Les anti-motifs pour l'analyse de grands graphes de connaissances incohérents

Thomas de Groot[1], Joe Raad[1,2], Stefan Schlobach[1]

[1]Department of Computer Science, Vrije Universiteit Amsterdam, Pays-Bas
[2] Université Paris-Saclay, LISN, Orsay, France

{t.j.a.de.groot, k.s.schlobach}@vu.nl, joe.raad@universite-paris-saclay.fr

Résumé

Un certain nombre de graphes de connaissances (GC) sur le Web de données contiennent des déclarations contradictoires et sont donc logiquement incohérents. Des méthodes existent pour expliquer une seule contradiction en trouvant ses justifications, qui représente l'ensemble minimal d'axiomes suffisant pour la produire. Dans les grands GC, ces justifications peuvent être fréquentes et peuvent faire référence de manière redondante au même type d'erreur en modélisation. De plus, ces justifications sont par définition dépendantes du domaine, donc difficiles à interpréter ou à comparer. Cet article utilise la notion d'anti-motif (anti-pattern) pour généraliser ces justifications, et présente une approche pour détecter presque tous les anti-motifs à partir de n'importe quel GC incohérent. Des expériences sur des GC de plus de 28 milliards de faits montrent l'évolutivité de cette approche et les avantages des anti-motifs pour analyser et comparer les erreurs logiques entre différents GC. Cet article a été déjà publié à ESWC 2021.

Mots-clés

Web des données, raisonnement, incohérence

1 Introduction

De nos jours, des graphes de connaissances (GC) de milliards de faits sont régulièrement déployés par des chercheurs de divers domaines et entreprises. Étant donné que la plupart des GC sont traditionnellement construits sur une plus longue période de temps, par différents collaborateurs, ces GC sont très susceptibles de contenir des déclarations logiquement contradictoires. En conséquence, le raisonnement sur ces GC devient limité et la connaissance formellement inutile. En générale, une fois que ces déclarations contradictoires dans un GC sont récupérées, elles sont soit expliquées logiquement et réparées, soit ignorées via un raisonnement non standard. Ce travail s'inscrit dans la première catégorie d'approches où l'accent est mis sur la recherche et l'explication de ce qui a été énoncé dans le GC qui cause l'incohérence logique. Comprendre comment ces contradictions se forment et à quelle fréquence elles peuvent se produire est essentiel pour résoudre et éviter de telles contradictions. Au moins, c'est une étape nécessaire pour développer de meilleurs outils capables de gérer des GC incohérents. Pour expliquer les contradictions, la notion de *justification*, qui est un sous-ensemble minimal du GC suffisant pour que la contradiction tienne, joue un rôle clé. Des méthodes existent pour expliquer une seule contradiction en trouvant ses justifications. Bien que les justifications fournissent une bonne base pour expliquer les problèmes de qualité des données et de modélisation dans les GC, leur spécificité dans l'explication des contradictions augmente dans certains cas la complexité d'analyse et de gestion des contradictions détectées. Particulièrement dans les grands GC, ces complexités sont amplifiées et rencontrées dans différentes dimensions :

1. Passage à l'échelle des outils. Les méthodes existantes pour récupérer les justifications de contradiction ne s'adaptent pas aux GC contenant de milliards de faits.

2. Fréquence des justifications. Les contradictions détectées avec leurs justifications peuvent être trop fréquentes pour analyser et comprendre manuellement les erreurs de modélisation commises. Ceci est particulièrement problématique lorsqu'un nombre important de ces justifications récupérées se réfèrent en fait au même type d'erreur, mais instanciées dans différentes parties du GC.

3. Dépendance au domaine des justifications. Vu que les justifications représentent un sous-ensemble du GC, elles sont par définition dépendantes du domaine et nécessitent une certaine connaissance du domaine pour comprendre la contradiction. En plus, ce fait rend la comparaison des contradictions entre différents GC plus difficiles.

Ces différents défis pour trouver et comprendre les justifications dans leur forme traditionnelle, posent les questions de recherche suivantes :

Q1 : Peut-on définir une explication plus générale des contradictions qui catégorise les erreurs les plus courantes dans un GC, indépendamment de son domaine ?

Q2 : Peut-on retrouver ces explications généralisées à partir de n'importe quel GC, indépendamment de sa taille ?

Q3 : Comment ces explications généralisées peuvent-elles aider à analyser et comparer certaines caractéristiques entre les GC les plus couramment utilisés sur le Web ?

2 Approche

Cet article publié à ESWC 2021 présente une méthode pour extraire et généraliser les justifications de tout GC incohérent. Nous appelons ces justifications généralisées des *anti-motifs* (anti-pattern) car elles peuvent être considérées comme un type d'erreurs courantes, produites soit dans les phases de modélisation ou de population du GC, soit éventuellement issues de liage erroné de données.

2.1 Exemples d'anti-motifs

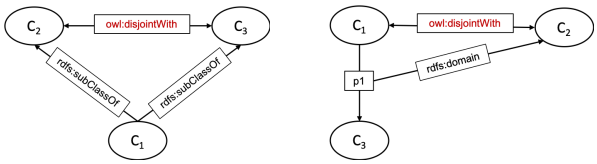
Par exemple, dans l'ontologie de Pizza [1] qui sert de tutoriel pour OWL et l'éditeur d'ontologie Protégé, nous pouvons trouver deux contradictions ajoutées en exprès par ses développeurs. La contradiction (A) démontre la classe insatisfaisable *CheesyVegetableTopping*, qui a deux parents disjoints *CheeseTopping* et *VegetableTopping*. La deuxième contradiction (B) démontre une erreur courante commise, où la classe *Pizza* est affirmée comme le domaine de *hasTopping* malgré la présente d'une restriction de propriété sur la classe *IceCream* stipulant que tous les membres de cette classe doivent utiliser la propriété *hasTopping*. Cependant, comme il est également spécifié que les classes *Pizza* et *IceCream* sont disjointes, forcer maintenant une classe insatisfiable à avoir un membre conduit à une incohérence dans l'ontologie.

Bien que cet exemple se réfère à un cas spécifique d'une contradiction découlant de la description de ces classes, il se réfère également à un type commun d'erreur qui peut être présent dans un autre GC. Cette formalisation de certains types d'erreurs est ce que nous appelons des anti-motifs. Figure 1 présentent les deux anti-motifs généralisant les justifications des contradictions A et B. Afin de transformer une justification en anti-motif, nous remplaçons les éléments en position de sujet et d'objet du BGP par des variables (C_1, C_2, C_3 et p_1 dans figure 1). Afin d'éviter d'éliminer la contradiction, les éléments apparaissant en position prédicat d'une justification ne sont pas remplacés dans l'anti-motif, à l'exception d'un cas : les éléments apparaissant en position prédicat et apparaissant également en position sujet ou objet de la même justification.

2.2 Détection d'anti-motifs

Nous avons développé une méthode qui peut récupérer ces anti-motifs à partir de n'importe quel GC (incohérent). Garantir la récupération de *tous* les anti-motifs nécessite d'abord de trouver *toutes* les contradictions avec leurs justifications, puis de généraliser ces justifications en anti-motifs. En pratique, et comme moyen de relever les défis du passage à l'échelle, notre approche introduit un certain nombre d'heuristiques qui ne permettent pas de garantir son exhaustivité en ce qui concerne la détection de tous les anti-motifs. Cette approche est composée des trois étapes :

Étape 1. La première étape de l'approche consiste à partitionner le GC original en sous-graphes plus petits et qui se chevauchent. Selon la stratégie de partition, cette étape peut avoir un impact sur le nombre de justifications récupérées, qui à son tour peut potentiellement impacter le nombre d'anti-motifs récupérés. Nos expérimentations de la stratégie de partition sur deux GC : LOV (888,017 faits) et YAGO (158 millions faits) montre que partitionner le GC original en des sous-graphes de 5,000 triples impacte le moins le nombre de justifications récupérées.



(a) Anti-motif de la contradiction A

(b) Anti-motif de la contradiction B

FIGURE 1 – Représentation graphique des anti-motif des deux contradictions trouvées dans l'ontologie Pizza

Étape 2. La deuxième étape de l'approche consiste à exécuter un algorithme de récupération de justification pour détecter les contradictions avec leurs justifications. Pour cela, nous utilisons l'algorithme de récupération de justification dans le raisonneur Openllet avec le profil OWL 2 EL, qui parcourt le graphe et trouve la justification minimale pour chaque contradiction. Cette étape est exécutée pour chaque sous-graphe, et toutes les justifications sont ensuite poussées à l'étape finale du pipeline.

Étape 3. La dernière étape consiste à généraliser les justifications en anti-motifs. Les justifications ayant le "même type" sont regroupées. Par conséquent, étant donné une justification et sa généralisation en anti-motif, nous vérifions si un anti-motif avec la même structure existe déjà. Comparer des anti-motifs avec des noms de variables différents consiste à vérifier si ces anti-motifs sont isomorphes. Pour cela, nous implémentons une version de l'algorithme VF2.

Pour évaluer notre approche, nous avons montré sur des GC relativement petits que notre approche peut détecter en pratique tous les anti-motifs malgré le partitionnement du graphe, et montré sur des GC de milliards de faits que notre approche peut être appliquée à l'échelle du Web. Plus précisément, nous avons montré sur les jeux de données *LOD-a-lot* (28,3 milliards faits), *DBpedia* (1 milliards faits), *YAGO* (158 millions faits) que des milliards de justifications peuvent être généralisées en des centaines d'anti-motifs. Bien que ces résultats prouvent la propagation de milliards de faits logiquement contradictoires dans le Web des données, ce travail montre également qu'en utilisant les anti-motifs, ces contradictions peuvent désormais être facilement localisées dans d'autres GC (par exemple, en utilisant une requête *SELECT*), et éventuellement réparées (par exemple, en utilisant une requête *CONSTRUCT*). Enfin, le code source [2] de cette méthode implémenté en JAVA, ainsi que la liste des anti-motifs détectés à partir de ces GC sont en libre d'accès sous forme de requêtes SPARQL.

1. `https://protege.stanford.edu/ontologies/pizza/pizza.owl`

2. https ://github.com/thomasdegroot18/kbgenerator