depth of a class) *Let $depth_{\mathcal{KG}}(C)$ be the distance between the class C and the class T (i.e., owl:Thing in RDF) in $\mathcal{KG}$. $depth_{\mathcal{KG}}(C)$ is called the depth of C.*

By definition, $\forall \mathcal{KG}, depth_{\mathcal{KG}}(T) = 0$. (The *Top* class is equivalent to *owl:Thing* in RDF, see [Baader, Horrocks, and Sattler 2008] for more details.)

Example 12 *If $\mathcal{KG} = dbo$*[6] *then $depth_{dbo}(Agent) = 1$ and $depth_{dbo}(Biologist) = 4$ since Agent is a direct sub class of owl:Thing and $Biologist \sqsubseteq Scientist \sqsubseteq Person \sqsubseteq Agent \sqsubseteq owl{:}Thing$.*

We retrieve the deepest common class between x_1 and x_2, i.e., the most specific. It is this class that will be used to compute the *weight of the properties* and the *discriminating power* of the property-value pairs. The idea is that if, for example, two entities are scientists, we will obtain better results if we use the *Scientist* class than the *Human* class.

Finally, Algorithm 2 shows how our approach handles the similarity between two objects. The *fragment* function gives the last part of an IRI, i.e., after the last / or #, to compare

[6]DBpedia ontology: `http://wiki.dbpedia.org/services-resources/ontology`

IRIs in last resort. We compare data types, and if they match, we use an appropriate similarity measure (line 11), e.g., if they are both dates, then we use a similarity function working with dates. If both are resources, we check for direct semantic proof (as we have seen it before), and if we do not find anything, we trivially compare IRIs. There has been no attempt to use recursivity in this part yet.

Furthermore, in the main algorithm (Algo. 1), there are three different aggregation functions (lines 10 and 15) that are used. We used the mean for all three.

4.4 Experiments

In this section, to evaluate our approach, we will present two experiments performed on DBpedia/Wikidata and the SPIMBENCH SANDBOX track from OAEI 2017 and then discuss their results.

4.4.1 Results

We have developed a prototype in C# that can be run either on Linux or Windows machines. It is also available as a Docker image. We choose the open-source library dotNetRDF[7] to handle SPARQL, OWL, and RDF parts. All experiments are performed on a computer with an I7 processor (3.10 GHz) and 16 Go of RAM. All our code and datasets can be found on this Github repository, so our experiments are reproducible: `https://github.com/PHParis/im_prototype`.

For each experiment, we have calculated the precision, recall, and F-measure with result linkset as follows:

- True positive (tp): number of alignments predicted (by our approach) that are actually true
- False positive (fp): number of alignments predicted and actually wrong
- False Negative (fn): number of true alignment not found among those predicted by our approach

[7] `https://github.com/dotnetrdf/dotnetrdf`

- $Precision = \frac{tp}{tp+fp}$
- $Recall = \frac{tp}{tp+fn}$
- F-measure = $\frac{2 \times Precision \times Recall}{Precision+Recall}$

4.4.1.1 First experiment

For the first experiment, we performed an instance matching task on real-world and well-known knowledge graphs. Thus, we used subsets of DBpedia and Wikidata. More precisely, we used *DBpedia Wikidata*[8] in place of Wikidata since it is expressed with DBpedia ontology. Hence, we respect the following condition from Definition 4.2.1: $\mathcal{T}_1 \sqsubseteq \mathcal{T}_2$ (source TBox must be a subset of target TBox). The version of DBpedia used is "2016-10", and the version of *DBpedia Wikidata* is "03.30.2015".

Now we describe how we built our two test knowledge graphs from real-world datasets. In the source knowledge graph KG_s, we have selected entities of persons from DBpedia having at least 15 homonyms (according to *rdfs:label*) in Wikidata. We arbitrarily chose 15 homonyms to have a sufficient challenge without having to recover too many entities (the scaling of our approach is not our main concern at the moment). We queried all triples mentioning one of these entities to construct the source knowledge graph. In the target knowledge graph KG_t, we retrieved all homonyms (belonging to *DBpedia Wikidata*) of entities from our DBpedia selection. For example, the entity *dbpedia:John_Williams* has the label "John Williams", and there are 88 distinct entities in Wikidata with the label "John Williams". The DBpedia selection contains all triples having *dbpedia:John_Williams* as subject or object, and the *DBpedia Wikidata* contains all triples having one of the 88 entities as subject or object. From both source and target knowledge graphs, we deleted *owl:sameAs* links after having assessed them (none of them were erroneous). There were 36 *owl:sameAs*. Those links were then used as a gold standard. The source knowledge graph contains 277 entities, 3468 triples, and 36 candidate entities belonging to the class to be matched with the target knowledge graph. The target knowledge graph contains 1170 entities, 7667 triples, and 552 candidate entities belonging to the class to be matched with

[8] `http://wikidata.dbpedia.org/`

the source knowledge graph.

After applying our approach (that took 12 seconds), we evaluated the generated linkset. We obtained a precision and a recall of 91.7%. In detail, our approach produced 33 true positives, 3 false positives, and 3 false negatives.

Therefore, our approach works well on real-world data. Next, we go further and compare it with other approaches.

4.4.1.2 Second experiment

The goal of this experiment is two folds. First, we want to compare our approach with state of the art approaches. Secondly, other approaches we selected all use advanced terminology or structural comparison techniques (see Sect. 2.3) to perform their task. Hence, we want to prove that a structure-based approach with simple string matching can perform as well as these approaches that use more advanced techniques. To compare our approach with others (see Sect. 2.3), we performed tests using the SPIMBENCH SANDBOX task from OAEI 2017[9]. This task has a source and target knowledge graphs and a gold standard. A class name is provided, so only the entities of this class might be linked. SPIMBENCH SANDBOX datasets are alterations of an original one through value-based, structure-based, and semantics-aware transformations. The source knowledge graph contains 1432 entities, 10883 triples, and 349 candidate entities belonging to the class to be matched with the target knowledge graph. The target knowledge graph contains 1453 entities, 10868 triples, and 443 candidate entities belonging to the class to be matched with the source knowledge graph.

Table 4.3 shows a comparison of our results with each participant of the OAEI 2017 Instance Matching Track for the SPIMBENCH SANDBOX's task. In detail, our approach produced 298 true positives, 51 false positives, and one false negative.

4.4.2 Discussion

In the first experiment, results are good since we obtain 91.7% for all measures. Right links are well found, indicating that our approach is promising. Precision and recall are the

[9] `http://islab.di.unimi.it/content/im_oaei/2017/`

Participants	Precision	Recall	F-Measure
SPIMBENCH Sandbox			
AML	0.849	**1.000**	0.918
I-Match	0.854	0.997	**0.920**
Legato	**0.980**	0.730	0.840
LogMap	0.938	0.763	0.841
Our approach	0.854	0.996	**0.920**

Table 4.3: Comparison with other approaches

same because, several times, a wrong candidate has been selected instead of the right one. This candidate selection issue is due to similarity scores that are too close between the right candidate and the (wrongly) selected one. It seems that our aggregation functions (see Section 4.3.2.2) are responsible for both true and false negatives.

In the second experiment, for the SPIMBENCH SANDBOX knowledge graph, our approach performed well since recall is 99.6% and precision 85.4%. We reached the same F-Measure as I-Match, which was the best competitor on F-measure. In the same way as in the first experiment, several times, a wrong candidate has been selected instead of the correct one. The simple aggregation of the weights is responsible for most of the false positives. Besides, the use of more advanced similarity calculation techniques could improve candidate selection and thus reduce the number of false positives. For example, external knowledge graph or NLP techniques can improve the comparison of strings.

One of the weaknesses of our approach we observed is the case where a wrong candidate is proposed with more corresponding property-value pairs of lesser importance than the right candidate has corresponding property-value pairs of importance. When entity descriptions are too close, our approach may not detect false positives.

There are several areas for improvement, like the three different agg, as mentioned in the previous section. We use simple arithmetic mean, and we must investigate other ways to aggregate the sub-scores (i.e., scores for each common property from the loop line 6 in Algorithm 1) and the three weights (i.e., discriminating power and weight of a property, and the quantification of information). Also, we focus on the source knowledge graph for computing the discriminating power and weight of a property. It may be interesting

to use both knowledge graphs in the process. Likewise, we use only one common class between the two entities, although to use all common classes may strengthen the score. Scalability should also be addressed because if there are more than 1000 entities to match, our approach takes more than ten minutes to complete. This scalability issue is mainly due to the absence of any code optimization for the moment. We also need to improve the post-processing part concerning the validation of matches found. In fact, for now, we simply select for each source entity the best candidate in the target knowledge graph, but if the target does not contain an identical entity, then we produce a false positive. Finally, properties that are not shared by entities (we are trying to match) are discarded, but they may provide hints too. Unlike some other approaches, we do not use external resources as background knowledge. Furthermore, some approaches perform post-processing to eliminate false positives. We could benefit from these last two points, so our approach combining statistics on structure and semantics can be improved.

4.5 Conclusion

In this chapter, we have proposed a fully automatized approach to perform an instance matching task between two knowledge graphs sharing their T-Boxes. This approach uses semantics at its disposal but also uses statistics about properties and property-value pairs according to the most specific common class between the compared entities.

The results show that our approach is a promising way towards better interlinking. The recall is good, which means that our approach works well to find links.

Regarding our thesis framework, we use the predominance of properties to compute a vector that is a weighted mean representing a set of indiscernible properties (see Chapter 5).

Chapter 5

Propagation of properties

This chapter is based on the following publications:

- Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Propagation contextuelle des propriétés pour les graphes de connaissances : une approche fondée sur les plongements de phrases. In *Ingénierie des Connaissances*, 2020b (To be published)
- Pierre-Henri Paris, Fayçal Hamdi, Nobal B Niraula, and Samira Si-Said Cherfi. Contextual propagation of properties for knowledge graphs: A sentence embedding based approach. In *International Semantic Web Conference*, 2020d (under review)

With the ever-increasing number of RDF-based knowledge graphs, the number of interconnections between these graphs using the *owl:sameAs* property has exploded. Moreover, as several works indicate, the identity as defined by the semantics of *owl:sameAs* could be too rigid, and this property is therefore often misused. Indeed, identity must be seen as context-dependent. These facts lead to poor quality data when using *owl:sameAs* inference capabilities. Therefore, contextual identity could be a possible path to better quality knowledge. Unlike classical identity, with contextual identity, only certain properties can be propagated between contextually identical entities. Continuing this work on contextual identity, we propose an approach, based on sentence embedding, to find semi-automatically a set of properties, for a given identity context, that can be propagated between contextually

identical entities. We conducted qualitative and quantitative experiments to validate our approach. The use cases provided demonstrate that identifying the properties that can be propagated helps users achieve the desired results that meet their needs when querying a knowledge graph, i.e., more complete and accurate answers.

The rest of the chapter is organized as follows. In the following section, we motivate the purpose of this work. Then, in Section 5.3, we present our approach. In Section 5.4, we present the quantitative and qualitative experiments we have conducted. Finally, we conclude and define the next directions for our future work.

5.1 Introduction

Open and RDF-based knowledge graphs, like prominent Wikidata[1] or DBpedia[2], are continuously growing in terms of size and usage. Consequently, the number of entities described in those KGs leads to a problem for both data publishers and data users: **how to know if two entities are the same or not?** To interlink knowledge graphs, the *owl:sameAs* property has been defined by the W3C[3] in 2004 to link entities that are allegedly the same. Indeed, a (real world) object is described among several knowledge graphs, and those descriptions are linked thanks to the *owl:sameAs* property. However, the semantic definition of *owl:sameAs* is very strict. It is based on Leibniz's identity definition, i.e., the identity of indiscernibles (see Section 2.2).

Hence, two entities are considered identical if they share all their $\langle$ *property,value* $\rangle$ couples in all possible and imaginable contexts. In other words, two entities are identical if **all their properties are indiscernible** for each value. Once an identity link is stated between two entities, it is possible to use $\langle property, value \rangle$ couples from one entity to another. However, it is a very strong assertion to state that two objects are the same whatever the context. From a philosophical point of view, there are multiple counter-arguments to the definition of Leibniz's identity. For example, if we consider two glasses from the same set of glasses, they are indiscernible from each other and yet they are two

[1] `https://www.wikidata.org`
[2] `https://wiki.dbpedia.org/`
[3] `https://www.w3.org/TR/owl-ref/`

different physical objects. Is a person the same as she was ten years ago?

It is also a technical problem because of the open-world assumption [Drummond and Shearer 2006], on the one hand, and on the other hand, because of what a data publisher has in mind that could be different from what the user expects when using data. Besides, when data is published, it is "almost" impossible to know the **consensus** behind the decision of creating an *owl:sameAs* link. Several works ([Halpin, Hayes, McCusker, McGuinness, and Thompson 2010] or [Ding, Shinavier, Finin, and McGuinness 2010]) have demonstrated that the use of *owl:sameAs* was inadequate. Indeed, established links might be considered as true only in specific contexts. According to [Noy, Gao, Jain, Narayanan, Patterson, and Taylor 2019], the problem of identity management in knowledge graphs remains one of the top challenges in the industry.

As a first intuition, a contextual identity between two entities might be seen as a subset of properties Π for which these entities share the same values for each $p \in \Pi$.

Example 13 *Two different generic drugs Drug1 and Drug2 can be identical when considering the active ingredient. If a knowledge graph contains the triples* $\langle$ *Drug1 activeIngredient Molecule1* $\rangle$ *and* $\langle Drug2\ activeIngredient\ Molecule1\rangle$*, then* $Drug1 \equiv_{activeIngredient} Drug2$ *when the context is activeIngredient.*

One of the core features of *owl:sameAs* is to be able to **propagate all properties** from an entity to other identical entities. Hence, *owl:sameAs* allows discovering more knowledge and to increase completeness. In the same way, contextual identity must help to discover **more knowledge and to increase completeness**, but only under specific circumstances. So, to be useful, a contextual identity must specify what is happening with properties that are not part of the context. In other words, **an identity context must have propagable properties**.

Example 14 *Following the example 13, stating only* $Drug1 \equiv_{activeIngredient} Drug2$ *has a limited interest, if we do not know what to do with other properties besides activeIngredient. Considering the context activeIngredient, the property targetDisease is propagable, and if the statement* $\langle$ *Drug1 targetDisease Disease1* $\rangle$ *exists then we can state that*

$\langle Drug2\ targetDisease\ Disease1 \rangle$. *But if we consider the property excipient, then it is not propagable.*

Moreover, the ability to propagate a property between entities depends on the context, i.e., the same property might be propagable in a context C_1 and not propagable in a context C_2.

Several works have attempted to propose a solution to the contextual identity. [Beek, Schlobach, and van Harmelen 2016], [Idrissou, Hoekstra, van Harmelen, Khalili, and van den Besselaar 2017] and [Raad, Pernelle, and Saïs 2017] defined three different ways to handle identity under a given context. However, none of those works propose a solution to discover properties that can be propagated given a specific context.

Research questions: With a given identity context between two entities, how to find properties that can be propagated? Is it possible to find propagable properties (semi-)automatically?

In this chapter, based on the context definition of [Idrissou, Hoekstra, van Harmelen, Khalili, and van den Besselaar 2017], we propose an approach to **find propagable properties** to facilitate knowledge discovery for users. Instead of manually listing the propagating properties as in [Idrissou, Hoekstra, van Harmelen, Khalili, and van den Besselaar 2017], we automatically identify the propagating properties for a given context using semantic textual similarity, significantly reducing burden to users. The semantic similarity is based on the sentence embeddings corresponding to the textual descriptions of the properties. Our intuition is inspired by Tobler's first law [Tobler 1970], that is:

> "*Everything is related to everything else, but near things are more related than distant things.*"

Therefore, **we hypothesize that, from a semantic point of view, the closer a property is to the identity context, the more likely it could be a right candidate for propagation**. So, the idea is to **compute a distance between indiscernible properties** and **candidate properties for propagation**. Consequently, numbers, and in our case numerical vectors, are best suited to compute this distance. A numerical

representation of the textual description of each property through its *rdfs:comment*[4] or *schema:description*[5] can provide a basis to get this vector. Indeed, sentence embeddings of properties descriptions give us numerical vectors which distributions in the vector space comply with the semantic similarity of the sentences. We validated our approach through quantitative and qualitative experiments.

In this work, we propose to remove partially this burden from the user, i.e., to **semi-automatically compute the propagation set of properties given an indiscernibility set of properties**. For this, we will use a sentence embedding approach (presented in Section 5.3.3) to compute embeddings of (the description of) properties to discover **propagable properties** with respect to a given identity context (as defined in [Idrissou, Hoekstra, van Harmelen, Khalili, and van den Besselaar 2017]).

5.2 Motivation

Sometimes, real-world entities may be close regarding their properties but not exactly the same. For example, the French capital, Paris, is both a city and a department (an administrative subdivision of the French territory). While considering that the city and the department are the same concerning their geography, they are two distinct entities administratively (or legally) speaking. Now, suppose both Paris are represented in a knowledge graph as distinct entities, and both are linked to (possibly distinct) movie theaters. If one wants to retrieve movie theaters located in the city of Paris, results will not be complete if some of them are linked to the department (see Figure 5.1).

A French citizen might know this ground truth, but how to allow an automated agent to discover this fact? Contextual identity is a possible answer to this question, i.e., a set of properties for which values are the same for both entities. Considering the present example, both Paris (city and department) are geographically the same, and some properties related to geography might be **propagated**. In Figure 5.1, the red properties (*geo* and *label*) are indiscernible (have the same values), and the blue properties (*located in*) are propagating. In the real world, movie theaters located either in the city or the department, according to

[4] `https://www.w3.org/TR/rdf-schema/#ch_comment`
[5] `https://schema.org/description`

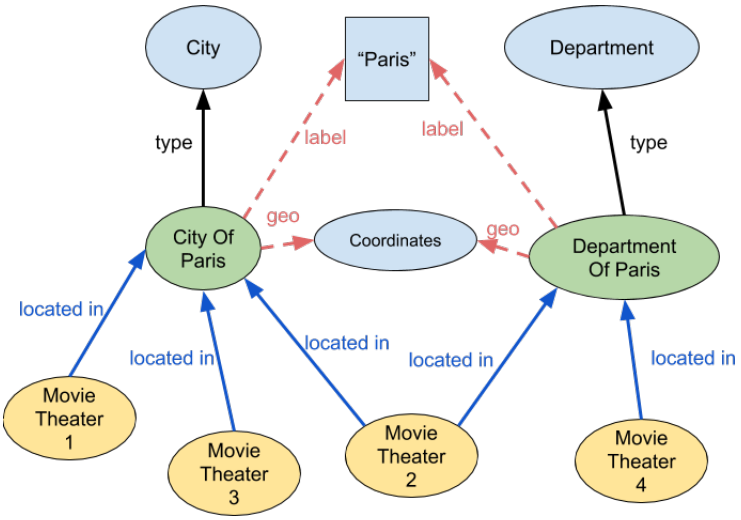

Figure 5.1: Excerpt of a knowledge graph about Paris, France. The properties in red are indiscernible for both the city and the department. The properties in blue are propagating given the red properties are indiscernible.

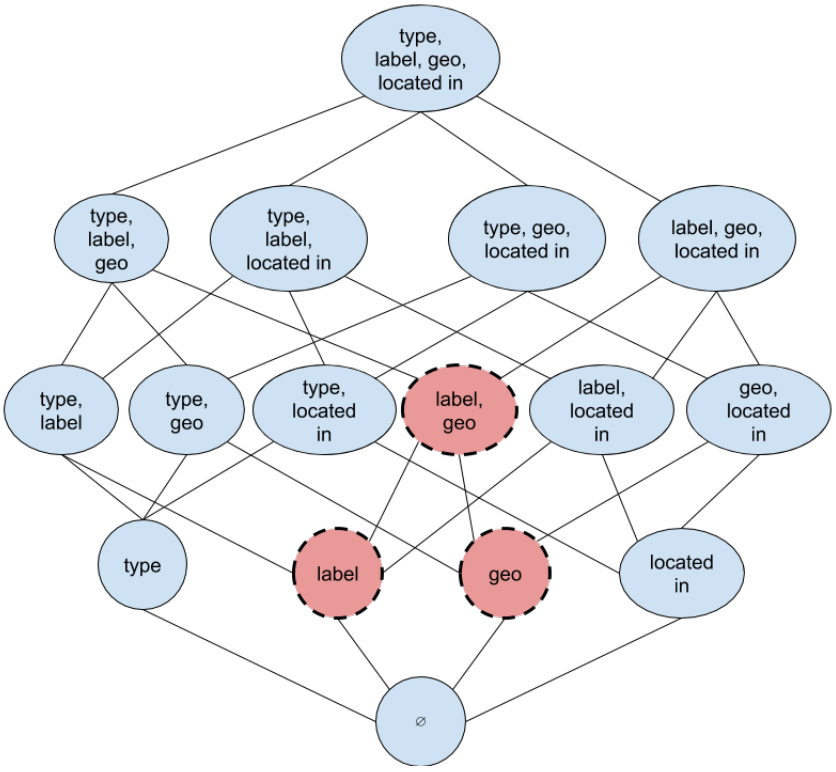

Figure 5.2: Simplified identity lattice from Figure 5.1: each node is an indiscernible set of properties. Only the red nodes have similar entities.

the knowledge graph, are located in the same place. Although the two entities do not share the same values for the *located in* property, this one is related to the geographic context. Indeed, for a human agent, the *located in* property might be obviously propagated between the two entities.

While we expected to have the four movie theaters located in Paris, the query in Listing 5.1 will only return movie theaters 1, 2, and 3 (see Figure 5.1).

```
SELECT DISTINCT ?movieTheater WHERE {
    ?movieTheater :locatedIn :CityOfParis .
}
```

Listing 5.1: SPARQL query retrieving all movie theaters in Paris, France.

Thus, discovering such contexts of identity between entities might improve the completion of query results. Our intuition is inspired by Tobler's first law ([Tobler 1970]), that is:

> "*Everything is related to everything else, but near things are more related than distant things.*"

Therefore, **we hypothesize that, from a semantic point of view, the closer a property is to the identity context, the more likely it could be a right candidate for propagation**. In the previous example, *located in* clearly refers to a geographic fact, and the context of identity is about geography since it is composed of geographical coordinates. So, the idea is to **compute a distance between indiscernible properties** and **candidate properties for propagation**. Consequently, numbers, and in our case, numerical vectors, are best suited to compute this distance. A numerical representation of the textual description of each property through its *rdfs:comment* or *schema:description* can provide a basis to get this vector. Indeed, the embedding of property descriptions gives us numerical vectors whose distributions in vector space respect the semantic similarity of sentences.

5.3 Approach

In this section, before diving deeper into the core approach, we give some definitions needed later to describe the approach.

5.3.1 Preliminaries

We first need to formalize the definition of a propagable property.

Definition 5.3.1 *(**Propagable Property**) The property p can be propagated from an entity e_1 to an entity $e_2 \leftrightarrow (\forall o : \langle e_1, p, o\rangle \rightarrow \langle e_2, p, o\rangle)$.*

As seen in Section 2.5, several propositions have been made to define an identity context. We choose the one from [Idrissou, Hoekstra, van Harmelen, Khalili, and van den Besselaar 2017] since it is the only one that considers the propagation of properties. They give the following definition of the identity context:

Definition 5.3.2 *(**Identity Context**) An identity context $\mathcal{C} = (\Pi, \Psi, \approx)$ is defined by two sets of properties (Π and Ψ) and an **alignment procedure** ($\approx$). Π is the **indiscernibility set** of properties (equation 5.1) and Ψ is the **propagation set** of properties (equation 5.2). In the following, x and y are entities.*

$$\begin{aligned} x =_{(\Pi,\Psi,\approx)} y \leftrightarrow \forall (p_1, p_2) \in \Pi^2 \ \textit{with}\ p_1 \approx p_2 \\ \textit{and}\ \forall v_1, v_2 \ \textit{with}\ v_1 \approx v_2 : \langle x, p_1, v_1\rangle \leftrightarrow \langle y, p_2, v_2\rangle \end{aligned} \tag{5.1}$$

$$\begin{aligned} x =_{(\Pi,\Psi,\approx)} y \rightarrow \forall (p_1, p_2) \in \Psi^2 \ \textit{with}\ p_1 \approx p_2 \\ \textit{and}\ \forall v_1, v_2 \ \textit{with}\ v_1 \approx v_2 : \langle x, p_1, v_1\rangle \leftrightarrow \langle y, p_2, v_2\rangle \end{aligned} \tag{5.2}$$

*Moreover, we define the **level of a context** $|\Pi_{\mathcal{C}}|$ as the number of its indiscernible properties.*

In the case where similar entities according to an identity context belong to the same knowledge graph, it is not necessary to have an alignment procedure.

An entity can have several identity contexts, depending on properties in the indiscernibility set Π. Indeed, two different combinations of properties can give different sets of similar entities. The identity lattice of all identity contexts of an entity e is defined as follows (see Figure 5.2):

Definition 5.3.3 ***(Identity Lattice)*** *An identity lattice $\mathcal{L}$ is a lattice, where each element is an identity context. The set inclusion between indiscernibility set of properties of each context is the binary relation responsible for the partial order.*

The last notion is the seed of a lattice or a context that we define as follows:

Definition 5.3.4 ***(Seed of a lattice or a context)*** *Each context of a lattice is constructed from the* ***same entity*** *e for all the contexts of the lattice. This entity e is called the seed of the lattice.*

Indeed, to build an identity lattice, we need to start from a seed, despite the fact that the lattice could potentially be valid with another seed (see Figure 5.2).

Now that we have defined the necessary concepts, we will explain the core of our approach.

5.3.2 Computation of contexts

In this section, we explain how to compute a lattice and its contexts.

We present the first algorithm (see Algo. 3) that computes an identity lattice [6]. It takes as input the seed entity, the source knowledge graph to which the seed belongs, the target knowledge graph (possibly the same as the source knowledge graph) and an alignment procedure if the two knowledge graphs are distinct. The main idea is to start by computing level one identity contexts with each seed's property and finally combine those contexts to obtain upper-level identity contexts. The first part of a context is its indiscernibility set, from which we then get similar entities, to finally obtain candidate properties for propagation and in the end propagable properties.

[6] The detail of all presented algorithms are available on the GitHub repository.

Data: $\mathcal{KG}_1$: the source KG, $\mathcal{KG}_2$: the target KG, *seed*: an entity of $\mathcal{KG}_1$, $\approx$: an alignment procedure between $\mathcal{KG}_1$ and $\mathcal{KG}_2$
Result: $\mathcal{L}$: a lattice of identity contexts between the seed and entities in the target KG

$\mathcal{L} = \emptyset$;
/* Get all explicit and implicit types of the seed */
$\mathcal{T}_{seed} = \{t : \langle seed\ rdf{:}type\ t\rangle \in \mathcal{KG}_1\}$;
/* Then we get all logically identical entities (see Chapter 3 and 4) */
$\mathcal{I} = getLogicallyIdenticalEntities(seed, \mathcal{KG}_1)$;
/* the following will create all contexts of the level 1 (with only one indiscernible property) */
for *each property p of seed* **do**
    $candidateEntities = \emptyset$;
    for *each value o such as* $\langle seed\ p\ o\rangle \in \mathcal{KG}_1$ **do**
        /* $entities_{p,o}$ is the set of indiscernible entities with *seed* with respect to the p, o couple */
        $entities_{p,o} = \{e : (\exists(p', o'), p' \approx p, o' \approx o, \langle e\ p'\ o'\rangle \in \mathcal{KG}_2) \wedge (\exists t \in \mathcal{T}_{seed}, t' \approx t, \langle e\ rdf{:}type\ t'\rangle \in \mathcal{KG}_2)\}$;
        if $entities_{p,o} \neq \emptyset$ **then**
            $candidateEntities = candidateEntities \cup \{entities_{p,o}\}$;
        end
    end
    /* $entities_p$ is the set of indiscernible entities with *seed* with respect to the property p */
    $entities_p = (\bigcap candidateEntities) \cup \mathcal{I}$;
    $\Psi = getPropagationSet(seed, entities_p, \{p\})$;
    if $\Psi \neq \emptyset$ **then**
        $\Pi = \{p\}$;
        $\mathcal{C} = (\Pi, \Psi, \approx)$;
        $\mathcal{L} = \mathcal{L} \cup \mathcal{C}$;
    end
end
/* Now we can combine contexts of the same level */
return $constructUpperLevels(\mathcal{L}, \mathcal{KG}_1, \mathcal{KG}_2, seed, \mathcal{I}, \approx)$

Algorithm 3: createLattice: calculate identity lattice of an entity.

Data: $\mathcal{L}$: the lattice with only level 1 contexts, $\mathcal{KG}_1$: the source KG, $\mathcal{KG}_2$: the target KG, *seed*: an entity of $\mathcal{KG}_1$, $\mathcal{I}$: the set of logically identical entities, $\approx$: an alignment procedure between $\mathcal{KG}_1$ and $\mathcal{KG}_2$
Result: $\mathcal{L}$: a lattice of identity contexts between the seed and entities in the target KG

```
/* lvl is the current level in the lattice                              */
lvl = 1;
while ∅ ∉ L do
    contexts = ∅;
    for (C1, C2) ∈ {(Ci, Cj) ∈ L × L : |Π_Ci| = |Π_Cj| = lvl, i > j} do
        Π = Π_C1 ∪ Π_C2;
        /* getEntities function gives the set of entities that are
           similar under the given identity context in the given KG  */
        entities = getEntities(C1, KG2) ∩ getEntities(C2, KG2);
        if entities ≠ ∅ and Π ∉ L then
            entities = entities ∪ I;
            Ψ = getPropagationSet(seed, entities, Π);
            /* see Algo. 5                                              */
            if Ψ ≠ ∅ then
                C = (Π, Ψ, ≈);
                contexts = contexts ∪ C;
            end
        end
    end
    L = L ∪ contexts;
    lvl = lvl + 1;
end
return L
```

Algorithm 4: constructUpperLevels: calculate uper levels of the identity lattice of an entity.

We first step is to compute all logically identical entities to the seed (line 5). Indeed, based on Chapters 3 and 4, we search for all entities that can be proven to be identical to the seed by logical inferences, i.e., with the help of semantics. As a matter of fact, we believe that we should consider that identity is context-dependent most of the time, nevertheless we must not discard all classical identity links. We need to distinguish between *owl:sameAs* links produced with statistical approaches and the likes and those produced with logical evidence. The former cannot be trusted, unlike the latter, because semantics is one of the cornerstones of knowledge graphs. In the approach, we delete all *owl:sameAs* links to recompute only those that are logically grounded. Hence, using our ontology OntoSemStats (see Chapter 3), we can determine if the necessary OWL 2 features (see Section 4.3.2.1) are available in the knowledge graph. If so, we infer entities that are the same as the seed. The purpose of $\mathcal{I}$ is to increase the number of candidate properties for propagation, as explained later. The next important step, line 7, is to compute all level 1 identity contexts (see Definition 5.3.2). Indeed, for each property p of the seed, there is exactly one identity context (its indiscernibility set is $\Pi = \{p\}$). Later, identity contexts with only one indiscernibility property will be merged to give identity contexts of higher levels. Next, we retrieve similar entities $entities_p$ to the seed that have the same value(s) for the given property p. If p is multi-valuated, then entities in $entities_p$ are similar to the seed for all values o such that $\langle seed\ p\ o \rangle$. It is worth noting that, when filling $entities_p$, we search only entities that have the same type(s) with the seed. This is because we want to avoid absurd results. It also has the advantage of lowering the number of possible identity contexts to compute. So, in line 17, $entities_p$ is the set of all entities similar to the seed for the property p. Moreover, we add to this set the set $\mathcal{I}$ of logically identical entities to the seed as explained at the beginning of the paragraph. $entities_p$ will next be used to get all properties that could potentially be propagated. Finally, based on $entities_p$, we compute the propagation set Ψ (line 11) as explained in the following section (Section 5.3.3).

The second step (see Algo. 4) is to compute upper-level identity contexts based on those from level 1. The loop (line 3) of the algorithm calculates these upper levels by combining contexts of the same level and stops when it cannot construct new upper-level identity contexts. This calculation is based on an identity lattice operator which is the set

inclusion on indiscernibility sets. For example, a level 2 context is built on two contexts from level 1. Again, to lowering the number of possible identity contexts to compute, if there is no similar entity to the seed for a given context C_i, there is no need to compute higher-level contexts based on C_i.

5.3.3 Sentence embedding

Our approach for computing propagation set (Line 11 in Algo. 4) is elaborated in Algo. 5. It is based on sentence embedding which maps a sentence to a numerical vector. Ideally, semantically close sentences appear nearby in the numerical vector space.

Data: *seed*: the entity that generated Π,
entities: set of entities similar to *seed* with respect to Π,
Π: an indiscernibility set
Result: Ψ: a propagation set
1 `/* computation of the embeddings of each property in` Π `by using one of the encoder */`
2 $indiscernibilityEmbeddings \leftarrow getEmbeddings(\Pi)$;
3 `/* Based on Chapter 4, we compute the weighted mean of the vectors with the weight of each property (see Definition 4.3.3) */`
4 $meanVector \leftarrow mean(indiscernibilityEmbeddings)$;
5 `/*` $getCandidateProperties$ `function returns the set of all candidate properties for propagation */`
6 $candidates \leftarrow getCandidateProperties(\Pi, \{seed\} \cup entities)$;
7 `/* then compute their embeddings */`
8 $candidatesEmbeddings \leftarrow getEmbeddings(candidates)$;
9 $\Psi \leftarrow \emptyset$;
10 **for** $candidateVector$ *in* $candidatesEmbeddings$ **do**
11 $\quad similarity \leftarrow cosineSimilarity(candidateVector, meanVector)$;
12 $\quad$ **if** $similarity \geq threshold$ **then**
13 $\quad\quad \Psi \leftarrow \Psi \cup \{candidateVector\}$;
14 $\quad$ **end**
15 **end**
16 **return** Ψ

Algorithm 5: getPropagationSet: calculate the propagation set.

Sentence embedding is a technique that maps a sentence to a numerical vector. Ideally, semantically close sentences are represented by close vectors in the numerical space considered.

Example 15 *"A soccer game with multiple males playing" and "Some men are playing a sport" are semantically close, thus their vectors should be close in terms of distance.*

Reciprocally, two sentences that are not semantically related should have distant vectors.

Example 16 *"A man inspects the uniform of a figure in some East Asian country" and "The man is sleeping" should have distant vectors.*

In Section 5.3.3, the context of word w is a "window", i.e., words before and after w that can be found in a sentence. Those vectors enable usage of various mathematical operators that are obviously not available with chains of characters. One of the first major work in that field is *Word2Vec* [Mikolov, Chen, Corrado, and Dean 2013] which captures co-occurrence of words. Each word is processed atomically and provides an embedding through two possible and distinct approaches, namely Skip–Gram and Continuous Bag of Words (CBOW). While CBOW aims is to predict a word given its context (i.e., previous and following words in a sentence), Skip–Gram will try to predict words with which a word is usually seen. Similarly, *GloVe* [Pennington, Socher, and Manning 2014] provides embeddings for single words and might use Skip–Gram or CBOW. But *GloVe*, instead of capturing co-occurrence, focuses (in the end) on the count of appearance among contexts (i.e., previous and following words in a sentence). Then, *fastText* [Bojanowski, Grave, Joulin, and Mikolov 2017] is an extension of *Word2Vec* that treats words as n-gram of characters instead of as an atomic entity. N-grams sizes depend on input parameters. N-grams usage allows a better understanding of small words. Each n-gram is mapped to a vector and the sum of these vectors is the representation of the word. Another advantage of *fastText* is its capacity to provide an embedding even for unknown words, thanks to n-grams usage. While the three previous works are best suited to work with atomic words, the following computes embedding for a whole sentence.

The reasons behind using sentence embedding instead of a more classical distance measures, e.g., the edit distance, RDF graph embeddings like RDF2Vec [Ristoski and Paulheim 2016], or an ontological alignment technique are: *(i)* classical string distances ignore sentence semantics, *(ii)* RDF graph embedding techniques are not yet adapted to such a task, and *(iii)* ontological alignment techniques align pairwise properties and not

sets of properties.

Sentence embedding is widely used in several tasks such as computing semantic similarities between two texts. An encoder derives sentence embeddings, to capture the semantics of a language, from a large text corpus. A lot of attention has been given to sentence embeddings lately. Approaches like *Universal Sentence Encoder* [Cer, Yang, Kong, Hua, Limtiaco, John, Constant, Guajardo-Cespedes, Yuan, Tar, Sung, Strope, and Kurzweil 2018], *GenSen* [Subramanian, Trischler, Bengio, and Pal 2018] and *InferSent* [Conneau, Kiela, Schwenk, Barrault, and Bordes 2017] are among the state-of-the-art encoder for sentence embeddings. *InferSent*, proposed by [Conneau, Kiela, Schwenk, Barrault, and Bordes 2017], is a state-of-the-art encoder proved to be effective on sentence embedding. To train their supervised sentence embeddings model, the authors used the Stanford Natural Language Inference (SNLI) dataset that consists of more than 500K pairs of English sentences manually labeled with one of three categories (entailment, contradiction and neutral). They tested several architectures and find out that a BiLSTM network with max pooling offered the best results. A BiLSTM network is a bi-directional LSTM often used for sequence data, i.e., a recurrent neural network (with loops). Max pooling is a technique that allows reducing the number of parameters of the model by selecting the maximum value of a moving "window". Moreover, the pre-trained model is based on fastText, thus allowing computing meaningful representations even for out-of-vocabulary words, i.e., words that did not appear in the training data. *GenSen* [Subramanian, Trischler, Bengio, and Pal 2018] and *Universal Sentence Encoder* [Cer, Yang, Kong, Hua, Limtiaco, John, Constant, Guajardo-Cespedes, Yuan, Tar, Sung, Strope, and Kurzweil 2018] are both based on multi-task learning (MTL). MTL purpose is to learn multiple aspects of a sentence by switching from different tasks like translation or natural language inference. The former uses a bi-directional Gated Recurrent Units (GRU), which is a recurrent neural network like LSTM but with fewer parameters. The latter uses the transformer architecture that transforms a sequence into another but without recurrent neural network (unlike *InferSent* and *GenSen*). In Section 5.4.2.2, we will present results with those three encoders.

As presented in Section 5.1, our intuition, based on Tobler's first law, is that the propagation set of properties can be found given an indiscernibility set, if vectors of

descriptions of those two sets are close enough. As presented in Section 5.3.3, sentence embedding allows us to represent a sentence, i.e., a sequence of words, as a numerical vector. When two sentences are semantically close, their respective vectors should also be close in the considered space. In this work, we propose to use property descriptions (e.g., *rdfs:comment* or *schema:description*) as "*standard plug type for mains electricity in a country*") to find properties that are semantically related and consequently right candidates for propagation for a given indiscernibility set Π. For example, in Wikidata, the property "director" has the follow description: "director(s) of film, TV-series, stageplay, video game or similar". Descriptions are mainly composed of one sentence. Most of the properties are described with such annotations, e.g., properties of Wikidata are annotated with an english *schema:description* at 98.9%. For the embedding computation, any of the previously described encoders can be used. We will provide in Section 5.4.2.2, an analysis of the different results obtained by these encoders.

The last algorithm presents our proposition to compute Ψ given a Π. It takes as input three parameters: a seed (an entity), a set of property built from the seed (indiscernibility set Π), and a set of entities that are similar to the seed with respect to Π. The computation of Π is presented in the previous section (see algorithm 3).

First, for each property in the indiscernibility set Π, we calculate its representational vector (see line 2). Then, we compute the weighted mean vector that represents the indiscernibility set (line 4). With use as weights the weight of properties as defined in Chapter 4 (Definition 4.3.3). Indeed, as explained, some properties are more important to determine identity. In an identity context, the indiscernibility set Π is a set of properties, thus we can compute for each property in Π its vector (see line 2). Then we can compute the mean of the vectors and have a numerical representation of Π (it is also a vector of the same size). This vector representing the mean of vectors derived from Π properties is noted $\sqsubseteq_\Pi$ in the following. Similarly, we consider each property of the seed or its similar entities and compute their representational vectors. Therefore, on the one hand, we have one vector that represents the set of indiscernibility and, on the other hand, we have n vectors for the n properties that are candidates for propagation. Properties of similar entities (with respect to the indiscernibility set Π) are also considered as candidates since possibly one of

them can have a propagating property that the seed does not have (see line 6).

Then we loop on each candidate property to compute a cosine similarity [Singhal 2001] between each candidate vector and the mean vector representing the indiscernibility set Π (line 10). If the cosine similarity is high enough (above a specified threshold as explained in the following section) the candidate property is considered as a propagable property.

Now that our approach has been presented, we will introduce experiments to validate our work.

5.4 Experimental Results

To evaluate our approach, we first implemented our approach and then conducted two types of experiments. In the first experiment, we built a gold standard upon Wikidata. Then, we computed standard precision, recall and F-measure against this gold standard. In the second experiment, we present several SPARQL queries that benefited from our approach.

5.4.1 Implementation and set-up

We implemented our approach in Python. For the sake of reproducibility, the code is made available on a GitHub repository[7]. As mentioned earlier, we used three sentence embedding approaches, namely *InferSent*[8], *GenSen*[9] and *Universal Sentence Encoder*[10]. We used an HDT file (see [Martínez-Prieto, Gallego, and Fernández 2012] and [Fernández, Martínez-Prieto, Gutiérrez, Polleres, and Arias 2013]) that contains a dump of the last version of Wikidata[11]. HDT is a compressed serialization format for RDF graphs that allows a better reproducibility than a live SPARQL endpoint. Unlike Turtle or N-Triples, thanks to compression, HDT facilitates the manipulations needed to reproduce the experiments. The computer we used has an i7 processor and 32 GB of RAM. As an indication, the complete calculation of the identity lattice for an entity such as the city of Paris, France

[7] Anonymous URL
[8] `https://github.com/facebookresearch/InferSent`
[9] `https://github.com/Maluuba/gensen`
[10] `https://tfhub.dev/google/universal-sentence-encoder/2`
[11] `http://gaia.infor.uva.es/hdt/wikidata/wikidata2018_09_11.hdt.gz`

takes about 1396 ms. It has more than 1000 property-object couples and, in Wikidata, the mean number of property-object couples is about 60. Thus, it is a rather large entity and this approach could scale well.

5.4.2 Quantitative Study

The purpose of the quantitative experiment is to allow comparison with future approaches that may arise. To the best of our knowledge, our approach is the only one that has focused on the propagation of properties for contextual identity. Thus, one of the most important contributions of our work is the gold standard we provide. Indeed, it is not obvious, even for a human agent, to determine properties that might be propagated for a given indiscernibility set of properties.

5.4.2.1 Gold standard

As mentioned previously, we want to evaluate if our approach identifies **relevant propagable properties** to the user **according to a given context**. For this, we built a gold standard from the Wikidata knowledge graph that is known for its high data quality. It is also one of the most important actors of Linked Open Data initiative and is linked to many other knowledge graphs, e.g., DBpedia. Obviously, in this case, we no longer need an alignment procedure ($\approx$), since we do not consider multiple knowledge graphs (source and target knowledge graphs are the same).

We built 100 identity contexts, each context containing the indiscernibility set of properties Π and the propagation set of properties Ψ. We choose 5 classes (20 entities by class): country, comics character, political party, literary work and film. Those classes have been chosen for several reasons. Firstly, they are sufficiently different to challenge our approach. Secondly, because the experts must judge which properties are propagable, it is easier for them if they have a minimum of knowledge about the subjects. Finally, it allows us to further investigate if results are different for different classes. For each selected class, we randomly selected one entity. Then we computed its identity lattice.

As stated before, the most difficult part when building the gold standard is to obtain a consensus among experts. A set of properties representing the indiscernibility set Π, and

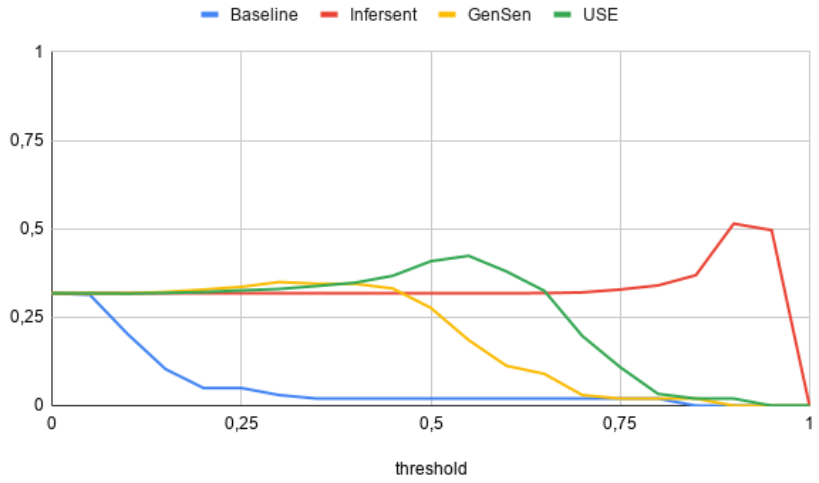

Figure 5.3: Comparison of the average precision by thresholds for all five classes (country, comics character, political party, literary work, film). The threshold takes values from 0.5 to 0.95 by steps of 0.05. The baseline is in blue, InferSent in red, GenSen in yellow and Universal Sentence Encoder in green.

the experts must accordingly choose propagable properties in a set of candidates. It is the most time-consuming part to build the gold standard.

5.4.2.2 Execution against the gold standard

For each entity in the gold standard, we retrieved its partial identity context from the gold standard, i.e., the context with only the indiscernibility set Π. Then, we applied our algorithm on each set of indiscernibility to find the corresponding propagation set Ψ. For each context, we calculate the precision, recall, and F-measure as follows: True positive (tp) is the number of selected properties (by our approach) that are actually in Ψ, false positive (fp) is the number of selected properties (by our approach) and actually not in Ψ, false negative (fn) is the number of properties in Ψ not selected by our approach, $Precision = \frac{tp}{tp+fp}$, $Recall = \frac{tp}{tp+fn}$, and $Fmeasure = \frac{2 \times Precision \times Recall}{Precision+Recall}$ We then aggregated precisions, recalls and F-measures of each context thanks to the standard mean.

As there are no other approaches, to the best of our knowledge, retrieving candidate properties for propagation with respect to an indiscernibility set Π, we compared our

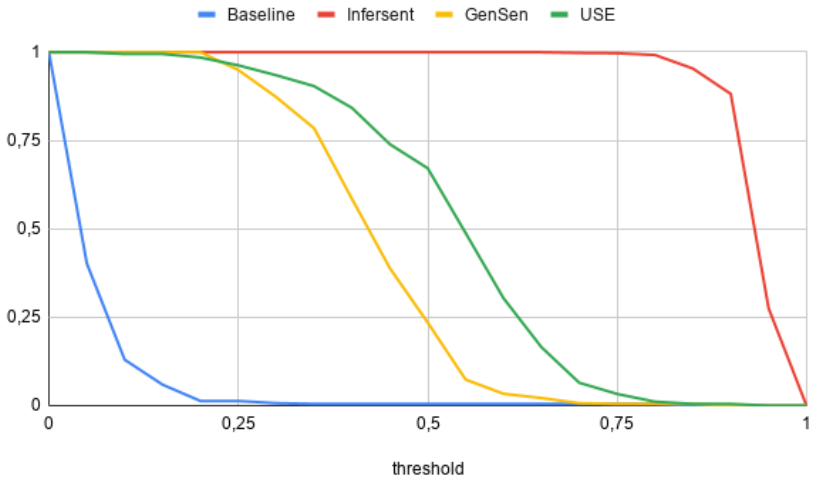

Figure 5.4: Comparison of the average recall by thresholds for all five classes (country, comics character, political party, literary work, film). The threshold takes values from 0.5 to 0.95 by steps of 0.05. The baseline is in blue, InferSent in red, GenSen in yellow and Universal Sentence Encoder in green.

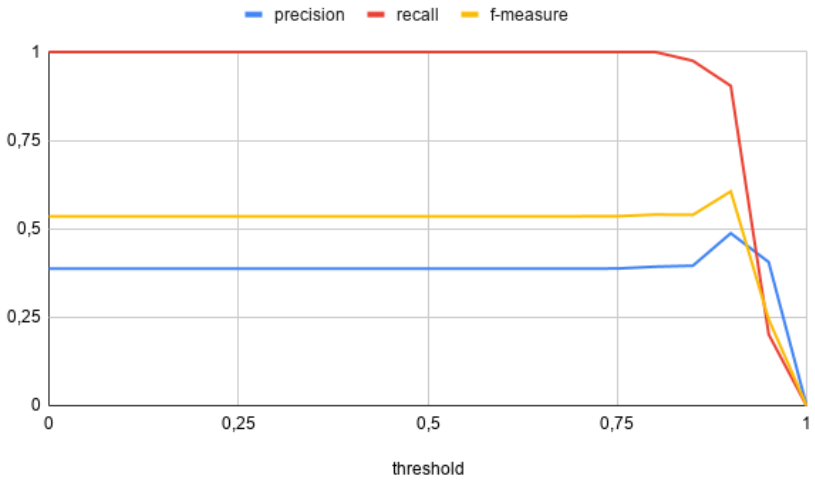

Figure 5.5: Precision (in blue), recall (in red) and F-measure (in yellow) with InferSent and the threshold at 0.9 for the "comics character" class.

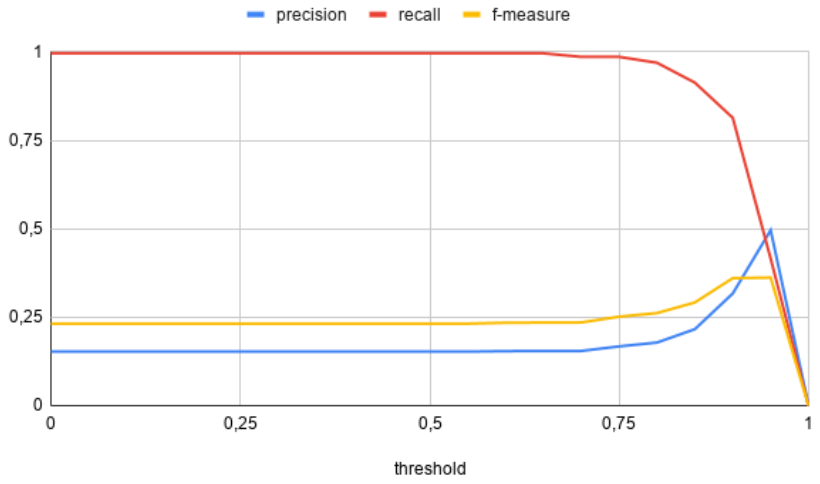

Figure 5.6: Precision (in blue), recall (in red) and F-measure (in yellow) with InferSent and the threshold at 0.9 for the "country" class.

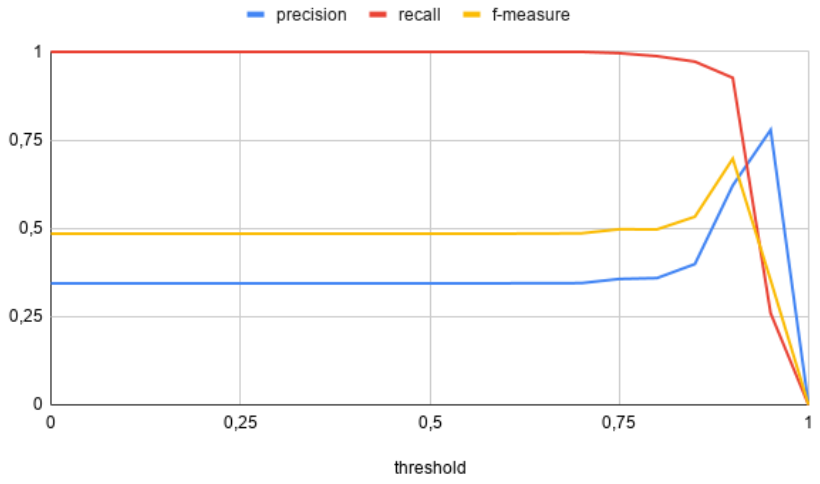

Figure 5.7: Precision (in blue), recall (in red) and F-measure (in yellow) with InferSent and the threshold at 0.9 for the "film" class.

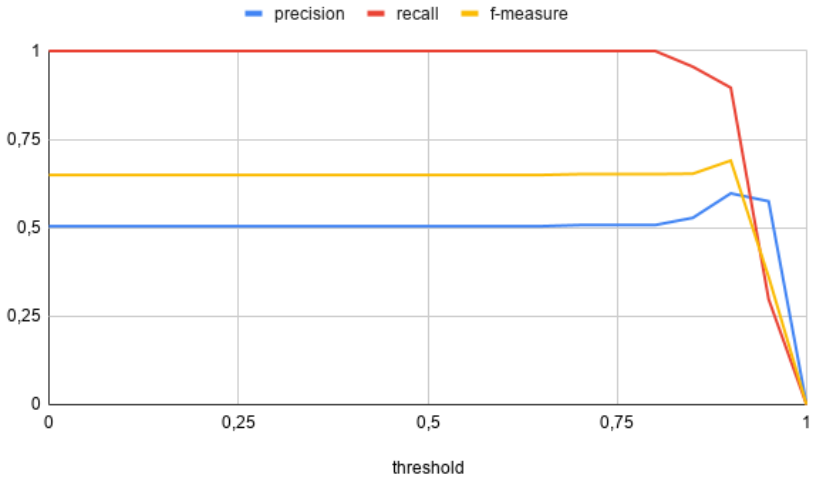

Figure 5.8: Precision (in blue), recall (in red) and F-measure (in yellow) with InferSent and the threshold at 0.9 for the "literary work" class.

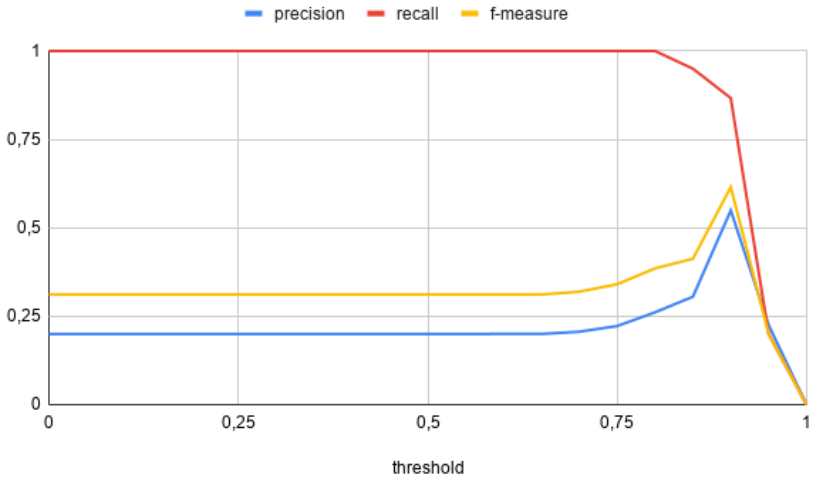

Figure 5.9: Precision (in blue), recall (in red) and F-measure (in yellow) with InferSent and the threshold at 0.9 for the "political party" class.

approach with a baseline method. Instead of computing embeddings of descriptions, we computed the Jaccard index (JI) [Jaccard 1908] of property descriptions. Since the JI is a metric distance, we can compute the standard mean of several JI values and still get a distance. Moreover, the corresponding similarity is defined as $distance = 1 - similarity$ and both distance and similarity are normalized, thus the similarity can be used in the same way as with cosine similarity of embedded vectors.

The first series of results is presented in the Figure 5.3. It shows the evolution of the average precision of the baseline (in blue), *InferSent* (in red), *GenSen* (in yellow) and *Universal Sentence Encoder* (in green). The threshold has been tested from 0 to 1 with steps of 0.05. For each threshold, we computed the standard mean of the precision for the 100 entities of the five classes (country, comics character, political party, literary work and film). The goal is to evaluate the suitability of the different embedding technic and the baseline. At first glance, we observe that the four methods produce the same results for very low thresholds with precisions in average at 0.3. This is because the median number of candidate properties is 12 and the median number of propagable properties (in Psi) is 3. Therefore, even if all properties were selected, we obtain thanks to the aforementioned precision formula $3/(3 + (12 - 3)) = 0.25$ as a minimum precision for any possible method of selection. The baseline quickly produces too many false positives without any peak when the threshold is above 0.05. Thus, as expected, the baseline is inadequate. For the three encoders, the same phenomenons append, but with very different threshold values and amplitudes. The three have a peak then a fall. For *GenSen* and *Universal Sentence Encoder* (USE), the peak is rather low and it can be explained by their difficulties to eliminate a sufficient number of false positives. Indeed, property descriptions are relatively close and some of them are well ordered (w.r.t. to their similarity), but not sufficiently to be useful to remove wrong candidates. The peak with *InferSent* is more interesting since it appears later and surpasses 0.5. Hence, *InferSent* can eliminate more false positives than the two others. When looking at the output of the algorithm, right candidates tend to be more grouped at the top of the list, while the two other encoders tend to mix right and wrong candidates. Moreover, because the peaks do not last, it means that the descriptions of the right candidates are very close, and for the three encoders the fall is more or less

sudden.

Figure 5.4 shows the evolution of the average recall of the approaches with the same colors as in Figure 5.3. As previously, the same pattern can be observed for all approaches except the baseline, but for very different thresholds. Indeed, immediately the baseline selects too many false negatives, demonstrating once again that it is not suitable to discriminate the right candidates from wrong ones. For the three encoders, at first, all properties are selected, thus there is no false negative. Because descriptions are close in terms of semantics, the cosine similarity produces similarities that are in a close range. Hence, (almost) all properties are detected as right candidates or (almost) none are. Then, very quickly for *GenSen* and USE, there is a sudden drop of the recall. Again, for both of them, the right candidates are distributed among the wrong candidates in a relatively uniform manner. Both encoders are unable to properly sort the right candidates on top of the list and the wrong ones at the bottom. While *InferSent* maintains its good selectivity a lot longer (until a threshold of 0.9). The F-measure of *GenSen* and USE never rises, to the contrary of *InferSent* that reaches 0.59. The latter produces vectors that are closer in their space than the two others, hence the range of cosine similarities is more compacted with *InferSent*. For example, for the entity "Wally West", with *InferSent* the highest similarity score with Π for a candidate property is 0.92 and the worst is 0.75. More important, *InferSent* is able to order more constantly and in a better way the properties. From those results, *InferSent* could be the right candidate to propose to the user an ordered list of candidate properties for propagation.

The second series of results, presented in Figures 5.6 and 5.7, illustrates the behavior of our approach with *InferSent*, since it produces the best results, for each of the five classes. As a reminder, for each class we randomly selected 20 entities, thus, for example, Figure 5.7 shows the average precision in blue of the 20 film entities, the average recall in red and the average F-measure in yellow. Of course, thresholds are the same as in the first series of results. The F-measure is always the better between 0.8 and 0.95, meaning that our approach is extremely sensitive to threshold variations. For all classes, the pattern is the same for the three measures. We only present two out of five classes to keep it concise, all results are available on the GitHub repository. The recall behaves the same for the

five classes, but for the precisions, there are more differences. Indeed, comics character and literary work have a quasi-flat part after the peak, meaning that descriptions of right candidates are more distant in the embedded space than the one's of other classes. Hence, similarities of both wrong and right candidates of country, film and political party entities are very close, since, after the peak, the fall is very sudden. Also, the fact that countries and political parties are described with many more properties on average appears here because their precisions start very below other precisions. It is more difficult for our approach to distinguish the right candidates from wrong ones when there are too many candidate properties. For comics characters and literary work entities, the approach seems to be less efficient in removing false positives, maybe because the range of similarity values is too narrow.

Since the choice of propagable properties is subjective for a given set of indiscernibility, three experts may not be sufficient. To establish the gold standard, a crowdsourcing approach might be more appropriate and should merit a formal investigation. As a result, the identity contexts of the gold standard could be more precise. Nevertheless, this approach could be used at least to present to the user an ordered list of candidate properties for propagation, hence helping her to make an educated decision. As a matter of fact, the good recall allows keeping almost all good candidates when the precision may help the user to quickly choose between available properties of the list.

However, we believe that our gold standard is sufficiently well built to state that our results are conclusive. False negatives are due to the fact that some properties are not semantically related to any indiscernible property of the context, and false positives ones are due to some properties that are semantically related to the context but not propagable. Hence, it is obvious that in some cases, considering only the semantic relatedness is a naïve approach. In addition to that, the lack of string-based description on a property is prohibitive, since our approach is based on property description consumption. Taking into account RDF semantics or using other embedding techniques should improve the results.

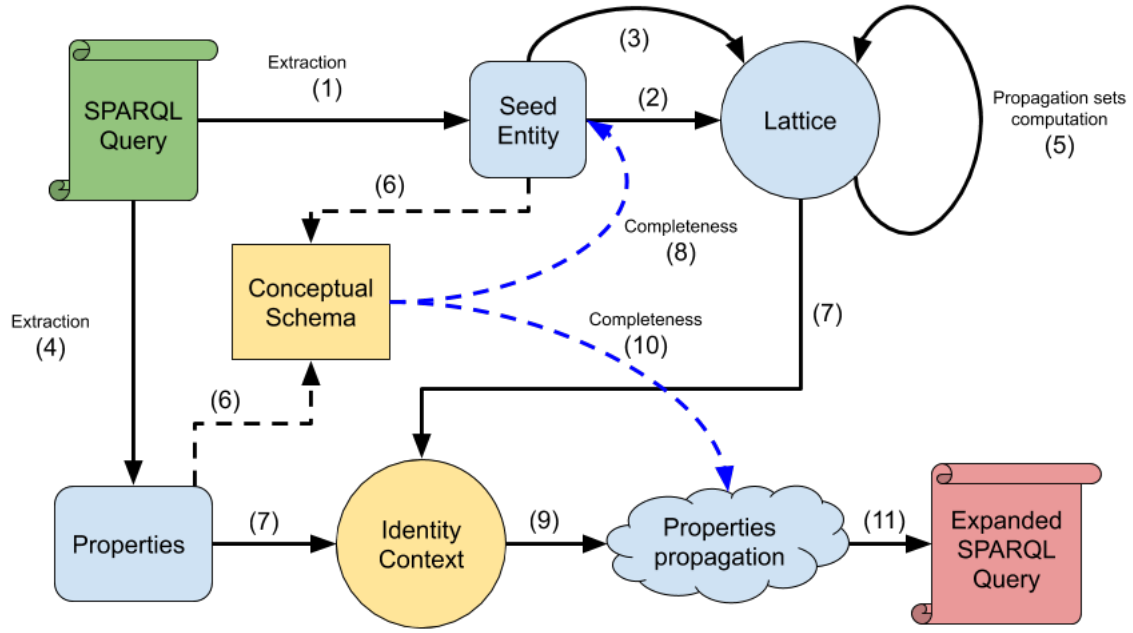

Figure 5.10: Qualitative experiment workflow: the elements in red are the inputs and the element in green is the output. To simplify the diagram, we consider only one instantiated entity linked to one instantiated property in the query.

5.4.3 Qualitative Study

In this section, we introduce three different queries that could benefit from our approach by extending their results. To achieve our goal, we used *InferSent* and a threshold value equal to 0.9. All of these queries are simplified queries tested on Wikidata (for ease of reading). The original queries can be found on the GitHub repository.

5.4.3.1 Task description

For each query, the goal is to find an identity context that will allow expanding the query with similar entities according to the user's objective. In this way, users can benefit from more complete results. The workflow is the following (see Figure 5.10): first, from the query, we extract the instantiated entity (or entities) that will be the seed(s) (step 1). Second, for each seed, we compute its identity lattice (steps 2, 3 and 5). As explained, we build the indiscernibility sets, then we get similar and logically identical entities, then we get candidate properties for propagation and finally, we get propagable properties (see

Section 5.3). Third, with the instantiated property (or set of properties) linked to the seed in the query, we select from the lattice, the node having this property in its propagation set (step 7). This node will be considered as the identity context of the query. Indeed, if multiple identity contexts are possible, the user must choose the best suited for its task purpose. Finally, based on the selected identity context, we can rewrite the query with the seed, logically identical entities and similar entities (steps 9 and 11). Furthermore, users can decide to check if the schema of the seed entity is more complete with LOD-CM (see Chapter 6) in steps 8 and 10.

5.4.3.2 Queries

We tested our approach with three queries. The first query in Listing 5.2 is about the "Paracetamol" drug. The query purpose is to retrieve all clinical trials of this drug. An interesting expansion of this query could be to find all trials of similar legal drugs in terms of medical conditions treated and physical interactions.

```
SELECT DISTINCT ?clinicalTrial WHERE
{
  ?clinicalTrial :researchIntervention :Paracetamol .
}
```

Listing 5.2: SPARQL query retrieving all studies about the painkiller named Paracetamol.

Table 5.1 shows additional results brought by our approach. Each column corresponds to a query. For the first query (and also for the next ones), there is only one seed "Paracetamol" ("France" and "the Republicans" in the second and the third columns respectively)as it is the only instantiated entity in the query. To fill this table, we first computed the lattice of the seed, then, selected a context containing the property "research intervention" in its Ψ. We chose as a context, legal drugs having the same medical conditions and the same physical interactions (obviously, any other context could be chosen depending on the users' needs). Finally, the query is expanded with similar entities as shown in Listing 5.3. The results show a 47% increase in the number of clinical trials for the considered context.

```
SELECT DISTINCT ?clinicalTrial WHERE
{
  VALUES (?drug)
  {
```

	Listing 5.2	**Listing 5.4**	**Listing 5.5**
Seed	Paracetamol	France	The Republicans
Ψ	research intervention	head of	member of political party
Π	condition treated, interacts with, legal status	capital, official language	country, political ideology
Similar entities	Ibuprofen Aspirin	French 2nd Republic, July Monarchy, French 3rd Republic, Bourbon Restoration, Kingdom of France, 2nd French Empire, Vichy France, French 4th Republic, 1st French Empire, Paris Commune	UMP, RPR, UDR, UNR UNR
# of results w/o context	586	12	2
# of results w/ context	860	99 (77)	13

Table 5.1: Identity context contribution to queries.

```
    (:Paracetamol) (:Ibuprofen) (:Aspirin)
  }
  ?clinicalTrial :researchIntervention ?drug .
}
```

Listing 5.3: Expanded SPARQL query retrieving all studies about Paracetamol similar entities.

The second query, in Listing 5.4, is about retrieving persons who once lead France. However, France has a complex history and has changed its political regime several times (for example, during World War II, or during the Napoleonian period). Thus, even if the French territory was "almost" always the same during the past centuries, each political regime has its own entity in Wikidata. Therefore, the query might not give all expected results. But if the user chooses the right identity context, i.e., $\mathcal{C}_{(\{capital, officialLanguage\},\{headOf\},\approx)}$ then all expected people will be retrieved.

```
SELECT DISTINCT ?headOfState WHERE
{
  ?headOfState :headOf :France .
```

```
}
```

Listing 5.4: SPARQL query retrieving all people who were head of the French state.

Similarly to the query about "Paracetamol", we computed the lattice and search for context with *headOf* in the propagable properties. The results are shown in the second column of Table 5.1. The expanded query could be rewritten as the previous one. It should be noted that among the 99 results, 22 persons were not head of France. Fourteen were head of Paris City Council and 8 were Grand Master of a Masonic obedience in France. This is due to the fact that the council and the obedience are misplaced in the Wikidata ontology. These errors cannot, therefore, be attributed to our approach. The results show a 542% increase in the number of France leaders for the considered context.

Finally, in Listing 5.5, we present a query about French politicians from The Republican party that have been convicted. The peculiarity here is that this major political party changed its name several times because of either political scandal or humiliating defeats. Consequently, if the knowledge graph is not up to date or not complete, some persons who were members of multiple versions of this party in the real world could not be actually linked to each version in the knowledge graph. This is the case of Wikidata that returns, for the query of Listing 5.5, only two politicians. However, there are more than a dozen politicians of this party who have been convicted of various felonies. By using our approach, it is possible to select a context composed of the political alignment and the country for which the *memberOf* property is propagable, and, hence, obtain a more complete result (of course depending on the completeness of data about politicians in Wikidata).

```
SELECT DISTINCT ?politician ?crime WHERE
{
  ?politician :memberOf :TheRepublicans ;
    :convictedOf ?crime .
}
```

Listing 5.5: SPARQL query retrieving all politicians member of French party named The Republicans that were convicted.

The same steps as for queries about "Paracetamol" and "France" were reproduced. Results are shown in the third column of Table 5.1. The results show a 550% increase in the number of convicted politicians for the considered context.

5.4.4 Discussion

As we have seen, our approach allows discovering propagable properties for a given indiscernibility set of properties Π. An identity context with its indiscernibility and propagation sets can provide more complete answers to queries through query expansion. The results are very promising but need to be confronted with more different kinds of knowledge graphs and a combination of distinct knowledge graphs. Also, our approach does not work well when the property of an entity lack property describing it (such as *rdfs:comment* or *schema:description*). It is a limitation since some ontologies do not provide textual descriptions of their properties. Hence, the first step for future work is to circumvent this flaw with a multi-faceted approach. Moreover, in a textual description, some words might be irrelevant (like a Wikidata identifier) and degrade the quality of the results.

5.5 Conclusion

In this chapter, we proposed an approach based on sentence embedding to discover propagable properties given an indiscernibility set of properties. Our approach computes, for an entity, an identity lattice that represents all its possible identity contexts, i.e., both indiscernible and propagable properties. Qualitative and quantitative studies have been conducted to evaluate the approach. Besides, an important part of our work was to make available a gold standard for the reproducibility of the quantitative study and, in general, for the research community working on contextual identity.

Chapter 6

Effects of Contextual Propagation on Entity Schema Completeness

---

This chapter is based on the following publications:

- Subhi Issa, Pierre-Henri Paris, and Fayçal Hamdi. Assessing the completeness evolution of DBpedia: A case study. In Sergio de Cesare and Ulrich Frank, editors, *Advances in Conceptual Modeling - ER 2017 Workshops AHA, MoBiD, MREBA, OntoCom, and QMMQ, Valencia, Spain, November 6-9, 2017, Proceedings*, volume 10651 of *Lecture Notes in Computer Science*, pages 238–247. Springer, 2017. doi: 10.1007/978-3-319-70625-2_22. URL `https://doi.org/10.1007/978-3-319-70625-2_22`

- Subhi Issa, Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Revealing the conceptual schemas of RDF datasets. In Paolo Giorgini and Barbara Weber, editors, *Advanced Information Systems Engineering - 31st International Conference, CAiSE 2019, Rome, Italy, June 3-7, 2019, Proceedings*, volume 11483 of *Lecture Notes in Computer Science*, pages 312–327. Springer, 2019. doi: 10.1007/978-3-030-21290-2_20. URL `https://doi.org/10.1007/978-3-030-21290-2_20`

- Subhi Issa, Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Revealing the conceptual schemas of RDF datasets - extended abstract. In *INFORSID 2020*, 2020 (To be published)

One of the objectives of establishing the identity between two entities is to be able to reuse information, i.e., to increase the completeness of an entity using one or more other entities. This completeness can take two forms: *(i)* at the schema level of the entity, i.e., the number of different properties it uses, and *(ii)* at the data level, i.e., the number of values a property can have. Measuring data completeness is very difficult since it is almost impossible to establish a gold standard for data in the Semantic Web domain since it is governed by the open-world assumption ([Darari, Nutt, Pirrò, and Razniewski 2013]). Hence, we measure the effects on the completeness, at the schema level, of the propagation of properties in a given identity context. Thus, the objective of the approach proposed in this chapter is to generate a conceptual schema. This schema will allow measuring the completeness of the schema of an entity. The completeness could be measured before and after the application of our general property propagation approach (see Chapter 5).

RDF-based knowledge graphs, thanks to their semantic richness, variety, and fine granularity, are increasingly used by both researchers and business communities. However, these knowledge graphs suffer a lack of completeness as the content evolves continuously, and data contributors are loosely constrained by the vocabularies and schemas related to the data sources. Conceptual schemas have long been recognized as a key mechanism for understanding and dealing with complex real-world systems. In the context of the Web of Data and user-generated content, the conceptual schema is implicit. Each data contributor has an implicit personal model that is not known by the other contributors. Consequently, revealing a meaningful conceptual schema is a challenging task that should take into account the data and the intended usage. In this chapter, we propose a completeness-based approach for revealing conceptual schemas of RDF data. We combine quality evaluation and data mining approaches to find a conceptual schema for a knowledge graph. This model meets user expectations regarding data completeness constraints. To achieve that, we propose a web-based completeness demonstrator for knowledge graphs: LOD-CM.

6.1 Introduction

Data became a strategic asset in the information-driven world. One of the challenges for companies and researchers is to improve the display and understandability of the data

they manage and use.

However, exploiting and using open Linked Data, even if it is more and more accessible, is not an easy task. Data is often incomplete and lacks metadata. These issues mean that the quality of published data is not as good as we could expect, leading to a low added value and low reliability of the derived conclusions. In [Jain, Hitzler, Yeh, Verma, and Sheth 2010], the authors believe that existing approaches that describe knowledge graphs focus on their statistical aspects rather than on capturing conceptual information.

A conceptual schema is an abstraction of a reality that can serve as a vehicle for understanding, communicating, reasoning, and adding knowledge about this reality.

In traditional information system development, conceptual modeling is driven by intended usage and needs. For knowledge graphs, as in all user-generated content, data is rather use-agnostic [Lukyanenko and Parsons 2015]. As a result, the data is represented according to many individual points of view. These points of view lead to a lack of semantics, whereas semantics is necessary for reasoning about the data. We believe that a conceptual schema that creates an abstract representation upon data would help to overcome the disparity of visions and will reveal the underlying semantics [Olivé 2007]. Let us consider, for instance, that we have a collaboratively built knowledge graph. In this case, the traditional top-down vision of a predefined schema is no more applicable. Both data and underlying schema evolve continuously, as several communities describe data with different views and needs. In this situation, a conceptual schema, defined as an abstract and consensual representation about the reality that is derived from requirements, could not be applied. The challenge is then to find a way to create a suitable conceptual schema having entities as a starting point.

The **research questions** of this chapter are as follows:

- How to compute the schema completeness of an entity in an RDF-based knowledge graph?
- How to facilitate access to the information structure of a knowledge graph?

In this chapter, we are interested in the conceptual modeling of RDF-based knowledge

graphs [Klyne and Carroll 2006]. Our objective is to define an approach for deriving conceptual schemas from existing data. The proposed solution should cope with the essential characteristics of a conceptual schema that are the ability to make an abstraction of relevant aspects from the universe of discourse and the one of meeting user's requirements [Rolland and Prakash 2000]. The approach we propose in this chapter takes into account the two facets, namely the universe of discourse represented by the data sources, and the user's needs represented by the user's decisions during the conceptual schema construction. As the model should express the meaningful state of the considered knowledge graph, we rely on a mining approach leading to taking into consideration the data model from a more frequent combination of properties. The relevancy of properties is handled by integrating a completeness measurement solution that drives the identification of relevant properties. To meet user's requirements, we propose to construct the conceptual schema by allowing the user to decide about the parts of the conceptual schema to reveal according to her needs and constraints.

The main contributions are:

1. We use a mining approach to infer a model from data, as we consider that no predefined schema exists. The underlying assumption is that the more frequent a schema is, the more representative for the knowledge graph it is.

2. We introduce a novel approach, called *LOD-CM*, for conceptual schema mining based on quality measures, and, in this chapter, on completeness measures as a way to drive the conceptual schema mining process.

The remainder of this chapter is organized as follows: Section 6.2 summarizes related literature on the subject while Section 6.3 details the mining-based approach for RDF data conceptual modeling. This section explains the tight link with the completeness quality criterion. Section 6.4 presents two use cases of LOD-CM. Finally, Section 6.5 draws conclusions.

6.2 Related work

RDF data is described as sets of statements called *triples*. A triple $\langle s, p, o \rangle$ is a fact where a subject s has a property p, and the property value is the object o. As an example, $\langle England, capital, London \rangle$ means that London is the capital city of England. Understanding and reasoning about this data require at least knowledge about its abstract model. Consequently, schema discovery has attracted several researchers originating from several communities. The research directions address objectives such as efficient storage, efficient querying, navigation through data or semantic representation, etc.

Completeness of Linked Data is one of the most critical data quality dimensions ([Batini and Scannapieco 2016]). This dimension is defined as the degree to which all required information is present in a particular knowledge graph ([Zaveri, Rula, Maurino, Pietrobon, Lehmann, Auer, and Hitzler 2013]). We have to know that a reference schema (or a gold standard) should be available to compare against a given knowledge graph.

In the database community, the question was how to store this kind of data. [Levandoski and Mokbel 2009] proposed a solution that derives a classical relational schema from an RDF data source to accelerate the processing of queries. In the FlexTable method ([Wang, Du, Lu, and Wang 2010]), authors proposed to replace RDF triples by RDF tuples resulting from the unification of a set of triples having the same subject. All these approaches do not target a human-readable schema and are more concerned with providing a suitable structure for a computer processing of data.

The Semantic Web community is more aware of data semantics through the usage of *ontologies* and *vocabularies*. Several semi-automatic or automatic proposals, mainly based on classification, clustering, and association analysis techniques are proposed. In [Völker and Niepert 2011] a statistical approach based on association rules mining allows generating ontologies from RDF data. Other works, such as those presented in [Christodoulou, Paton, and Fernandes 2015; Pham, Passing, Erling, and Boncz 2015; Kellou-Menouer and Kedad 2015], are closer to modeling. The authors propose to derive a data structure using a clustering algorithm. After manual labeling of clusters representing groups of frequent properties, a schema is derived. These approaches, however, do not consider the user's

needs and preferences, and the derived schema is the result of automatic preprocessing, apart from the labeling task.

In traditional conceptual modeling, models are generally derived from user's requirements. However, with the increasing use of external data sources in information systems, there is a need to apply bottom-up modeling from entities. This is motivated by the expressiveness and the analysis facilities that conceptual schemas could provide for such data. Similarly to our bottom-up approach, [Lukyanenko, Parsons, and Samuel 2019] proposed a conceptual modeling grammar based on the assumption that entities play a major role while human beings try to represent reality. In [Lukyanenko and Parsons 2015], the authors presented a set of principles for conceptual modeling within structured user-generated content. The authors highlighted the problem of quality in such produced content. They focused on the importance of capturing relevant properties from entities. However, the proposal does not provide an explicit solution for deriving such models. Concerning unstructured data, we can cite [Embley and Liddle 2013], where authors addressed the problem of deriving conceptual schemas based on regular-expression pattern recognition.

Recognizing that conceptual modeling is a powerful tool for data understanding, our proposal addresses the problem of deriving a conceptual schema from RDF data. By exploring entities, our approach integrates a completeness measurement as a quality criterion to ensure the relevancy of the derived schema as data from RDF data sources is the result of a free individual publication effort. The result would be a conceptual schema enriched with completeness values.

6.3 Conceptual schemas derivation

To illustrate our proposed approach, let us consider a user willing to obtain a list of artists with their names and birthplaces from an RDF-based knowledge graph; To do so, she can write the following SPARQL query[1]:

```
SELECT * WHERE
{
```

[1]Performed on: `http://dbpedia.org/sparql`

```
  ?actor rdf:type dbo:Actor .
  ?actor foaf:name ?name .
  ?actor dbo:birthPlace ?birthPlace .
}
```

Listing 6.1: SPARQL query retrieving all actor names and birthplaces.

Writing such a query is much more difficult in a Linked Open Data (LOD) source context than in a relational database one. In a relational context, the database schema is predefined, and the user writing the query is aware of it. With knowledge graphs, the schema does not exist. Moreover, there is another problem related to data completeness: The expressed query returns only the list of actors having values for all the properties listed in the query. In our example, only actors having values for both *foaf:name* and *dbo:birthPlace* are included in the result. Knowing that at most 74% of actors have a value for *dbo:birthPlace*, the user should probably appreciate getting this information to add, for example, `OPTIONAL` to the second pattern of the query and obtain more results. Besides, she would be aware of the fact that the result is complete to a certain degree (i.e., *dbo:birthPlace* is present in only 74% of actors).

To tackle these two problems, we propose an approach that aims to help "revealing" a conceptual schema from an RDF-based knowledge graph. This conceptual schema is driven by the user for both its content and completeness quality values.
In the context of the Web of Data, most of the knowledge graphs published in the Web are described by models called, in Linked Data jargon, vocabularies (or ontologies). However, these models are not used in a prescriptive manner. Consequently, a person who publishes data is not constrained by the underlying ontology leading to sparse descriptions of concepts. For example, the instances of class *Actor* from DBpedia use around 532 properties that are not equally relevant.

From these observations, it is clear that checking data (entities) is necessary to infer a relevant model that can be used to guarantee, for example, an expected completeness value. The approach that we propose deals with this issue through an iterative process, which infers a conceptual schema complying with the expected completeness. Figure 6.1 gives an overview of this process.

The process of inferring a conceptual schema goes through four steps: First, a subset

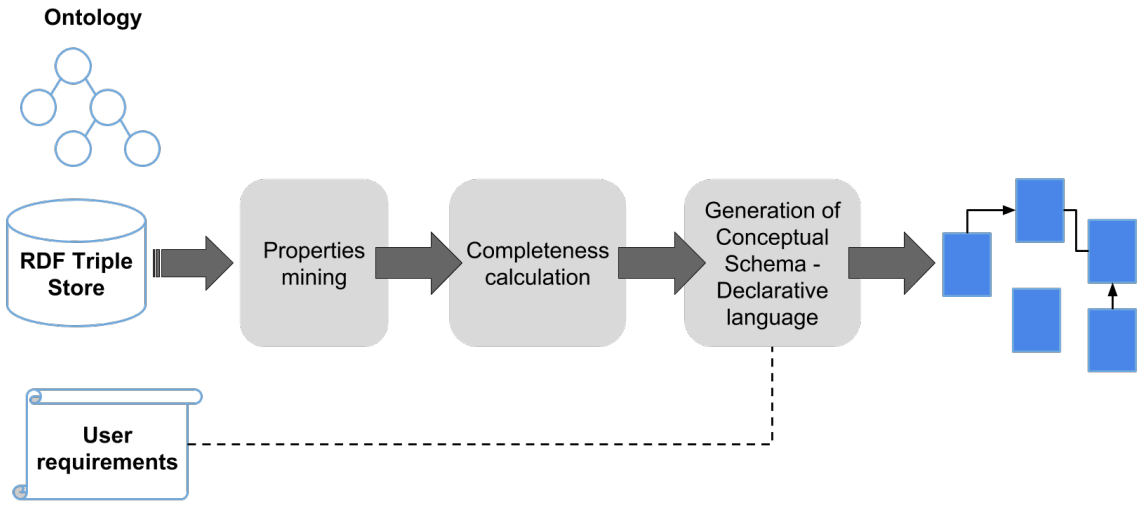

Figure 6.1: The *LOD-CM* Workflow

of data that corresponds to the user's scope is extracted from the knowledge graph (cf. Section 6.3.1). This subset is then transformed into transactions, and a mining algorithm is applied. In our approach, for efficiency reasons, we chose the well-known FP-growth algorithm [Han, Pei, and Yin 2000; Han, Pei, Yin, and Mao 2004] (any other itemset mining algorithm could obviously be used). From the generated frequent itemsets, only a subset of these frequent itemsets, called "Maximal" [Jr. 1998; Gouda and Zaki 2001; Grahne and Zhu 2003], is captured. This choice is motivated by the fact that, on the one hand, we are interested in the *expression* of the frequent pattern, and, on the other hand, the number of frequent patterns could be exponential when the transaction vector is huge (cf. Section 6.3.2). $\mathcal{MFP}$ is the set containing all maximal frequent patterns. Next, each pattern in $\mathcal{MFP}$ is used to calculate the completeness of each transaction. The presence or absence of the pattern is reflected in the completeness. Hence, the completeness of the whole knowledge graph regarding this pattern can be computed by aggregating transaction completenesses. The final completeness value will be the average of all completeness values calculated for each $\mathcal{MFP}$ pattern (cf. Section 6.3.3). Finally, based on the completeness value and $\mathcal{MFP}$ that guarantees this value, a conceptual schema is generated. The classes, the attributes, and the relations of the model will be tagged with the completeness value (cf. Section 6.3.4). All these steps are integrated into an iterative process: the user could

choose some parts in the generated model to refine. The data corresponding to the parts to refine is then extracted from the knowledge graph, and the same steps are carried out to generate a new model.

In the following subsections, we give a detailed description of each step of the workflow.

6.3.1 Scope and Completeness Specification

In this step, a subset of data is extracted from the knowledge graph. This subset could correspond to a class or a set of classes such as *Actor*, *Film* , or *Organization*. The subset defines what we call the user's scope that corresponds to the classes that the user plans to use in a query, or to the information she wants to explore or any kind of usage based on data consumption.

The user is also asked to indicate the degree of the desired completeness. Indeed, properties for a given class are not equally used. For example, for the class *Artist*, the property *foaf:name* has a value for 99% of the entities, whereas the *dbo:birthPlace* property has a value for at most 74% of the entities. Our approach gives the possibility to express a constraint on the completeness values desired for mined properties and associations. Once the classes are identified, the data is converted into transaction vectors, and a mining algorithm is applied to obtain a set of frequent itemsets.

Table 6.1 illustrates some entities of the *Film* class in the form of triples, taken from DBpedia. Each class is described by a set of properties (predicates). An entity of this class could have a value for all or a subset of these properties. This subset is called a transaction. Table 6.2 represents the set of transactions constructed from the triples of Table 6.1.

More formally, given a knowledge graph $\mathcal{KG}$, let us define a set of classes $C \in \mathcal{KG}$ (e.g., *Film*, *Artist*), $\mathcal{E}_C \in \mathcal{KG}$ is the set of entities for classes in C (e.g., *The_Godfather* is an entity of the *Film* class), and $P_C = \{p_1, p_2, ..., p_n\} \in \mathcal{KG}$ is the set of properties used by entities in $\mathcal{E}_C$ (e.g $director(Film, Person)$).

Given a subset of entities $E = \{e_1, e_2, ..., e_m\}$ with $E \subseteq \mathcal{E}_C$ (e.g., properties used to describe the *The_Godfather* entity are: *director* and *musicComposer*), $\mathcal{T}_E = (t_1, t_2, ..., t_m)$ is the list of transactions where $\forall k, 1 \leq k \leq m : t_k \subseteq P_C$ and t_k is the set of properties used in the description of $e_k \in E$, i.e., $\forall p \in t_k, \exists o \in \mathcal{KG} : \langle e_k, p, o \rangle \in \mathcal{KG}$. We consider $\mathcal{CP}$

the completeness of E against properties used in the description of each of its entities. Moreover, $\mathcal{P}(t_k)$ is the power set of transaction t_k.

6.3.2 Properties Mining

All statements having a subject from a class C_1 are grouped. The related properties of those statements could consequently constitute either the attributes (properties) of the class C_1 or relationships to other classes when the property value (the object in the triple $< s, p, o >$) refers to another class. In this step, the objective is to find the properties patterns that are the most shared by the subset of entities extracted from the knowledge graph. This set will be then used to calculate a completeness value regarding these patterns. Let $C, \mathcal{E}_C, P_C$ be respectively classes, instances (of classes in C), and properties (of entities in $\mathcal{E}_C$) of a knowledge graph $\mathcal{KG}$ and E be a subset of data (entities) extracted from $\mathcal{KG}$ with $E \subseteq \mathcal{E}_C$. We first initialize $\mathcal{T}_E = \emptyset$, $\mathcal{MFP} = \emptyset$. For each $e \in E$, we generate a transaction t, i.e., properties used by e. Indeed, each entity e is related to values (either resources or literals) through a set of properties. Therefore, a transaction t_k of an entity e_k is a set of properties such that $t_k \subseteq P_C$. Transactions generated for all the entities of E are then added to the $\mathcal{T}_E$ list.

Example 17 *Referring Table 6.1, let E be a subset of entities such that:*
$E = \{The\_Godfather, Goodfellas, True\_Lies\}$. The list of transactions $\mathcal{T}_E$ would be:

$$\mathcal{T}_E = (\{director, musicComposer\}, \{director, editing\},$$
$$\{director, editing, musicComposer\})$$

The objective is then to compute the set of frequent patterns $\mathcal{FP}$ from the transaction vector $\mathcal{T}_E$.

Definition 6.3.1 ***(Pattern)*** *Let $\mathcal{T}_E$ be a set of transactions. A pattern $\hat{P}$ is a sequence of properties shared by one or several transactions t in $\mathcal{T}_E$. It is sometimes called an* itemset.

Subject	Predicate	Object
The_Godfather	director	Francis_Ford_Coppola
The_Godfather	musicComposer	Nino_Rota
Goodfellas	director	Martin_Scorsese
Goodfellas	editing	Thelma_Schoonmaker
True_Lies	director	James_Cameron
True_Lies	editing	Conrad_Buff_IV
True_Lies	musicComposer	Brad_Fiedel

Table 6.1: A sample of triples from DBpedia

entity	Transaction
The_Godfather	director, musicComposer
Goodfellas	director, editing
True_Lies	director, editing, musicComposer

Table 6.2: Transactions extracted from triples

For any pattern $\hat{P}$, let $\mathscr{P}(\hat{P})$ be the power set of $\hat{P}$ (composed, in our case, of properties), and $T(\hat{P}) = \{t \in \mathcal{T}_E \mid \mathscr{P}(\hat{P}) \subseteq \mathscr{P}(t)\}$ be the corresponding set of transactions. $\mathscr{P}(\hat{P})$ designates the *expression* of $\hat{P}$, and $|T(\hat{P})|$ the *support* of $\hat{P}$. A pattern $\hat{P}$ is frequent if $\frac{1}{|\mathcal{T}_E|}|T(\hat{P})| \geq \xi$, where ξ is a user-specified threshold.

Example 18 *Referring Table 6.2, let* $\hat{P} = \{director, musicComposer\}$ *and* $\xi = 60\%$ *.* $\hat{P}$ *is frequent as its relative support (*66.7%*) is greater than* ξ*.*

To find all the frequent patterns $\mathcal{FP}$, we used, as we mentioned above, the FP-growth itemsets mining algorithm. However, according to the size of the transactions vector, the FP-growth algorithm could generate a very large $\mathcal{FP}$ set. As a reminder, our objective is to see how a transaction (a description of an entity) is *complete* against a set of properties. Thus, we focus on the pattern *expression* (in terms of items it contains) instead of its *support*.

For completeness calculation, we need to select a pattern to serve as a reference schema. This pattern should present the right balance between frequency and expressiveness. Therefore we use the concept, called "Maximal" frequent patterns, to find this subset. Thus, to reduce $\mathcal{FP}$, we generate a subset containing only "Maximal" patterns.

Definition 6.3.2 *($\mathcal{MFP}$) Let* $\hat{P}$ *be a frequent pattern.* $\hat{P}$ *is maximal if none of its proper*

superset is frequent. We define the set of Maximal Frequent Patterns $\mathcal{MFP}$ as:

$$\mathcal{MFP} = \{\hat{P} \in \mathcal{FP} \mid \nexists \hat{P}' \in \mathcal{FP} : \hat{P} \subset \hat{P}' \wedge \frac{|T(\hat{P}')|}{|\mathcal{T}_E|} < \xi\}$$

Example 19 *Referring Table 6.2, let $\xi = 60\%$ and the set of frequent patterns $\mathcal{FP}$ = {{director},{musicComposer},{editing}, {director,musicComposer},{director,editing}}. The $\mathcal{MFP}$ set would be:*

$$\mathcal{MFP} = \{\{director, musicComposer\}, \{director, editing\}\}$$

6.3.3 Completeness calculation

In this step, we carry out for each transaction a comparison between its corresponding properties and each pattern of the $\mathcal{MFP}$ set (regarding the presence or the absence of the pattern). An average is, therefore, calculated to obtain the completeness of each transaction $t \in \mathcal{T}_E$. Finally, the completeness of the whole $t \in \mathcal{T}_E$ will be the average of all the completeness values calculated for each transaction.

Definition 6.3.3 *(*Completeness*) Let E be a subset of entities, $\mathcal{T}_E$ the set of transactions constructed from E, and $\mathcal{MFP}$ a set of maximal frequent pattern. The completeness of E corresponds to the completeness of its transaction vector $\mathcal{T}_E$ obtained by calculating the average of the completeness of $\mathcal{T}_E$ regarding each pattern in $\mathcal{MFP}$. Therefore, we define the completeness $\mathcal{CP}$ of a subset of entities E as follows:*

$$\mathcal{CP}(E) = \frac{1}{|\mathcal{T}_E|} \sum_{k=1}^{|\mathcal{T}_E|} \sum_{j=1}^{|\mathcal{MFP}|} \frac{\delta(\hat{P}_j, \mathcal{P}(t_k))}{|\mathcal{MFP}|} \tag{6.1}$$

$$\textit{such that: } \hat{P}_j \in \mathcal{MFP}, \textit{ and}$$

$$\delta(\hat{P}_j, \mathcal{P}(t_k)) = \begin{cases} 1 & \textit{if } \hat{P}_j \subset \mathcal{P}(t_k) \\ 0 & \textit{otherwise} \end{cases}$$

Algorithm 6 shows the pseudo-codes for calculating $\mathcal{CP}(E)$.

Example 20 *Let $\xi = 60\%$. The completeness of the subset of entities in Table 6.1 regarding $\mathcal{MFP}$ = {{director,musicComposer}, {director,editing}} would be:*

$$\mathcal{CP}(E) = \frac{2 \times (1/2) + (2/2)}{3} = 0.67$$

This value corresponds to the completeness average value for the whole knowledge graph regarding the inferred patterns in $\mathcal{MFP}$.

input : $\mathcal{KG}$, E, ξ
output : $\mathcal{CP}(E)$

foreach $e \in E$ **do**
  $t_i = \left|p_1 \quad p_2 \quad \dots \quad p_n\right|$;
  $\mathcal{T}_E = \mathcal{T}_E + t_i$;
end
/* *Properties mining* */
$\mathcal{MFP} = Maximal(FP\text{-}growth(\mathcal{T}_E, \xi))$;
/* *Using equation 6.1* */
return $CP(E) = CalculateCompleteness(E, \mathcal{T}_E, \mathcal{MFP})$

Algorithm 6: Completeness calculation

6.3.4 Generation of Enriched Conceptual Schemas

In this step, the goal is to generate a conceptual schema enriched with the completeness values calculated in the previous step. The $\mathcal{MFP}$ patterns used to get the completeness values are transformed into a class diagram. Figure 6.2 illustrates the user's interface of our LOD-CM web service. Using the graphical interface [2], the user can choose her constraints. The web service permits the user to enter the class name in the text box, and the user may select the threshold completeness she wants to apply. Currently, our demo supports DBpedia and Wikidata knowledge graphs.

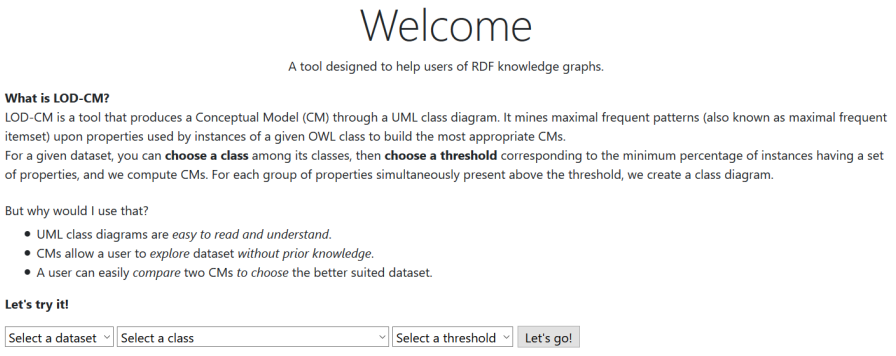

Figure 6.2: LOD-CM main interface

[2] `http://cedric.cnam.fr/lod-cm`

After the user selects the class name and desired completeness and clicks the "Submit" button, the algorithm runs to find the attributes, relationships, and the missed domains/ranges based on user's constraints.

The structure of the model is constructed regarding the definitions of the properties of patterns in the ontology. Figure 6.3 represents a class diagram derived by our approach, from a set of films extracted from DBpedia.

In this example, the expectation of the user is a model that guarantees at least 50% of completeness. To generate the model, the first step consists of obtaining the set of properties $p \in \bigcup_{j=1}^{n} \mathcal{P}(\hat{P}_j)$, and $\hat{P}_j \in \mathcal{MFP}$ that composes the union of all the $\mathcal{MFP}$, mined from the extracted subset, with a minimum support $\xi = 50\%$. For this example, the set of properties are: {*director, label, name, runtime, starring, type*}, {*director, label, name, starring, type, writer*} and {*label, name, runtime, type, writer*}. OWL 2 distinguishes between two main classes of properties: *(i)* datatype properties, where the value is a literal, and *(ii)* object properties, where the value is an individual (i.e., another entity of a different class). Each property is considered as an attribute (e.g., name) of the class or a relationship (e.g., director) with another class. Two types of links will be used during the generating of conceptual schemas: inheritance and association links. Inheritance link describes the relationship between the class and the superclass, and the association link describes the relationship between two classes and points to the property. A dotted link was added to illustrate that a class has been inferred to complete the relationship. For this reason, based on the approach that has been proposed in [Töpper, Knuth, and Sack 2012], we infer missed domains (and ranges) of properties. In our example, the class names and the inheritance links between the classes are derived from class names and subsumptions described in the DBpedia ontology. We do not derive new class names nor new subsumptions as the conceptual schema should conform to the data used. Indeed, even if the derived conceptual schema is not satisfactory from conceptual modeling principles, it should faithfully reflect the reality of data while taking into account user preferences. Finally, the diagram is enriched by the completeness values calculated in the previous step. These values are associated with each component of the model.

A new iteration is triggered when the user chooses to get more details about a part of

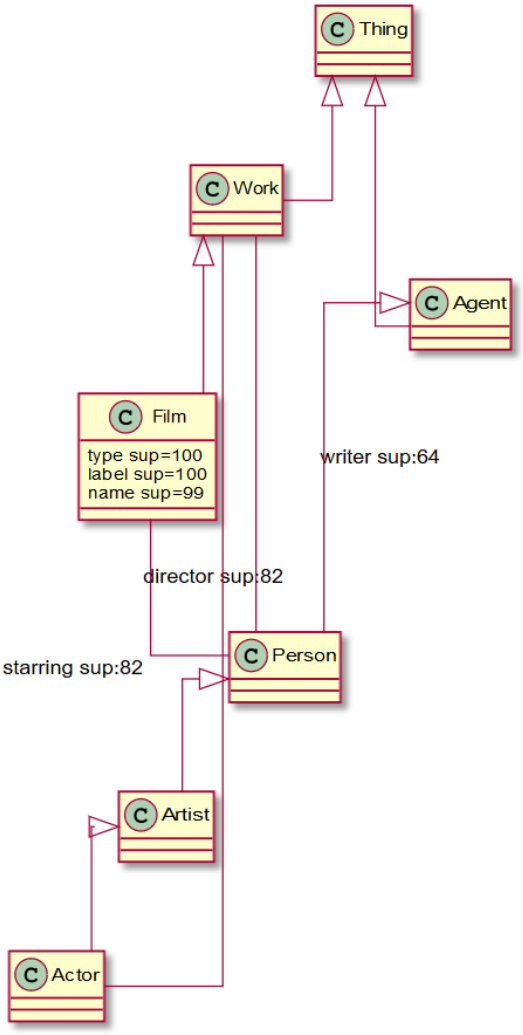

Figure 6.3: The *Film* conceptual schema as a class diagram

the model (e.g., the *Artist* class, see Fig. 6.4). In this case, a new query is executed on the knowledge graph to extract data corresponding to this part. The previous three steps are then executed to generate a new model integrating the new desired details. Figure 6.5 shows an example that details a part of the model from Figure 6.3. In this example, a set of classes, relationships, and attributes are added to the *Artist* class with corresponding completeness values. This way of revealing the conceptual schema is similar to a magnifying glass that allows the user to navigate around a targeted concept, here the *Film* class.

The output of our algorithm is a file written in a declarative language. The file includes the chosen class, the attributes, and the relationships tagged by completeness values. We

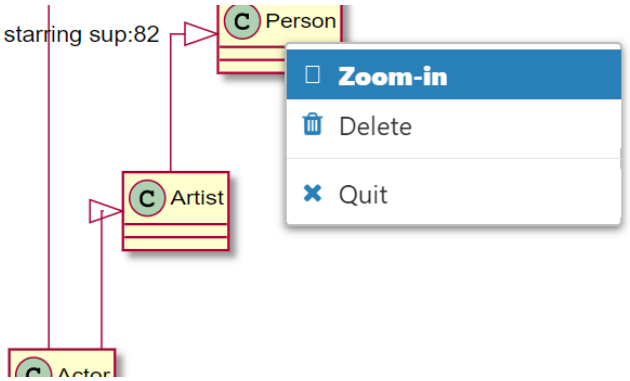

Figure 6.4: Contextual menu for navigation and editing.

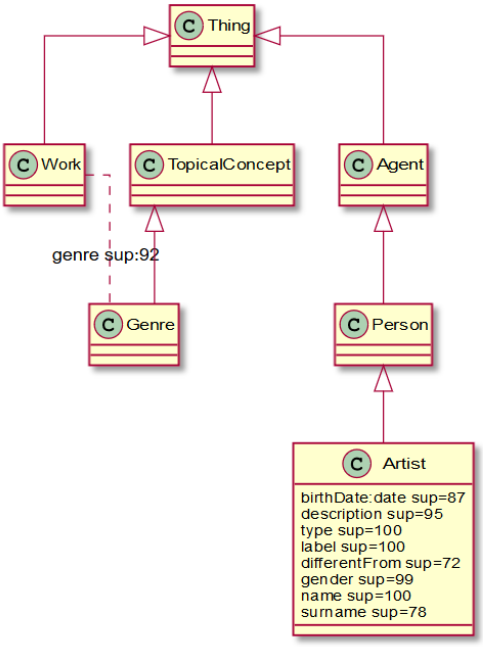

Figure 6.5: The *Artist* diagram class

use PlantUML [3] to transfer this generated file into a picture to illustrate it to the user.

6.4 Use cases

The objective of the Linked Open Data cloud is to enable large-scale data integration so that we can have a contextually relevant Web and find quick answers to a much wider range of questions. LOD-CM is a web-based completeness demonstrator for RDF-based

[3] `http://plantuml.com/`

knowledge graphs. It is used to display data related to the chosen class of a knowledge graph. In this section, we provide a summary of two use cases related to schema discovery based on user's needs. The displayed model could help the user to understand the schema and discover the related properties. LOD-CM only supports two knowledge graphs that are DBpedia and Wikidata.

6.4.1 Class diagram to facilitate data browsing

LOD-CM aims to visualize the discovered schema based on the user's requirements. Suppose a user wants to find the directors and budgets of a list of films. 82% of films have a director in the DBpedia knowledge graph. Besides, only 15% of films have budget value for the same knowledge graph. Only the list of films that have the properties (director and budget) will be displayed (i.e., at most 15% of the films). The outcome model could help the user to present the properties that are related to the chosen class in a proportion greater than a specified threshold. Besides, it illustrates the relationship between the concerned classes. For example, the classes *Person* and *Film* are linked by the property *director*. Furthermore, the model illustrates the inheritance relationship, such as *Artist* is a subclass of *Person*.

6.4.2 Discovering a subset of MFP

As mentioned in Section 6.3.2, our goal is also to find the set of properties that can be used together in the query and does not exceed the selected threshold. For example, for the *Film* class with 60% of completeness, four sets of properties are greater than 60% {{type, name, label, director, writer}, {type, name, label, director, runtime}, {type, name, label, director, starring},{type, name, label, runtime, starring}} For this reason, our LOD-CM interface enables the user to check the desired properties that appear in the returned model. It should be noted that the property which does not achieve the completeness threshold with other selected properties will be inactivated, such as *starring* and *writer* in our previous example. This case could help the user to be sure that the returned results for its query with this set of properties are equal or greater than the desired threshold.

Finally, Table 6.3 shows the number of properties we get (at the end of our pipeline)

Class/threshold	0.1	0.3	0.5	0.7	0.9
Film	18	12	7	6	3
Settlement	18	14	8	5	4
Organisation	18	4	4	3	3
Scientist	19	16	12	9	5

Table 6.3: DBpedia number of predicates by classes and thresholds

for several classes according to several thresholds. The lower the threshold is, the more properties there are. Thus, lower thresholds produce more complex conceptual schemas but with more noise. Hence, this tool can help the user to find the right balance between those two.

6.4.3 Application to our propagation framework

As explained in Chapter 5, our propagation approach of properties in contextual identity enables to increase the schema completeness of an entity. Moreover, LOD-CM enables the generation of a conceptual schema for a given class. Hence, one can measure the completeness of an entity against its class conceptual schema. This measure can be done before and after the application of the propagation approach to control the potential increase.

For example, in DBpedia, the french political party Rally for the Republic has no *country* property. However, the conceptual schema of the *political party* class contains a link to the *country* class (see Figure 6.6). The completeness computed against the conceptual schema before the approach is 75%. When choosing the *ideology* property as indiscernibility set Π, one also get the *country* property in the propagation set Ψ. Hence, applying our approach enables to propagate the *country* property to discover that the Rally for the Republic is a french political party. After using our propagation approach, the completeness is 100%.

6.5 Conclusion

We have presented an approach for revealing conceptual schemas from RDF knowledge graphs. The approach is an iterative process that computes a plausible model from the

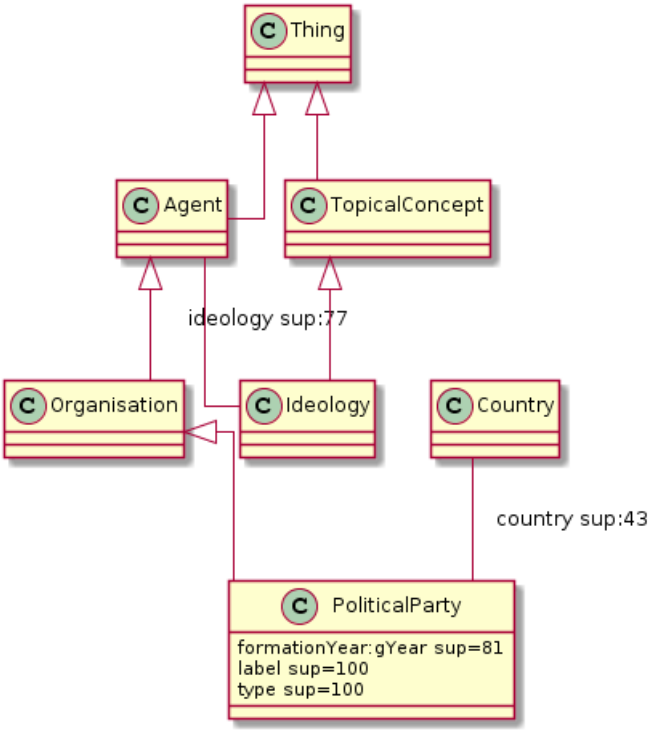

Figure 6.6: Conceptual schema of the *PoliticalParty* class in DBpedia.

data values. We have shown how to automatically extract schema and represent it as a model from a data source using a user-specified threshold. The inferred model takes into account the data and user quality expectations. The result is a conceptual schema enriched by both completeness values as a relevancy indicator on the elements of the models, and existence constraints that inform about how often these elements co-exist or co-appear in the real data.

The elements composing the model (classes, relationships, and properties) are obtained by applying a mining algorithm with an underlying assumption stating that the more frequent a schema is, the more relevant it is. The user can decide on the desired completeness, the parts of the data for which the model will be inferred, and the possibility to focus on a different class through an iterative process. Currently, our demo supports only the DBpedia and Wikidata knowledge graphs.

We have provided several use cases to demonstrate the usefulness of such a tool. We believe that it can help in the discovery of a new knowledge graph and its internal structure.

Therefore, it can help in the adoption of Linked Data knowledge graphs.

Our analysis revealed some interesting characteristics allowing the characterization of the sources and the behavior of the community that maintains each of the data sources. The results show the rich opportunities of analysis offered by our approach and underlying outputs.

Chapter 7

Conclusion and perspectives

The main objective of this thesis work is to bring to the Semantic Web community a way to manage contextual identity. Indeed, we stressed the importance of considering the identity as a relative relationship, i.e., to consider the context. This journey leads us to explore several secondary research questions. Hence, in this chapter, we summarize our different contributions before concluding in several directions for our future work.

7.1 Thesis summary

As we just explained, several trails had to be explored to provide a good solution to our problem with contextual identity.

Large scale study One of the main elements of the Semantic Web field is, of course, semantics. As we needed some semantic features of OWL 2, we discovered their absence in the knowledge graphs we used. We wondered whether this absence was general or punctual. This question led us to conduct a large-scale study on the presence of RDFS and OWL 2 semantics in the Web of Data. We found out that many OWL 2 features are almost not used, and that only some features are extensively used. We also noticed that there is a wide variety of knowledge graph in terms of modeling since some heavily used semantic constructs, and the majority relies only on the most simple features of RDFS. We also discovered that several specific domains use more semantics than others. All programs and data are freely available online to promote reproducibility and reuse of our work.

Ontology The large-scale study leads us to propose an ontology, called OntoSemStats, to capture semantics define and used in a given knowledge graph. Indeed, it is time-consuming to discover manually, by writing SPARQL queries and dealing with a timeout of the SPARQL endpoint, which are semantics features proposed by a knowledge graph. Hence, the instance of our ontology OntoSemStats brings all this information to the user. Not only does it make its work more comfortable, but it can also satisfy data publishers by promoting the reuse of their data and improving community adoption of knowledge graphs. Moreover, we provide several open-source tools to instantiate the ontology for a given SPARQL endpoint. In the present work, the ontology allows us to immediately decide whether it is worth trying to use semantics in our approaches.

Instance matching In this contribution, we aimed at testing the hypothesis that not all properties contribute to the same degree to discover identical entities. The approach finds if two entities are the same or not and is based on two phases: the first is to find logical evidence that the two entities are the same when possible, i.e., if our ontology informs us that there are the necessary semantic features to perform the logical tests and that we can find logical evidence based on those features and data. Hence, if no logical evidence can or has been found, then we compute for each property of the two entities several scores that represent the importance or the predominance of a property. Our approach performs as well as the OAEI 2017 winner, hence it proves its usefulness for identity. Moreover, the code is freely available.

Property propagation Finally, we proposed an approach mainly based on sentence embedding that computes propagable properties for a given identity context. This approach allows rewriting a SPARQL query to deliver more results to the user.

Conceptual schema As explained, one of the purposes of propagating information between contextually identical entities is to increase their completeness. Then, to compute the schema completeness of an entity, one must have a conceptual schema to perform the comparison against it. Because of their nature, knowledge graphs do not have schema like traditional relational databases. Hence, we propose this approach based on property

mining, which allows discovering a conceptual schema for a given class of a knowledge graph.

7.2 Future directions

Several directions are still worth investigating in our work. We want to conduct a survey to take into consideration user feedback to improve the OntoSemStats ontology.

As stated in Chapter 4, some improvements can be made on the instance matching algorithm to better take into consideration the structure of the knowledge graphs. Primarily to address our false positive detection that can do better. Thus, a first step could be to test other ways to aggregate the different weights. In the future, we may also investigate other ways to refine linkset we produced to have fewer false positives results. Besides, more parallelization can improve both scalability and speed performance.

Regarding our conceptual schema generation approach, presented in Chapter 6, we plan to investigate the role of conceptual modeling in an integration context where the universe of discourse is not only one data source but an integrated system upon several Linked Open Data. We plan to make more knowledge graphs available and allow the user to easily compare two conceptual schemas side by side (from two knowledge graphs). We believe that the ability to compare two conceptual schemas of two knowledge graphs side by side can help to choose the one that is best suited for its use.

Regarding the approach of propagation in Chapter 5, some limitations of our approach need further investigation. Firstly, only properties with a textual description could be processed. Using other features to improve the results, like values of properties or semantic features of the property, should be tried. However, capturing ontological information of a property when embedding is still an open problem. Secondly, using only sentence embedding, combined with intuition from Tober's first law, might be naïve in some cases. Therefore, there is a need to challenge our work with a combination of distinct knowledge graphs. For the time being, we only considered in lattices the case where the entity is subject to a triple, and we should also consider cases where it is the value of a triple. Moreover, using SPARQL queries to help the user to select the best-suited identity context

might be an interesting starting point for later work. Finally, to explore SPARQL queries expansion (presented in Section 5.4.3), a prototype should be implemented to allow users to select the proper context according to the ranked list of contexts. Also, using RDF* and SPARQL* [Hartig and Thompson 2014] to represent the context as defined in this chapter should be investigated.

List of publications

International Journals

Pierre-Henri Paris, Nathalie Abadie, and Carmen Brando. Linking spatial named entities to the web of data for geographical analysis of historical texts. *Journal of Map & Geography Libraries*, 13(1):82–110, 2017

International Conference/Workshop articles

Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Interlinking rdf-based datasets: A structure-based approach. In Imre J. Rudas, János Csirik, Carlos Toro, János Botzheim, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES-2019, Budapest, Hungary, 4-6 September 2019*, volume 159 of *Procedia Computer Science*, pages 162–171. Elsevier, 2019a. doi: 10.1016/j.procs.2019.09.171. URL `https://doi.org/10.1016/j.procs.2019.09.171` (**Best Student Paper Award**)

Subhi Issa, Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Revealing the conceptual schemas of RDF datasets. In Paolo Giorgini and Barbara Weber, editors, *Advanced Information Systems Engineering - 31st International Conference, CAiSE 2019, Rome, Italy, June 3-7, 2019, Proceedings*, volume 11483 of *Lecture Notes in Computer Science*, pages 312–327. Springer, 2019. doi: 10.1007/978-3-030-21290-2_20. URL `https://doi.org/10.1007/978-3-030-21290-2_20`

Pierre-Henri Paris, Nathalie Abadie, and Carmen Brando. Georeferencing historical ressources. In *Time Machine conference*, 2018

Subhi Issa, Pierre-Henri Paris, and Fayçal Hamdi. Assessing the completeness evolution of DBpedia: A case study. In Sergio de Cesare and Ulrich Frank, editors, *Advances in Conceptual Modeling - ER 2017 Workshops AHA, MoBiD, MREBA, OntoCom, and QMMQ, Valencia, Spain, November 6-9, 2017, Proceedings*, volume 10651 of *Lecture Notes in Computer Science*, pages 238–247. Springer, 2017. doi: 10.1007/978-3-319-70625-2_22. URL `https://doi.org/10.1007/978-3-319-70625-2_22`

Pierre-Henri Paris, Fayçal Hamdi, Nobal B Niraula, and Samira Si-Said Cherfi. Contextual propagation of properties for knowledge graphs: A sentence embedding based approach. In *International Semantic Web Conference*, 2020d (under review)

International Conference posters

Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. A study about the use of OWL 2 semantics in RDF-based knowledge graphs. In *The Semantic Web: ESWC 2020 Satellite Events - ESWC 2020 Satellite Events*, 2019b (To be published)

Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Ontosemstats: An ontology to express the use of semantics in rdf-based knowledge graphs. In Mária Bieliková, Tommi Mikkonen, and Cesare Pautasso, editors, *Web Engineering - 20th International Conference, ICWE 2020, Helsinki, Finland, June 9-12, 2020, Proceedings*, volume 12128 of *Lecture Notes in Computer Science*, pages 561–565. Springer, 2020a. doi: 10.1007/978-3-030-50578-3_45. URL `https://doi.org/10.1007/978-3-030-50578-3_45`

Pierre-Henri Paris. Assessing the quality of owl: sameas links. In Aldo Gangemi, Anna Lisa Gentile, Andrea Giovanni Nuzzolese, Sebastian Rudolph, Maria Maleshkova, Heiko Paulheim, Jeff Z. Pan, and Mehwish Alam, editors, *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers*, volume 11155 of *Lecture Notes in Computer Science*, pages 304–313. Springer, 2018. doi: 10.1007/978-3-319-98192-5_49. URL `https:`

//doi.org/10.1007/978-3-319-98192-5_49

National Conference articles

Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-said Cherfi. État des lieux de l'utilisation de OWL 2 : Analyse et proposition pour capturer les utilisations de la sémantique OWL 2 dans les graphes de connaissances RDF. *Revue des Nouvelles Technologies de l'Information*, Extraction et Gestion des Connaissances , RNTI-E-36:145–156, 2020c

Subhi Issa, Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Revealing the conceptual schemas of RDF datasets - extended abstract. In *INFORSID 2020*, 2020 (To be published) Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Propagation contextuelle des propriétés pour les graphes de connaissances : une approche fondée sur les plongements de phrases. In *Ingénierie des Connaissances*, 2020b (To be published)

Technical Reports

Tayeb Abderrahmani Ghor, Esha Agrawal, Mehwish Alam, Omar Alqawasmeh, Claudia d'Amato, Amina Annane, Amr Azzam, Andrew Berezovskyi, Russa Biswas, Mathias Bonduel, Quentin Brabant, Cristina-Iulia Bucur, Elena Camossi, Valentina Anita Carriero, Shruthi Chari, David Chaves-Fraga, Fiorela Ciroku, Michael Cochez, Hubert Curien, Vincenzo Cutrona, Rahma Dandan, Danilo Dess, Valerio Di Carlo, Ahmed El Amine Djebri, Marieke van Erp, Faiq Miftakhul Falakh, Alba Fernndez Izquierdo, Giuseppe Futia, Aldo Gangemi, Simone Gasperoni, Arnaud Grall, Lars Heling, Pierre-Henri Paris, Noura Herradi, Subhi Issa, Samaneh Jozashoori, Nyoman Juniarta, Lucie-Aimée Kaffee, Ilkcan Keles, Prashant Khare, Viktor Kovtun, Valentina Leone, Siying Li, Sven Lieber, Pasquale Lisena, Tatiana Makhalova, Ludovica Marinucci, Thomas Minier, Benjamin Moreau, Alberto Moya Loustaunau, Durgesh Nandini, Sylwia Ozdowska, Amanda Pacini de Moura, Swati Padhee, Guillermo Palma, Pedro Del Pozo Jimnez, Valentina Presutti, Roberto Reda, Ettore Rizza, Henry Rosales-mndez, Sebastian Rudolph, Harald Sack, Luca Sciullo, Humasak Simanjuntak, Carlo Stomeo, Thiviyan Thanapalasingam, Tabea Tietz, Dalia Varanka, Maria-Esther Vidal, Michael Wolowyk, and Maximilian Zocholl. Linked open data validity - A technical report from ISWS 2018. *CoRR*, abs/1903.12554, 2019.

URL http://arxiv.org/abs/1903.12554

Bibliography

Manel Achichi, Zohra Bellahsene, and Konstantin Todorov. A survey on web data linking. *Ingénierie des Systèmes d'Information*, 21(5-6):11–29, 2016. doi: 10.3166/isi.21.5-6.11-29. URL `https://doi.org/10.3166/isi.21.5-6.11-29`.

Manel Achichi, Zohra Bellahsene, and Konstantin Todorov. Legato results for OAEI 2017. In Pavel Shvaiko, Jérôme Euzenat, Ernesto Jiménez-Ruiz, Michelle Cheatham, and Oktie Hassanzadeh, editors, *Proceedings of the 12th International Workshop on Ontology Matching co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017*, volume 2032 of *CEUR Workshop Proceedings*, pages 146–152. CEUR-WS.org, 2017. URL `http://ceur-ws.org/Vol-2032/oaei17_paper6.pdf`.

Keith Alexander, Richard Cyganiak, Michael Hausenblas, and Jun Zhao. Describing linked datasets. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Kingsley Idehen, editors, *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009.*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009. URL `http://ceur-ws.org/Vol-538/ldow2009_paper20.pdf`.

Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, 2001. doi: 10.1023/A:1013882326814. URL `https://doi.org/10.1023/A:1013882326814`.

Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics. In Frank van Harmelen, Vladimir Lifschitz, and Bruce W. Porter, editors, *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 135–179. Else-

vier, 2008. doi: 10.1016/S1574-6526(07)03003-9. URL https://doi.org/10.1016/S1574-6526(07)03003-9.

Carlo Batini and Monica Scannapieco. *Data and Information Quality - Dimensions, Principles and Techniques*. Data-Centric Systems and Applications. Springer, 2016. ISBN 978-3-319-24104-3. doi: 10.1007/978-3-319-24106-7. URL https://doi.org/10.1007/978-3-319-24106-7.

Wouter Beek, Laurens Rietveld, Hamid R. Bazoobandi, Jan Wielemaker, and Stefan Schlobach. LOD laundromat: A uniform way of publishing other people's dirty data. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig A. Knoblock, Denny Vrandecic, Paul T. Groth, Natasha F. Noy, Krzysztof Janowicz, and Carole A. Goble, editors, *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, volume 8796 of *Lecture Notes in Computer Science*, pages 213–228. Springer, 2014. doi: 10.1007/978-3-319-11964-9_14. URL https://doi.org/10.1007/978-3-319-11964-9_14.

Wouter Beek, Stefan Schlobach, and Frank van Harmelen. A contextualised semantics for owl: sameas. In Harald Sack, Eva Blomqvist, Mathieu d'Aquin, Chiara Ghidini, Simone Paolo Ponzetto, and Christoph Lange, editors, *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, volume 9678 of *Lecture Notes in Computer Science*, pages 405–419. Springer, 2016. doi: 10.1007/978-3-319-34129-3_25. URL https://doi.org/10.1007/978-3-319-34129-3_25.

Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

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

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *TACL*, 5:135–146, 2017. URL https://transacl.org/ojs/index.php/tacl/article/view/999.

Vannevar Bush. As we may think. *The Atlantic Monthly*, 176(1):101–108, 1945. URL http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018. URL http://arxiv.org/abs/1803.11175.

Klitos Christodoulou, Norman W. Paton, and Alvaro A. A. Fernandes. Structure inference for linked data sources using clustering. *Trans. Large-Scale Data- and Knowledge-Centered Systems*, 19:1–25, 2015. doi: 10.1007/978-3-662-46562-2_1. URL https://doi.org/10.1007/978-3-662-46562-2_1.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 670–680. Association for Computational Linguistics, 2017.

Mathieu d'Aquin, Claudio Baldassarre, Laurian Gridinoc, Sofia Angeletou, Marta Sabou, and Enrico Motta. Characterizing knowledge on the semantic web with watson. In Raul Garcia-Castro, Denny Vrandecic, Asunción Gómez-Pérez, York Sure, and Zhisheng Huang, editors, *Proceedings of the 5th International Workshop on Evaluation of Ontologies and Ontology-based Tools, EON2007, Co-located with the ISWC2007, Busan, Korea, November 11th, 2007*, volume 329 of *CEUR Workshop Proceedings*, pages 1–10. CEUR-WS.org, 2007. URL http://ceur-ws.org/Vol-329/paper01.pdf.

Fariz Darari, Werner Nutt, Giuseppe Pirrò, and Simon Razniewski. Completeness statements about RDF data sources and their use for query answering. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul T. Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha F. Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, volume 8218 of *Lecture Notes in Computer Science*, pages 66–83. Springer, 2013. doi: 10.1007/978-3-642-41335-3_5. URL https://doi.org/10.1007/978-3-642-41335-3_5.

Gerard de Melo. Not quite the same: Identity constraints for the web of linked data. In Marie desJardins and Michael L. Littman, editors, *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. AAAI Press, 2013. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6491.

Li Ding, Joshua Shinavier, Tim Finin, and Deborah L McGuinness. owl: sameas and linked data: An empirical study. 2010.

Nick Drummond and Rob Shearer. The open world assumption. In *eSI Workshop: The Closed World of Databases meets the Open World of the Semantic Web*, volume 15, 2006.

Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. In Michael Martin, Martí Cuquet, and Erwin Folmer, editors, *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*, volume 1695 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. URL http://ceur-ws.org/Vol-1695/paper4.pdf.

David W. Embley and Stephen W. Liddle. Big data - conceptual modeling to the rescue. In *Conceptual Modeling - 32th International Conference, ER 2013, Hong-Kong, China, November 11-13, 2013. Proceedings*, pages 1–8, 2013. doi: 10.1007/978-3-642-41924-9_1. URL http://dx.doi.org/10.1007/978-3-642-41924-9_1.

Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. Linked data quality of dbpedia, freebase, opencyc, wikidata, and YAGO. *Semantic Web*, 9(1):77–129, 2018. doi: 10.3233/SW-170275. URL https://doi.org/10.3233/SW-170275.

Javier D. Fernández, Miguel A. Martínez-Prieto, Claudio Gutiérrez, Axel Polleres, and Mario Arias. Binary RDF representation for publication and exchange (HDT). *J. Web Semant.*, 19:22–41, 2013. doi: 10.1016/j.websem.2013.01.002. URL https://doi.org/10.1016/j.websem.2013.01.002.

Alfio Ferrara, Andriy Nikolov, and François Scharffe. Data linking for the semantic web. *Int. J. Semantic Web Inf. Syst.*, 7(3):46–76, 2011. doi: 10.4018/jswis.2011070103. URL https://doi.org/10.4018/jswis.2011070103.

Tayeb Abderrahmani Ghor, Esha Agrawal, Mehwish Alam, Omar Alqawasmeh, Claudia d'Amato, Amina Annane, Amr Azzam, Andrew Berezovskyi, Russa Biswas, Mathias Bonduel, Quentin Brabant, Cristina-Iulia Bucur, Elena Camossi, Valentina Anita Carriero, Shruthi Chari, David Chaves-Fraga, Fiorela Ciroku, Michael Cochez, Hubert Curien, Vincenzo Cutrona, Rahma Dandan, Danilo Dess, Valerio Di Carlo, Ahmed El Amine Djebri, Marieke van Erp, Faiq Miftakhul Falakh, Alba Fernndez Izquierdo, Giuseppe Futia, Aldo Gangemi, Simone Gasperoni, Arnaud Grall, Lars Heling, Pierre-Henri Paris, Noura Herradi, Subhi Issa, Samaneh Jozashoori, Nyoman Juniarta, Lucie-Aimée Kaffee, Ilkcan Keles, Prashant Khare, Viktor Kovtun, Valentina Leone, Siying Li, Sven Lieber, Pasquale Lisena, Tatiana Makhalova, Ludovica Marinucci, Thomas Minier, Benjamin Moreau, Alberto Moya Loustaunau, Durgesh Nandini, Sylwia Ozdowska, Amanda Pacini de Moura, Swati Padhee, Guillermo Palma, Pedro Del Pozo Jimnez, Valentina Presutti, Roberto Reda, Ettore Rizza, Henry Rosales-mndez, Sebastian Rudolph, Harald Sack, Luca Sciullo, Humasak Simanjuntak, Carlo Stomeo, Thiviyan Thanapalasingam, Tabea Tietz, Dalia Varanka, Maria-Esther Vidal, Michael Wolowyk, and Maximilian Zocholl. Linked open data validity - A technical report from ISWS 2018. *CoRR*, abs/1903.12554, 2019. URL http://arxiv.org/abs/1903.12554.

Birte Glimm, Aidan Hogan, Markus Krötzsch, and Axel Polleres. OWL: yet to arrive on the web of data? In Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, editors, *WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012*, volume 937 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012. URL http://ceur-ws.org/Vol-937/ldow2012-paper-16.pdf.

Karam Gouda and Mohammed Javeed Zaki. Efficiently mining maximal frequent itemsets. In Nick Cercone, Tsau Young Lin, and Xindong Wu, editors, *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San*

Jose, California, USA, pages 163–170. IEEE Computer Society, 2001. doi: 10.1109/ICDM.2001.989514. URL https://doi.org/10.1109/ICDM.2001.989514.

Gösta Grahne and Jianfei Zhu. Efficiently using prefix-trees in mining frequent itemsets. In Bart Goethals and Mohammed Javeed Zaki, editors, *FIMI '03, Frequent Itemset Mining Implementations, Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

Nicola Guarino and Christopher A. Welty. Evaluating ontological decisions with ontoclean. *Commun. ACM*, 45(2):61–65, 2002. doi: 10.1145/503124.503150. URL https://doi.org/10.1145/503124.503150.

Christophe Guéret, Paul T. Groth, Claus Stadler, and Jens Lehmann. Assessing linked data mappings using network measures. In Elena Simperl, Philipp Cimiano, Axel Polleres, Óscar Corcho, and Valentina Presutti, editors, *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, volume 7295 of *Lecture Notes in Computer Science*, pages 87–102. Springer, 2012. doi: 10.1007/978-3-642-30284-8_13. URL https://doi.org/10.1007/978-3-642-30284-8_13.

Harry Halpin, Patrick J. Hayes, James P. McCusker, Deborah L. McGuinness, and Henry S. Thompson. When owl: sameas isn't the same: An analysis of identity in linked data. In Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, volume 6496 of *Lecture Notes in Computer Science*, pages 305–320. Springer, 2010. doi: 10.1007/978-3-642-17746-0_20. URL https://doi.org/10.1007/978-3-642-17746-0_20.

Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18,*

2000, Dallas, Texas, USA, pages 1–12. ACM, 2000. doi: 10.1145/342009.335372. URL https://doi.org/10.1145/342009.335372.

Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8 (1):53–87, 2004. doi: 10.1023/B:DAMI.0000005258.31418.83. URL https://doi.org/10.1023/B:DAMI.0000005258.31418.83.

Olaf Hartig and Bryan Thompson. Foundations of an alternative approach to reification in RDF. *CoRR*, abs/1406.3399, 2014. URL http://arxiv.org/abs/1406.3399.

Pascal Hitzler and Frank van Harmelen. A reasonable semantic web. *Semantic Web*, 1 (1-2):39–44, 2010.

Aidan Hogan, Andreas Harth, Alexandre Passant, Stefan Decker, and Axel Polleres. Weaving the pedantic web. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, editors, *Proceedings of the WWW2010 Workshop on Linked Data on the Web, LDOW 2010, Raleigh, USA, April 27, 2010*, volume 628 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible SROIQ. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006*, pages 57–67. AAAI Press, 2006. URL http://www.aaai.org/Library/KR/2006/kr06-009.php.

Al Koudous Idrissou, Rinke Hoekstra, Frank van Harmelen, Ali Khalili, and Peter van den Besselaar. Is my: sameas the same as your: sameas?: Lenticular lenses for context-specific identity. In Óscar Corcho, Krzysztof Janowicz, Giuseppe Rizzo, Ilaria Tiddi, and Daniel Garijo, editors, *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*, pages 23:1–23:8. ACM, 2017. doi: 10.1145/3148011.3148029. URL https://doi.org/10.1145/3148011.3148029.

Al Koudous Idrissou, Frank van Harmelen, and Peter van den Besselaar. Network metrics for assessing the quality of entity resolution between multiple datasets. In Catherine Faron-

Zucker, Chiara Ghidini, Amedeo Napoli, and Yannick Toussaint, editors, *Knowledge Engineering and Knowledge Management - 21st International Conference, EKAW 2018, Nancy, France, November 12-16, 2018, Proceedings*, volume 11313 of *Lecture Notes in Computer Science*, pages 147–162. Springer, 2018. doi: 10.1007/978-3-030-03667-6_10. URL https://doi.org/10.1007/978-3-030-03667-6_10.

Subhi Issa, Pierre-Henri Paris, and Fayçal Hamdi. Assessing the completeness evolution of DBpedia: A case study. In Sergio de Cesare and Ulrich Frank, editors, *Advances in Conceptual Modeling - ER 2017 Workshops AHA, MoBiD, MREBA, OntoCom, and QMMQ, Valencia, Spain, November 6-9, 2017, Proceedings*, volume 10651 of *Lecture Notes in Computer Science*, pages 238–247. Springer, 2017. doi: 10.1007/978-3-319-70625-2_22. URL https://doi.org/10.1007/978-3-319-70625-2_22.

Subhi Issa, Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Revealing the conceptual schemas of RDF datasets. In Paolo Giorgini and Barbara Weber, editors, *Advanced Information Systems Engineering - 31st International Conference, CAiSE 2019, Rome, Italy, June 3-7, 2019, Proceedings*, volume 11483 of *Lecture Notes in Computer Science*, pages 312–327. Springer, 2019. doi: 10.1007/978-3-030-21290-2_20. URL https://doi.org/10.1007/978-3-030-21290-2_20.

Subhi Issa, Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Revealing the conceptual schemas of RDF datasets - extended abstract. In *INFORSID 2020*, 2020.

Paul Jaccard. Nouvelles recherches sur la distribution florale. *Bull. Soc. Vaud. Sci. Nat.*, 44:223–270, 1908.

Prateek Jain, Pascal Hitzler, Peter Z. Yeh, Kunal Verma, and Amit P. Sheth. Linked data is merely more data. In *Linked Data Meets Artificial Intelligence, Papers from the 2010 AAAI Spring Symposium, Technical Report SS-10-07, Stanford, California, USA, March 22-24, 2010*. AAAI, 2010. URL http://www.aaai.org/ocs/index.php/SSS/SSS10/paper/view/1130.

Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham

Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, volume 7031 of *Lecture Notes in Computer Science*, pages 273–288. Springer, 2011. doi: 10.1007/978-3-642-25073-6_18. URL https://doi.org/10.1007/978-3-642-25073-6_18.

Roberto J. Bayardo Jr. Efficiently mining long patterns from databases. In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA.*, pages 85–93. ACM Press, 1998. doi: 10.1145/276304.276313.

Kenza Kellou-Menouer and Zoubida Kedad. Schema discovery in RDF data sources. In Paul Johannesson, Mong-Li Lee, Stephen W. Liddle, Andreas L. Opdahl, and Oscar Pastor López, editors, *Conceptual Modeling - 34th International Conference, ER 2015, Stockholm, Sweden, October 19-22, 2015, Proceedings*, volume 9381 of *Lecture Notes in Computer Science*, pages 481–495. Springer, 2015. doi: 10.1007/978-3-319-25264-3_36. URL https://doi.org/10.1007/978-3-319-25264-3_36.

Abderrahmane Khiat and Maximilian Mackeprang. I-match and ontoidea results for OAEI 2017. In Pavel Shvaiko, Jérôme Euzenat, Ernesto Jiménez-Ruiz, Michelle Cheatham, and Oktie Hassanzadeh, editors, *Proceedings of the 12th International Workshop on Ontology Matching co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017*, volume 2032 of *CEUR Workshop Proceedings*, pages 135–137. CEUR-WS.org, 2017. URL http://ceur-ws.org/Vol-2032/oaei17_paper4.pdf.

Graham Klyne and Jeremy J Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006.

Justin J. Levandoski and Mohamed F. Mokbel. RDF data-centric storage. In *IEEE International Conference on Web Services, ICWS 2009, Los Angeles, CA, USA, 6-10 July 2009*, pages 911–918. IEEE Computer Society, 2009. doi: 10.1109/ICWS.2009.49. URL https://doi.org/10.1109/ICWS.2009.49.

Roman Lukyanenko and Jeffrey Parsons. Principles for modeling user-generated content. In Paul Johannesson, Mong-Li Lee, Stephen W. Liddle, Andreas L. Opdahl, and Oscar Pastor López, editors, *Conceptual Modeling - 34th International Conference, ER 2015, Stockholm, Sweden, October 19-22, 2015, Proceedings*, volume 9381 of *Lecture Notes in Computer Science*, pages 432–440. Springer, 2015. doi: 10.1007/978-3-319-25264-3_32. URL https://doi.org/10.1007/978-3-319-25264-3_32.

Roman Lukyanenko, Jeffrey Parsons, and Binny M. Samuel. Representing instances: the case for reengineering conceptual modelling grammars. *Eur. J. Inf. Syst.*, 28(1): 68–90, 2019. doi: 10.1080/0960085X.2018.1488567. URL https://doi.org/10.1080/0960085X.2018.1488567.

Miguel A. Martínez-Prieto, Mario Arias Gallego, and Javier D. Fernández. Exchange and consumption of huge RDF data. In Elena Simperl, Philipp Cimiano, Axel Polleres, Óscar Corcho, and Valentina Presutti, editors, *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, volume 7295 of *Lecture Notes in Computer Science*, pages 437–452. Springer, 2012. doi: 10.1007/978-3-642-30284-8_36. URL https://doi.org/10.1007/978-3-642-30284-8_36.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL http://arxiv.org/abs/1301.3781.

Theodor Holm Nelson. Complex information processing: a file structure for the complex, the changing and the indeterminate. In *ACM '65*, 1965.

Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. A survey of current link discovery frameworks. *Semantic Web*, 8(3):419–436, 2017. doi: 10.3233/SW-150210. URL https://doi.org/10.3233/SW-150210.

Andriy Nikolov, Victoria S. Uren, Enrico Motta, and Anne N. De Roeck. Integration of

semantically annotated data by the knofuss architecture. In Aldo Gangemi and Jérôme Euzenat, editors, *Knowledge Engineering: Practice and Patterns, 16th International Conference, EKAW 2008, Acitrezza, Italy, September 29 - October 2, 2008. Proceedings*, volume 5268 of *Lecture Notes in Computer Science*, pages 265–274. Springer, 2008. doi: 10.1007/978-3-540-87696-0_24. URL https://doi.org/10.1007/978-3-540-87696-0_24.

Natalya Fridman Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM*, 62(8):36–43, 2019. doi: 10.1145/3331166. URL https://doi.org/10.1145/3331166.

Antoni Olivé. *Conceptual modeling of information systems*. Springer, 2007. doi: 10.1007/978-3-540-39390-0. URL https://doi.org/10.1007/978-3-540-39390-0.

Laura Papaleo, Nathalie Pernelle, Fatiha Saïs, and Cyril Dumont. Logical detection of invalid sameas statements in RDF data. In Krzysztof Janowicz, Stefan Schlobach, Patrick Lambrix, and Eero Hyvönen, editors, *Knowledge Engineering and Knowledge Management - 19th International Conference, EKAW 2014, Linköping, Sweden, November 24-28, 2014. Proceedings*, volume 8876 of *Lecture Notes in Computer Science*, pages 373–384. Springer, 2014. doi: 10.1007/978-3-319-13704-9_29. URL https://doi.org/10.1007/978-3-319-13704-9_29.

Pierre-Henri Paris. Assessing the quality of owl: sameas links. In Aldo Gangemi, Anna Lisa Gentile, Andrea Giovanni Nuzzolese, Sebastian Rudolph, Maria Maleshkova, Heiko Paulheim, Jeff Z. Pan, and Mehwish Alam, editors, *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers*, volume 11155 of *Lecture Notes in Computer Science*, pages 304–313. Springer, 2018. doi: 10.1007/978-3-319-98192-5_49. URL https://doi.org/10.1007/978-3-319-98192-5_49.

Pierre-Henri Paris, Nathalie Abadie, and Carmen Brando. Linking spatial named entities to the web of data for geographical analysis of historical texts. *Journal of Map & Geography Libraries*, 13(1):82–110, 2017.

Pierre-Henri Paris, Nathalie Abadie, and Carmen Brando. Georeferencing historical ressources. In *Time Machine conference*, 2018.

Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Interlinking rdf-based datasets: A structure-based approach. In Imre J. Rudas, János Csirik, Carlos Toro, János Botzheim, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES-2019, Budapest, Hungary, 4-6 September 2019*, volume 159 of *Procedia Computer Science*, pages 162–171. Elsevier, 2019a. doi: 10.1016/j.procs.2019.09.171. URL `https://doi.org/10.1016/j.procs.2019.09.171`.

Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. A study about the use of OWL 2 semantics in RDF-based knowledge graphs. In *The Semantic Web: ESWC 2020 Satellite Events - ESWC 2020 Satellite Events*, 2019b.

Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Ontosemstats: An ontology to express the use of semantics in rdf-based knowledge graphs. In Mária Bieliková, Tommi Mikkonen, and Cesare Pautasso, editors, *Web Engineering - 20th International Conference, ICWE 2020, Helsinki, Finland, June 9-12, 2020, Proceedings*, volume 12128 of *Lecture Notes in Computer Science*, pages 561–565. Springer, 2020a. doi: 10.1007/978-3-030-50578-3_45. URL `https://doi.org/10.1007/978-3-030-50578-3_45`.

Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-Said Cherfi. Propagation contextuelle des propriétés pour les graphes de connaissances : une approche fondée sur les plongements de phrases. In *Ingénierie des Connaissances*, 2020b.

Pierre-Henri Paris, Fayçal Hamdi, and Samira Si-said Cherfi. État des lieux de l'utilisation de OWL 2 : Analyse et proposition pour capturer les utilisations de la sémantique OWL 2 dans les graphes de connaissances RDF. *Revue des Nouvelles Technologies de l'Information*, Extraction et Gestion des Connaissances , RNTI-E-36:145–156, 2020c.

Pierre-Henri Paris, Fayçal Hamdi, Nobal B Niraula, and Samira Si-Said Cherfi. Contextual propagation of properties for knowledge graphs: A sentence embedding based approach. In *International Semantic Web Conference*, 2020d.

Heiko Paulheim. Identifying wrong links between datasets by multi-dimensional outlier detection. In Patrick Lambrix, Guilin Qi, Matthew Horridge, and Bijan Parsia, editors, *Proceedings of the Third International Workshop on Debugging Ontologies and Ontology Mappings, WoDOOM 2014, co-located with 11th Extended Semantic Web Conference (ESWC 2014), Anissaras/Hersonissou, Greece, May 26, 2014*, volume 1162 of *CEUR Workshop Proceedings*, pages 27–38. CEUR-WS.org, 2014. URL `http://ceur-ws.org/Vol-1162/paper3.pdf`.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014. doi: 10.3115/v1/d14-1162. URL `https://doi.org/10.3115/v1/d14-1162`.

Minh-Duc Pham, Linnea Passing, Orri Erling, and Peter A. Boncz. Deriving an emergent relational schema from RDF data. In Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi, editors, *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 864–874. ACM, 2015. doi: 10.1145/2736277.2741121. URL `http://doi.acm.org/10.1145/2736277.2741121`.

Axel Polleres, Aidan Hogan, Andreas Harth, and Stefan Decker. Can we ever catch up with the web? *Semantic Web*, 1(1-2):45–52, 2010. doi: 10.3233/SW-2010-0016. URL `https://doi.org/10.3233/SW-2010-0016`.

Joe Raad, Nathalie Pernelle, and Fatiha Saïs. Detection of contextual identity links in a knowledge base. In Óscar Corcho, Krzysztof Janowicz, Giuseppe Rizzo, Ilaria Tiddi, and Daniel Garijo, editors, *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*, pages 8:1–8:8. ACM, 2017. doi: 10.1145/3148011.3148032. URL `https://doi.org/10.1145/3148011.3148032`.

Joe Raad, Wouter Beek, Frank van Harmelen, Nathalie Pernelle, and Fatiha Saïs. Detecting erroneous identity links on the web using network metrics. In Denny Vrandecic, Kalina

Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors, *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, volume 11136 of *Lecture Notes in Computer Science*, pages 391–407. Springer, 2018. doi: 10.1007/978-3-030-00671-6_23. URL `https://doi.org/10.1007/978-3-030-00671-6_23`.

Petar Ristoski and Heiko Paulheim. Rdf2vec: RDF graph embeddings for data mining. In Paul T. Groth, Elena Simperl, Alasdair J. G. Gray, Marta Sabou, Markus Krötzsch, Freddy Lécué, Fabian Flöck, and Yolanda Gil, editors, *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, volume 9981 of *Lecture Notes in Computer Science*, pages 498–514, 2016. doi: 10.1007/978-3-319-46523-4_30. URL `https://doi.org/10.1007/978-3-319-46523-4_30`.

Colette Rolland and Naveen Prakash. From conceptual modelling to requirements engineering. *Ann. Software Eng.*, 10:151–176, 2000. doi: 10.1023/A:1018939700514. URL `https://doi.org/10.1023/A:1018939700514`.

Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24 (4):35–43, 2001.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J. Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *CoRR*, abs/1804.00079, 2018.

Ruben Taelman, Joachim Van Herwegen, Miel Vander Sande, and Ruben Verborgh. Comunica: A modular SPARQL query engine for the web. In *International Semantic Web Conference (2)*, volume 11137 of *Lecture Notes in Computer Science*, pages 239–255. Springer, 2018.

Waldo R Tobler. A computer movie simulating urban growth in the detroit region. *Economic geography*, 46(sup1):234–240, 1970.

Gerald Töpper, Magnus Knuth, and Harald Sack. Dbpedia ontology enrichment for inconsistency detection. In Valentina Presutti and Helena Sofia Pinto, editors, *I-SEMANTICS 2012 - 8th International Conference on Semantic Systems, I-SEMANTICS '12, Graz, Austria, September 5-7, 2012*, pages 33–40. ACM, 2012. doi: 10.1145/2362499.2362505. URL `https://doi.org/10.1145/2362499.2362505`.

André Valdestilhas, Tommaso Soru, and Axel-Cyrille Ngonga Ngomo. CEDAL: time-efficient detection of erroneous links in large-scale link repositories. In Amit P. Sheth, Axel Ngonga, Yin Wang, Elizabeth Chang, Dominik Slezak, Bogdan Franczyk, Rainer Alt, Xiaohui Tao, and Rainer Unland, editors, *Proceedings of the International Conference on Web Intelligence, Leipzig, Germany, August 23-26, 2017*, pages 106–113. ACM, 2017. doi: 10.1145/3106426.3106497. URL `https://doi.org/10.1145/3106426.3106497`.

Johanna Völker and Mathias Niepert. Statistical schema induction. In Grigoris Antoniou, Marko Grobelnik, Elena Paslaru Bontas Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Z. Pan, editors, *The Semantic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings, Part I*, volume 6643 of *Lecture Notes in Computer Science*, pages 124–138. Springer, 2011. doi: 10.1007/978-3-642-21034-1_9. URL `https://doi.org/10.1007/978-3-642-21034-1_9`.

Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk - A link discovery framework for the web of data. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Kingsley Idehen, editors, *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009. URL `http://ceur-ws.org/Vol-538/ldow2009_paper13.pdf`.

Yan Wang, Xiaoyong Du, Jiaheng Lu, and Xiaofang Wang. Flextable: Using a dynamic relation model to store RDF data. In Hiroyuki Kitagawa, Yoshiharu Ishikawa, Qing Li, and Chiemi Watanabe, editors, *Database Systems for Advanced Applications, 15th International Conference, DASFAA 2010, Tsukuba, Japan, April 1-4, 2010, Proceedings, Part I*, volume 5981 of *Lecture Notes in Computer Science*, pages 580–594.

Springer, 2010. doi: 10.1007/978-3-642-12026-8_44. URL https://doi.org/10.1007/978-3-642-12026-8_44.

Stuart Weibel, John A. Kunze, Carl Lagoze, and Misha Wolf. Dublin core metadata for resource discovery. *RFC*, 2413:1–8, 1998. doi: 10.17487/RFC2413. URL https://doi.org/10.17487/RFC2413.

Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, Sören Auer, and Pascal Hitzler. Quality assessment methodologies for linked open data. *Submitted to Semantic Web Journal*, 2013.

Appendix A

Annex

```
<?xml version="1.0" ?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     ontologyIRI=
          "http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl">
     <Prefix name=""
          IRI="http://cedric.cnam.fr/isid/ontologies/OntoSemStats.owl#" />
     <Prefix name="owl"
          IRI="http://www.w3.org/2002/07/owl#" />
     <Prefix name="rdf"
          IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
     <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace" />
     <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#" />
     <Prefix name="obda" IRI="https://w3id.org/obda/vocabulary#" />
     <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#" />
     <Prefix name="void" IRI="http://rdfs.org/ns/void#" />
     <Annotation>
          <AnnotationProperty
               IRI="http://creativecommons.org/ns#license" />
          <IRI>http://creativecommons.org/licenses/by/2.0/</IRI>
     </Annotation>
     <Annotation>
          <AnnotationProperty
               IRI="http://purl.org/dc/elements/1.1/creator" />
          <IRI>https://github.com/PHParis</IRI>
     </Annotation>
     <Annotation>
          <AnnotationProperty
               IRI="http://purl.org/dc/terms/contributor" />
          <IRI>http://cedric.cnam.fr/~hamdif/</IRI>
     </Annotation>
     <Annotation>
          <AnnotationProperty
               IRI="http://purl.org/dc/terms/contributor" />
          <IRI>http://cedric.cnam.fr/~sisaid/</IRI>
```

```
</Annotation>
<Annotation>
    <AnnotationProperty
        abbreviatedIRI="owl:versionInfo" />
    <Literal
        datatypeIRI="http://www.w3.org/2001/XMLSchema#decimal">
        1.0
    </Literal>
</Annotation>
<Declaration>
    <Class IRI="#Assertion" />
</Declaration>
<Declaration>
    <Class IRI="#Axiom" />
</Declaration>
<Declaration>
    <Class IRI="#BooleanConnective" />
</Declaration>
<Declaration>
    <Class IRI="#Class" />
</Declaration>
<Declaration>
    <Class IRI="#ClassExpression" />
</Declaration>
<Declaration>
    <Class IRI="#DataRange" />
</Declaration>
<Declaration>
    <Class IRI="#IndividualEnumeration" />
</Declaration>
<Declaration>
    <Class IRI="#OwlAllDifferent" />
</Declaration>
<Declaration>
    <Class IRI="#OwlAllDisjointClasses" />
</Declaration>
<Declaration>
    <Class IRI="#OwlAllDisjointProperties" />
</Declaration>
<Declaration>
    <Class IRI="#OwlAllValuesFrom" />
</Declaration>
<Declaration>
    <Class IRI="#OwlAsymmetricProperty" />
</Declaration>
<Declaration>
    <Class IRI="#OwlCardinality" />
</Declaration>
<Declaration>
    <Class IRI="#OwlComplementOf" />
</Declaration>
<Declaration>
    <Class IRI="#OwlDatatypeComplementOf" />
</Declaration>
<Declaration>
    <Class IRI="#OwlDifferentFrom" />
```

```
</Declaration>
<Declaration>
    <Class IRI="#OwlDisjointUnionOf" />
</Declaration>
<Declaration>
    <Class IRI="#OwlDisjointWith" />
</Declaration>
<Declaration>
    <Class IRI="#OwlEquivalentClass" />
</Declaration>
<Declaration>
    <Class IRI="#OwlEquivalentProperty" />
</Declaration>
<Declaration>
    <Class IRI="#OwlFunctionalProperty" />
</Declaration>
<Declaration>
    <Class IRI="#OwlHasSelf" />
</Declaration>
<Declaration>
    <Class IRI="#OwlHasValue" />
</Declaration>
<Declaration>
    <Class IRI="#OwlIntersectionOf" />
</Declaration>
<Declaration>
    <Class IRI="#OwlInverseFunctionalProperty" />
</Declaration>
<Declaration>
    <Class IRI="#OwlInverseOf" />
</Declaration>
<Declaration>
    <Class IRI="#OwlIrreflexiveProperty" />
</Declaration>
<Declaration>
    <Class IRI="#OwlMaxCardinality" />
</Declaration>
<Declaration>
    <Class IRI="#OwlMaxQualifiedCardinality" />
</Declaration>
<Declaration>
    <Class IRI="#OwlMinCardinality" />
</Declaration>
<Declaration>
    <Class IRI="#OwlMinQualifiedCardinality" />
</Declaration>
<Declaration>
    <Class IRI="#OwlNegativePropertyAssertion" />
</Declaration>
<Declaration>
    <Class IRI="#OwlOneOf" />
</Declaration>
<Declaration>
    <Class IRI="#OwlPropertyChainAxiom" />
</Declaration>
<Declaration>
```

```
    <Class IRI="#OwlPropertyDisjointWith" />
</Declaration>
<Declaration>
    <Class IRI="#OwlQualifiedCardinality" />
</Declaration>
<Declaration>
    <Class IRI="#OwlReflexiveProperty" />
</Declaration>
<Declaration>
    <Class IRI="#OwlSameAs" />
</Declaration>
<Declaration>
    <Class IRI="#OwlSomeValuesFrom" />
</Declaration>
<Declaration>
    <Class IRI="#OwlSymmetricProperty" />
</Declaration>
<Declaration>
    <Class IRI="#OwlTransitiveProperty" />
</Declaration>
<Declaration>
    <Class IRI="#OwlUnionOf" />
</Declaration>
<Declaration>
    <Class IRI="#OwlWithRestrictions" />
</Declaration>
<Declaration>
    <Class IRI="#PropertyAxiom" />
</Declaration>
<Declaration>
    <Class IRI="#PropertyRelation" />
</Declaration>
<Declaration>
    <Class IRI="#PropertyRestriction" />
</Declaration>
<Declaration>
    <Class IRI="#PropertySignature" />
</Declaration>
<Declaration>
    <Class IRI="#PropertyType" />
</Declaration>
<Declaration>
    <Class IRI="#RdfType" />
</Declaration>
<Declaration>
    <Class IRI="#RdfsDomain" />
</Declaration>
<Declaration>
    <Class IRI="#RdfsRange" />
</Declaration>
<Declaration>
    <Class IRI="#RdfsSubClassOf" />
</Declaration>
<Declaration>
    <Class IRI="#RdfsSubProperty" />
</Declaration>
```

```
<Declaration>
    <Class IRI="#SemanticFeature" />
</Declaration>
<Declaration>
    <Class IRI="#Stat" />
</Declaration>
<Declaration>
    <Class abbreviatedIRI="void:Dataset" />
</Declaration>
<Declaration>
    <ObjectProperty IRI="#hasSemanticFeature" />
</Declaration>
<Declaration>
    <ObjectProperty IRI="#hasStat" />
</Declaration>
<Declaration>
    <DataProperty IRI="#definitionCount" />
</Declaration>
<Declaration>
    <DataProperty IRI="#usageCount" />
</Declaration>
<Declaration>
    <AnnotationProperty
        IRI="http://creativecommons.org/ns#license" />
</Declaration>
<Declaration>
    <AnnotationProperty
        IRI="http://purl.org/dc/elements/1.1/creator" />
</Declaration>
<Declaration>
    <AnnotationProperty
        IRI="http://purl.org/dc/terms/contributor" />
</Declaration>
<EquivalentClasses>
    <Class IRI="#Stat" />
    <ObjectExactCardinality cardinality="1">
        <ObjectProperty IRI="#hasSemanticFeature" />
        <Class IRI="#SemanticFeature" />
    </ObjectExactCardinality>
</EquivalentClasses>
<SubClassOf>
    <Class IRI="#Assertion" />
    <Class IRI="#Axiom" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#Axiom" />
    <Class IRI="#SemanticFeature" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#BooleanConnective" />
    <Class IRI="#Class" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#Class" />
    <Class IRI="#SemanticFeature" />
</SubClassOf>
```

```
<SubClassOf>
    <Class IRI="#ClassExpression" />
    <Class IRI="#Axiom" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#DataRange" />
    <Class IRI="#SemanticFeature" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#IndividualEnumeration" />
    <Class IRI="#Class" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlAllDifferent" />
    <Class IRI="#Assertion" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlAllDisjointClasses" />
    <Class IRI="#ClassExpression" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlAllDisjointProperties" />
    <Class IRI="#PropertyRelation" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlAllValuesFrom" />
    <Class IRI="#PropertyRestriction" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlAsymmetricProperty" />
    <Class IRI="#PropertyType" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlCardinality" />
    <Class IRI="#PropertyRestriction" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlComplementOf" />
    <Class IRI="#BooleanConnective" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlDatatypeComplementOf" />
    <Class IRI="#DataRange" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlDifferentFrom" />
    <Class IRI="#Assertion" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlDisjointUnionOf" />
    <Class IRI="#ClassExpression" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlDisjointWith" />
    <Class IRI="#ClassExpression" />
</SubClassOf>
```

```
<SubClassOf>
    <Class IRI="#OwlEquivalentClass" />
    <Class IRI="#ClassExpression" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlEquivalentProperty" />
    <Class IRI="#PropertyRelation" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlFunctionalProperty" />
    <Class IRI="#PropertyType" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlHasSelf" />
    <Class IRI="#PropertyRestriction" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlHasValue" />
    <Class IRI="#PropertyRestriction" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlIntersectionOf" />
    <Class IRI="#BooleanConnective" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlInverseFunctionalProperty" />
    <Class IRI="#PropertyType" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlInverseOf" />
    <Class IRI="#PropertyRelation" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlIrreflexiveProperty" />
    <Class IRI="#PropertyType" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlMaxCardinality" />
    <Class IRI="#PropertyRestriction" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlMaxQualifiedCardinality" />
    <Class IRI="#PropertyRestriction" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlMinCardinality" />
    <Class IRI="#PropertyRestriction" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlMinQualifiedCardinality" />
    <Class IRI="#PropertyRestriction" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlNegativePropertyAssertion" />
    <Class IRI="#Assertion" />
</SubClassOf>
```

```
<SubClassOf>
    <Class IRI="#OwlOneOf" />
    <Class IRI="#IndividualEnumeration" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlPropertyChainAxiom" />
    <Class IRI="#PropertyRelation" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlPropertyDisjointWith" />
    <Class IRI="#PropertyRelation" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlQualifiedCardinality" />
    <Class IRI="#PropertyRestriction" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlReflexiveProperty" />
    <Class IRI="#PropertyType" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlSameAs" />
    <Class IRI="#Assertion" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlSomeValuesFrom" />
    <Class IRI="#PropertyRestriction" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlSymmetricProperty" />
    <Class IRI="#PropertyType" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlTransitiveProperty" />
    <Class IRI="#PropertyType" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlUnionOf" />
    <Class IRI="#BooleanConnective" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#OwlWithRestrictions" />
    <Class IRI="#DataRange" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#PropertyAxiom" />
    <Class IRI="#Axiom" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#PropertyRelation" />
    <Class IRI="#PropertyAxiom" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#PropertyRestriction" />
    <Class IRI="#Class" />
</SubClassOf>
```

```
<SubClassOf>
    <Class IRI="#PropertySignature" />
    <Class IRI="#PropertyAxiom" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#PropertyType" />
    <Class IRI="#PropertyAxiom" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#RdfType" />
    <Class IRI="#Assertion" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#RdfsDomain" />
    <Class IRI="#PropertySignature" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#RdfsRange" />
    <Class IRI="#PropertySignature" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#RdfsSubClassOf" />
    <Class IRI="#ClassExpression" />
</SubClassOf>
<SubClassOf>
    <Class IRI="#RdfsSubProperty" />
    <Class IRI="#PropertyRelation" />
</SubClassOf>
<DisjointUnion>
    <Class IRI="#Assertion" />
    <Class IRI="#OwlAllDifferent" />
    <Class IRI="#OwlDifferentFrom" />
    <Class IRI="#OwlNegativePropertyAssertion" />
    <Class IRI="#OwlSameAs" />
    <Class IRI="#RdfType" />
</DisjointUnion>
<DisjointUnion>
    <Class IRI="#Axiom" />
    <Class IRI="#Assertion" />
    <Class IRI="#ClassExpression" />
    <Class IRI="#PropertyAxiom" />
</DisjointUnion>
<DisjointUnion>
    <Class IRI="#BooleanConnective" />
    <Class IRI="#OwlComplementOf" />
    <Class IRI="#OwlIntersectionOf" />
    <Class IRI="#OwlUnionOf" />
</DisjointUnion>
<DisjointUnion>
    <Class IRI="#Class" />
    <Class IRI="#BooleanConnective" />
    <Class IRI="#IndividualEnumeration" />
    <Class IRI="#PropertyRestriction" />
</DisjointUnion>
<DisjointUnion>
    <Class IRI="#ClassExpression" />
```

```
        <Class IRI="#OwlAllDisjointClasses" />
        <Class IRI="#OwlDisjointUnionOf" />
        <Class IRI="#OwlDisjointWith" />
        <Class IRI="#OwlEquivalentClass" />
        <Class IRI="#RdfsSubClassOf" />
    </DisjointUnion>
    <DisjointUnion>
        <Class IRI="#DataRange" />
        <Class IRI="#OwlDatatypeComplementOf" />
        <Class IRI="#OwlWithRestrictions" />
    </DisjointUnion>
    <DisjointUnion>
        <Class IRI="#PropertyAxiom" />
        <Class IRI="#PropertyRelation" />
        <Class IRI="#PropertySignature" />
        <Class IRI="#PropertyType" />
    </DisjointUnion>
    <DisjointUnion>
        <Class IRI="#PropertyRelation" />
        <Class IRI="#OwlAllDisjointProperties" />
        <Class IRI="#OwlEquivalentProperty" />
        <Class IRI="#OwlInverseOf" />
        <Class IRI="#OwlPropertyChainAxiom" />
        <Class IRI="#OwlPropertyDisjointWith" />
        <Class IRI="#RdfsSubProperty" />
    </DisjointUnion>
    <DisjointUnion>
        <Class IRI="#PropertyRestriction" />
        <Class IRI="#OwlAllValuesFrom" />
        <Class IRI="#OwlCardinality" />
        <Class IRI="#OwlHasSelf" />
        <Class IRI="#OwlHasValue" />
        <Class IRI="#OwlMaxCardinality" />
        <Class IRI="#OwlMaxQualifiedCardinality" />
        <Class IRI="#OwlMinCardinality" />
        <Class IRI="#OwlMinQualifiedCardinality" />
        <Class IRI="#OwlQualifiedCardinality" />
        <Class IRI="#OwlSomeValuesFrom" />
    </DisjointUnion>
    <DisjointUnion>
        <Class IRI="#PropertySignature" />
        <Class IRI="#RdfsDomain" />
        <Class IRI="#RdfsRange" />
    </DisjointUnion>
    <DisjointUnion>
        <Class IRI="#PropertyType" />
        <Class IRI="#OwlAsymmetricProperty" />
        <Class IRI="#OwlFunctionalProperty" />
        <Class IRI="#OwlInverseFunctionalProperty" />
        <Class IRI="#OwlIrreflexiveProperty" />
        <Class IRI="#OwlReflexiveProperty" />
        <Class IRI="#OwlSymmetricProperty" />
        <Class IRI="#OwlTransitiveProperty" />
    </DisjointUnion>
    <DisjointUnion>
        <Class IRI="#SemanticFeature" />
```

```
        <Class IRI="#Axiom" />
        <Class IRI="#Class" />
        <Class IRI="#DataRange" />
    </DisjointUnion>
    <FunctionalObjectProperty>
        <ObjectProperty IRI="#hasSemanticFeature" />
    </FunctionalObjectProperty>
    <AsymmetricObjectProperty>
        <ObjectProperty IRI="#hasSemanticFeature" />
    </AsymmetricObjectProperty>
    <AsymmetricObjectProperty>
        <ObjectProperty IRI="#hasStat" />
    </AsymmetricObjectProperty>
    <IrreflexiveObjectProperty>
        <ObjectProperty IRI="#hasSemanticFeature" />
    </IrreflexiveObjectProperty>
    <IrreflexiveObjectProperty>
        <ObjectProperty IRI="#hasStat" />
    </IrreflexiveObjectProperty>
    <ObjectPropertyDomain>
        <ObjectProperty IRI="#hasSemanticFeature" />
        <Class IRI="#Stat" />
    </ObjectPropertyDomain>
    <ObjectPropertyDomain>
        <ObjectProperty IRI="#hasStat" />
        <Class abbreviatedIRI="void:Dataset" />
    </ObjectPropertyDomain>
    <ObjectPropertyRange>
        <ObjectProperty IRI="#hasSemanticFeature" />
        <Class IRI="#SemanticFeature" />
    </ObjectPropertyRange>
    <ObjectPropertyRange>
        <ObjectProperty IRI="#hasStat" />
        <Class IRI="#Stat" />
    </ObjectPropertyRange>
    <FunctionalDataProperty>
        <DataProperty IRI="#definitionCount" />
    </FunctionalDataProperty>
    <FunctionalDataProperty>
        <DataProperty IRI="#usageCount" />
    </FunctionalDataProperty>
    <DataPropertyDomain>
        <DataProperty IRI="#definitionCount" />
        <Class IRI="#Stat" />
    </DataPropertyDomain>
    <DataPropertyDomain>
        <DataProperty IRI="#usageCount" />
        <Class IRI="#Stat" />
    </DataPropertyDomain>
    <DataPropertyRange>
        <DataProperty IRI="#definitionCount" />
        <Datatype abbreviatedIRI="xsd:integer" />
    </DataPropertyRange>
    <DataPropertyRange>
        <DataProperty IRI="#usageCount" />
        <Datatype abbreviatedIRI="xsd:integer" />
```

```
    </DataPropertyRange>
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:comment" />
        <IRI>#definitionCount</IRI>
        <Literal xml:lang="en">
            Number of definition of a semantic feature.
        </Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:label" />
        <IRI>#definitionCount</IRI>
        <Literal xml:lang="en">definition count</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:comment" />
        <IRI>#hasSemanticFeature</IRI>
        <Literal xml:lang="en">
            Specify which OWL 2 or RDFS semantic feature is
            the target of the given stat.
        </Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:label" />
        <IRI>#hasSemanticFeature</IRI>
        <Literal xml:lang="en">has semantic feature</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:comment" />
        <IRI>#hasStat</IRI>
        <Literal xml:lang="en">
            Add a statistic to a void:Dataset.
        </Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:label" />
        <IRI>#hasStat</IRI>
        <Literal xml:lang="en">has stat</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:comment" />
        <IRI>#usageCount</IRI>
        <Literal xml:lang="en">
            Number of usage of a semantic feature.
        </Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:label" />
        <IRI>#usageCount</IRI>
        <Literal xml:lang="en">usage count</Literal>
    </AnnotationAssertion>
</Ontology>
```

Listing A.1: Ontology OntoSemStats.

Pierre-Henri PARIS

Identity in RDF knowledge graphs: Propagation of properties between contextually identical entities

Abstract:
Due to a large number of knowledge graphs and, more importantly, their even more numerous interconnections using the *owl:sameAs* property, it has become increasingly evident that this property is often misused. Indeed, the entities linked by the *owl:sameAs* property must be identical in all possible and imaginable contexts. This is not always the case and leads to a deterioration of data quality. Identity must be considered as context-dependent. We have, therefore, proposed a large-scale study on the presence of semantics in knowledge graphs since specific semantic characteristics allow us to deduce identity links. This study naturally led us to build an ontology allowing us to describe the semantic content of a knowledge graph. We also proposed a interlinking approach based both on the logic allowed by semantic definitions, and on the predominance of certain properties to characterize the identity relationship between two entities. We looked at completeness and proposed an approach to generate a conceptual schema to measure the completeness of an entity. Finally, using our previous work, we proposed an approach based on sentence embedding to compute the properties that can be propagated in a specific context. Hence, the propagation framework allows the expansion of SPARQL queries and, ultimately, to increase the completeness of query results.
Keywords:
Semantic Web, Contextual Identity, Property Propagation, Knowledge Graph, RDF, OWL, Ontology, Instance Matching, Linked Data, Sentence Embedding, Completeness, Conceptual Schema Mining.

Résumé :
En raison du grand nombre de graphes de connaissances et, surtout, de leurs interconnexions encore plus nombreuses à l'aide de la propriété *owl:sameAs*, il est devenu de plus en plus évident que cette propriété est souvent mal utilisée. En effet, les entités liées par la propriété *owl:sameAs* doivent être identiques dans tous les contextes possibles et imaginables. Dans les faits, ceci n'est pas toujours le cas et induit une détérioration de la qualité des données. L'identité doit être considérée comme étant dépendante d'un contexte. Nous avons donc proposé une étude à large échelle sur la présence de la sémantique dans les graphes de connaissances, puisque certaines caractéristiques sémantiques permettent justement de déduire des liens d'identités. Cette étude nous a amenés naturellement à construire une ontologie permettant de donner la teneur en sémantique d'un graphe de connaissances. Nous avons aussi proposé une approche de liage de données fondée à la fois sur la logique permise par les définitions sémantiques, et à la fois sur la prédominance de certaines propriétés pour caractériser la relation d'identité entre deux entités. Nous nous sommes aussi intéressés à la complétude et avons proposé une approche permettant de générer un schéma conceptuel afin de mesurer la complétude d'une entité. Pour finir, à l'aide des travaux précédents, nous avons proposé une approche fondée sur les plongements de phrases permettant de calculer les propriétés pouvant être propagées dans un contexte précis. Ceci permet l'expansion de requêtes SPARQL et, in fine, d'augmenter la complétude des résultats de la requête.
Mots clés :
Web Sémantique, Identité Contextuelle, Propagation de Propriétés, Graphe de Connaissances, RDF, OWL, Ontologie, Appariement d'Instances, Données Liées, Plongement de Phrases, Complétude, Extraction de Schéma Conceptuel.