PyGraft: un outil Python pour la génération de schémas et graphes de connaissance synthétiques

Nicolas Hubert, Pierre Monnin, Mathieu D'aquin, Davy Monticolo, Armelle Brun

PyGraft: un outil Python pour la génération de schémas et graphes de connaissance synthétiques

Nicolas Hubert[1,2], Pierre Monnin[3], Mathieu d'Aquin[2], Davy Monticolo[1], Armelle Brun[2]

[1] Université de Lorraine, ERPI, Nancy, France
[2] Université de Lorraine, CNRS, LORIA, Nancy, France
[3] Université Côte d'Azur, Inria, CNRS, I3S, Sophia-Antipolis, France

prenom.nom@univ-lorraine.fr

Résumé

Des graphes de connaissances (GCs) se sont imposés comme des benchmarks standards pour certaines tâches. Cependant, l'utilisation d'un nombre limité de jeux de données ne permet pas d'évaluer la généralité d'une approche. Nous proposons PyGraft, un outil Python qui génère des schémas et GCs agnostiques et hautement personnalisables. PyGraft rend possible la génération d'un éventail plus diversifié de ressources pour permettre une évaluation plus holistique. PyGraft est disponible en libre accès : `https://github.com/nicolas-hbt/pygraft`.

Mots-clés

Graphe de connaissance, schéma, Web sémantique, générateur synthétique

Abstract

A few knowledge graphs (KGs) have become standard benchmarks for some tasks. However, relying on a limited collection of datasets is insufficient to assess the generality of an approach. We introduce PyGraft, a Python-based tool that generates agnostic and highly customizable schemas and KGs. The aim of PyGraft is to encourage the generation of a more diverse array of resources to allow for a more holistic evaluation. PyGraft is available at : `https://github.com/nicolas-hbt/pygraft`.

Keywords

Knowledge graph, schema, semantic Web, synthetic data generator

1 Introduction

Cet article a été accepté à ESWC 2024 [4].

Les graphes de connaissances (GCs) sont de plus en plus utilisés comme structure de représentation des données en forme de graphe. Plus spécifiquement, un GC est une collection de triplets (s, p, o) où s (sujet) et o (objet) sont deux entités du graphe, et p est un prédicat qui qualifie la nature de la relation les liant [3].

Les GCs sont utilisés dans un large éventail de tâches, pour beaucoup desquelles une collection limitée de GCs s'est établie comme jeux de données standards pour évaluer la performance des modèles. Cependant, dans de nombreuses tâches telles que la classification de noeuds, les jeux de données standards exhibent des caractéristiques similaires, ce qui limite la possibilité d'une évaluation holistique d'un modèle ou d'une approche [6].

Il convient également de souligner le nombre limité de GCs publiquement disponibles dans certains domaines d'application à fort enjeu tels que l'éducation ou la médecine. Dans de tels cas, il devient difficile d'évaluer la qualité d'une approche en l'absence de jeux de données publiquement accessibles. Il faut alors se tourner vers des GCs privés – ce qui n'est pas toujours possible – ou des GCs généralistes – ce qui limite la pertinence et la portée de l'évaluation.

Les problèmes susmentionnés ont conduit à plusieurs tentatives de construction de générateurs synthétiques de schémas et de GCs [5, 7], même si dans la plupart des cas ces deux aspects (génération de schémas *vs.* génération de GCs) ont été considérés séparément [2, 5].

Dans ce travail, nous présentons PyGraft. Contrairement aux approches existantes, PyGraft génère à la fois des schémas et GCs synthétiques et agnostiques à tout domaine.

2 Approche

Les différentes spécifications et étapes dans la génération de schémas et de GCs sont représentées en Figure 1. Il convient de noter que PyGraft permet la génération d'un schéma, d'un GC, ou des deux à la fois. Nous nous focalisons sur le dernier cas pour illustrer notre propos.

L'utilisateur spécifie les paramètres souhaités pour la génération du schéma et du GC (partie gauche de la Figure 1). Un grand nombre de constructions `OWL` et `RDFS` sont permises. Dans un premier temps, PyGraft instancie puis appelle le `Class Generator`, lequel génère un ensemble de classes et de constructions associées, telles que `rdfs:subClassOf` et `rdfs:disjointWith`. Ensuite, ces informations de classes sont passées au `Relation Generator` qui, sur la base des contraintes établies par le `Class Generator` et les spécifications présentes dans les paramètres de schéma, va générer un ensemble de relations et de propriétés associées, telles que `owl:ReflexiveProperty` ou

owl:SymmetricProperty. Le schéma est alors établi. La consistance sémantique de ce dernier est validée via un raisonneur sémantique – dans notre cas HermiT [1]. Enfin, sur la base du schéma généré et validé ainsi que des paramètres utilisateur, le GC est généré sur la base d'autres paramètres tels que le nombre d'entités (num_entities), de triplets (num_triples), la proportion d'entités typées (prop_untyped), le nombre moyen de classes par entités (avg_multityping), etc. Enfin, la consistance sémantique du GC est elle aussi vérifiée (partie droite de la Figure 1).
De plus amples informations sur les constructions OWL et RDFS et les divers paramètres acceptés (p.e., profondeur de la hiérarchie des classes, domaines et co-domaines des relations, etc.) sont répertoriées dans la documentation officielle [1].

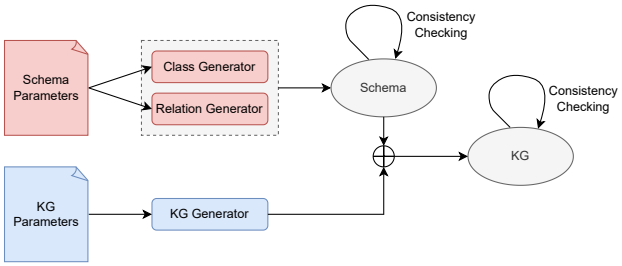

FIGURE 1 – Vue d'ensemble de PyGraft

3 Evaluation

Nous effectuons une évaluation exhaustive des capacités de PyGraft, tant sur le plan de la validité sémantique des ressources produites que de leur temps de génération. En particulier, nous établissons 27 combinaisons différentes de paramètres pour la génération de schémas et de GC : 9 spécifications de schémas et 3 spécifications de GCs avec différents niveaux de complexité et de taille.
La consistance sémantique des 27 combinaisons de ressources générées (schéma et GC) a été systématiquement validée dès la première tentative de génération. Ce résultat valide la capacité de PyGraft à générer des ressources dont la logique interne est vérifiée. Une analyse plus approfondie du temps de génération par combinaison de schéma et GC est présentée dans l'article original [4]. En l'essence, on observe que PyGraft permet de générer des GCs consistants très rapidement, même pour des tailles de GCs semblables aux benchmarks traditionnellement utilisés.

4 Conclusion et travaux futurs

PyGraft est un outil Python pour générer des schémas et GCs synthétiques intégrant un large évantail de constructions OWL et RDFS. PyGraft s'avère utile dans de nombreux scénarios : l'évaluation de nouvelles approches sur un panel plus large de jeux de données, la création de ressources agnostiques dans des domaines à fort enjeu, ou le développement de modèles neuro-symboliques tirant parti d'axiomes ontologiques.
Nos futurs directions de recherche concernent l'optimisation du processus de génération des GCs, afin d'en générer de plus grands et en un temps d'exécution réduit. Nous souhaitons aussi enrichir l'évantail des constructions OWL et RDFS modélisées. Enfin, nous chercherons à rendre possible la génération de littéraux, qui ne sont actuellement pas pris en charge.

1. https://pygraft.readthedocs.io/en/latest

Références

[1] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. Hermit : An OWL 2 reasoner. *J. Autom. Reason.*, 53(3) :245–269, 2014.

[2] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM : A benchmark for OWL knowledge base systems. *J. Web Semant.*, 3(2-3) :158–182, 2005.

[3] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. *Knowledge Graphs*. Synthesis Lectures on Data, Semantics, and Knowledge. Morgan & Claypool Publishers, 2021.

[4] Nicolas Hubert, Pierre Monnin, Mathieu d'Aquin, Armelle Brun, and Davy Monticolo. Pygraft : Configurable generation of synthetic schemas and knowledge graphs at your fingertips. In *The Semantic Web - 21st International Conference, ESWC 2024, Proceedings*.

[5] André Melo and Heiko Paulheim. Synthesizing knowledge graphs for link and type prediction benchmarking. In *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*, volume 10249 of *Lecture Notes in Computer Science*, pages 136–151, 2017.

[6] John Palowitch, Anton Tsitsulin, Brandon Mayer, and Bryan Perozzi. Graphworld : Fake graphs bring real insights for gnns. In *KDD '22 : The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 3691–3701. ACM, 2022.

[7] Jan Portisch and Heiko Paulheim. The DLCC node classification benchmark for analyzing knowledge graph embeddings. In *The Semantic Web - ISWC 2022 - 21st International Semantic Web Conference, Virtual Event, October 23-27, 2022, Proceedings*, volume 13489 of *Lecture Notes in Computer Science*, pages 592–609. Springer, 2022.