Démo : découverte de cardinalités maximales FAIR

Léo-Paul Renard, Arnaud Giacometti, Béatrice Markhoff, Arnaud Soulet

Découverte de cardinalités maximales FAIR

Léo-Paul Renard, Arnaud Giacometti, Béatrice Markhoff, Arnaud Soulet

Université de Tours, LIFAT, Blois
firstname.lastname@univ-tours.fr

Résumé. Les dimensions de qualité FAIR résumées par le slogan "make your data Findable, Accessible, Interoperable, Reusable", portées par le mouvement de la science ouverte, concernent aussi les programmes développés dans le cadre de la recherche. Bien que ces prototypes fonctionnent souvent avec des performances remarquables, ils parviennent rarement à un état réutilisable. Cette démonstration porte sur une démarche d'amélioration des dimensions FAIR pour l'algorithme C3M dédié à la découverte de contraintes de cardinalités maximales contextuelles dans des bases de connaissances du Web. C3M ne bénéficiait jusque-là que d'un dépôt de son code dans Github. Nous présentons comment nous avons amélioré les dimensions FAIR de C3M, en particulier en le rendant utilisable plus simplement en permettant de le tester en ligne depuis un site Web.

1 Introduction

Le Web sémantique relie d'énormes bases de connaissances (BC) dont le contenu a été généré à partir de plateformes collaboratives et par intégration de bases de données hétérogènes. Naturellement, ces bases de connaissances sont incomplètes et contiennent des données erronées. Connaître la qualité de leurs données est un objectif essentiel à long terme pour garantir que leur interrogation renvoie des résultats fiables. Avoir des contraintes de cardinalité qualifiant les propriétés est une avancée importante pour distinguer les individus correctement et complètement décrits de ceux dont les données sont incorrectes ou insuffisamment informées. L'algorithme C3M propose de découvrir automatiquement à partir du contenu de la base de connaissances les contraintes de cardinalité maximale des propriétés pour chaque classe, lorsqu'elles existent. En représentation des connaissances, les contraintes de cardinalité qui spécifient le nombre d'occurrences d'une propriété sont particulièrement utiles (Baader et al., 2003). Par exemple, une telle contrainte peut être utilisée pour définir une classe C comme étant l'ensemble des individus qui ont au plus 3 enfants. De plus, une contrainte de cardinalité peut être utilisée pour déclarer une *cardinalité maximale sur la propriété R dans le contexte C*, par exemple sur `parent` dans le contexte `Person`, pour dire que les individus de la classe `Person` ont au plus deux fois la propriété `parent`. Une telle déclaration permet aux raisonneurs de déduire si toutes les assertions sur la propriété R existent dans la BC pour tout individu appartenant à C. Cela permet d'associer aux réponses des informations sur le *rappel* par rapport à la réalité (Tanon et al., 2017).

L'extraction de contraintes de cardinalité à partir des données existantes est connue comme un problème important de la rétro-ingéniérie des bases de données relationnelles (Soutou,

Person / birthYear

i	n_i	τ_i	$\widetilde{\tau}_i$
1	**159 841**	**0,999**	**0,996**
2	91	0,928	0,775
3	4	0,571	0,000
4	2	0,667	0,000
5	1	1,000	0,000

Person / parent

i	n_i	τ_i	$\widetilde{\tau}_i$
1	10 643	0,529	0,518
2	**9 392**	**0,991**	**0,975**
3	75	0,882	0,718
4	9	0,900	0,420
6	1	1,000	0,000

⊤ / team

i	n_i	τ_i	$\widetilde{\tau}_i$
1	1 221 202	0,901	0,900
2	20 505	0,153	0,148
3	16 876	0,148	0,144
...	...	...	...
20	2	1,000	0,000

FootballMatch / team

i	n_i	τ_i	$\widetilde{\tau}_i$
1	26	0,008	0,000
2	**3 092**	**0,998**	**0,971**
3	3	0,500	0,000
4	2	0,667	0,000
5	1	1,000	0,000

TAB. 1 – *Distributions des cardinalités de plusieurs propriétés dans DBpedia*

1998; Yeh et al., 2008), mais par rapport à ce cadre traditionnel, ce problème est bien plus complexe pour les bases de connaissances du LOD. Pour mieux comprendre ses challenges, la table 1 fournit des statistiques pour trois propriétés de DBpedia, `birthYear` et `parent` dans le contexte de la classe `Person`, ainsi que `team` dans le contexte de tout DBpedia que nous notons $\top$, et dans le contexte de la classe `FootballMatch`. En particulier, les deux premières colonnes donnent le nombre n_i d'observations d'une cardinalité i. Premièrement, les bases de connaissances contiennent des assertions invalides. Par exemple, on s'attend à ce qu'une personne ait au plus une année de naissance et deux parents. Cependant, en considérant `birthYear` et `parent` dans DBpedia (voir la table 1), où i est la cardinalité de la propriété et n_i le nombre d'individus du contexte qui ont exactement i assertions), certaines personnes ont 5 années de naissance ou 6 parents. Deuxièmement, les bases de connaissances sont incomplètes. Par exemple, la plupart des personnes décrites dans DBpedia n'ont qu'un seul parent informé. Néanmoins, il faut tenir compte du fait que de nombreuses personnes ont deux parents informés pour ne pas sous-estimer la cardinalité maximale de la propriété `parent`.

Dans (Giacometti et al., 2019) nous avons présenté l'algorithme C3M qui répond à ces défis et calcule l'ensemble des contraintes contextuelles de cardinalité maximale significatives qui existent dans une BC. Cette démonstration vise non seulement à montrer ses résultats mais aussi la démarche entammée pour le rendre plus facile à trouver, accessible, intéropérable et réutilisable, en particulier grâce à un site Web pour le tester de manière simple. Dans la suite nous rappelons les principes de C3M, puis les principes FAIR, avant de présenter ce site.

2 C3M : découverte de contraintes de cardinalité maximale

Formulation du problème Etant donné un entier $M \geq 1$, une propriété R et une classe C d'une base de connaissances $\mathcal{K}$, une contrainte de cardinalité maximale contextuelle définie sur R pour C est une expression de la forme : $C \sqsubseteq (\leq M\ R)$ en logique de description. La classe C est appelée contexte de la contrainte. Par exemple, la contrainte contex-

tuelle $\texttt{Person} \sqsubseteq (\leq 1\, \texttt{birthYear})$ signifie que chaque personne a au plus une année de naissance, tandis que $\texttt{FootballMatch} \sqsubseteq (\leq 2\, \texttt{team})$ signifie qu'un match de football a au plus 2 équipes. On peut remarquer qu'il est vrai d'affirmer qu'un artiste a au plus 1 année de naissance (i.e., $\texttt{Artist} \sqsubseteq (\leq 1\, \texttt{birthYear})$), mais cette affirmation est moins générale que $\texttt{Person} \sqsubseteq (\leq 1\, \texttt{birthYear})$ puisque $\texttt{Artist} \sqsubset \texttt{Person}$. De manière similaire, affirmer que 1,000 est une cardinalité maximale pour parent (i.e., $\texttt{Person} \sqsubseteq (\leq 1,000\, \texttt{parent})$) est vrai, mais moins précis que $\texttt{Person} \sqsubseteq (\leq 2\, \texttt{parent})$. Pour éliminer de telles contraintes redondantes, C3M vise à extraire les contraintes de cardinalité minimales i.e., celles qui ne sont pas redondantes avec d'autres.

Description de C3M Afin de déterminer si une contrainte est significative au sein d'une base de connaissances $\mathcal{K}$, C3M mesure son taux de cohérence. Le taux de cohérence sur $\mathcal{K}$ que la cardinalité i soit maximale pour la propriété R dans le contexte C est le ratio $\frac{n_i^{C,R}}{n_{\geq i}^{C,R}}$ si $n_{\geq i}^{C,R} > 0$ et 0, sinon où $n_i^{C,R}$ (resp. $n_{\geq i}^{C,R}$) représente le nombre de sujets s du contexte C tels que i faits $R(s, o)$ (resp. i faits ou plus) appartiennent à $\mathcal{K}$. Par exemple, considérons la contrainte contextuelle $\texttt{Person} \sqsubseteq (\leq 2\ (parent)$ dans la table 1. On a $n_{\geq 2}^{\texttt{Person,parent}}$ qui est égal à 9 477 ($9\,477 = 9\,392 + 75 + 9 + 1$). De cette manière, le taux de cohérence $\tau_2^{\texttt{Person,parent}}(DBpedia)$ est de 0,991 (i.e., $9392/9477$).

Dans une base de connaissances complète et correcte $\mathcal{K}^*$, il n'y a aucun individu qui a une cardinalité sur-estimée et tous les individus sont décrits. Pour cette raison, les contraintes contextuelles que nous souhaitons découvrir ont un taux de cohérence égal à 1 dans $\mathcal{K}^*$. $\tau_M^{C,R}(\mathcal{K}^*)$ est appelé le taux de cohérence réel. Comme nous ne disposons pas de la base de connaissances idéale $\mathcal{K}^*$, nous pourrions en pratique estimer le taux de cohérence réel avec le taux de cohérence mesuré sur $\mathcal{K}$. Malheureusement, comme pour toute estimation, le taux de cohérence mesuré dans une base de connaissances est généralement différent du taux de cohérence réel, i.e. $\tau_i(\mathcal{K}) \neq \tau_i(\mathcal{K}^*)$. Par exemple, le taux de cohérence $\tau_2^{\texttt{Person,parent}}(\mathcal{K})$ est de 0,991 dans la table 1 alors que le taux de cohérence réel d'une cardinalité maximale de 2 pour le rôle parent concernant une personne est égal à 1. Plus grave, on a $\tau_6^{\texttt{Person,parent}}(\mathcal{K}) = 1$, alors que le taux de cohérence réel de cette cardinalité est 0 ! Intuitivement, si le fait que le taux de cohérence $\tau_6^{\texttt{Person,parent}}(\mathcal{K})$ soit égal à 1 ne fait pas sens, c'est que ce taux est calculé sur un nombre insuffisant d'observations (seulement une personne a 6 parents). De ce fait, l'estimation $\tau_i(\mathcal{K})$ de $\tau_i(\mathcal{K}^*)$ doit être corrigée. Pour ce faire, comme détaillé dans (Giacometti et al., 2019), C3M utilise le taux de cohérence pessimiste $\widetilde{\tau}_i^{C,R}(\mathcal{K})$ qui exploite l'inégalité de Hoeffding (1963). Formellement, l'algorithme C3M retourne toutes les contraintes minimales de la forme $C \sqsubseteq (\leq M\ R)$ telle que $\widetilde{\tau}_i^{C,R}(\mathcal{K}) \geq min_\tau$.

3 Principes FAIR pour les programmes

Les principes FAIR (Findable, Accessible, Interoperable, Reusable) sont des dimensions de qualité proposées par le mouvement de la science ouverte pour caractériser les productions des activités de recherche quant à leur capacité à être réutilisées. Les bénéfices promis vont d'une meilleure reconnaissance des travaux de recherche ainsi publiés à une multiplication des approches collaboratives, en passant par une meilleure mise en valeur des retombées de la recherche. La Commission Européenne, qui soutient ce mouvement à travers le European

Open Science Cloud (EOSC), a publié fin 2018 une feuille de route fixant des objectifs et des moyens à mettre en oeuvre en ce sens [1]. Si toutes les productions des activités de recherche sont visées par ce mouvement, à savoir les jeux de données, les résultats, les publications, les processus et protocoles, ainsi que les logiciels (les termes Digital Object (DO) sont parfois utilisés pour les désigner), les premières propositions de guides se sont concentrées sur les jeux de données. Le domaine des sciences du vivant est particulièrement moteur sur ce terrain, comme sur celui des ontologies et des données ouvertes liées. Les auteurs de (Wilkinson et al, 2016) ont proposé quinze principes directeurs pour publier des données de façon FAIR [2]. Le premier, **F1**, stipule que le jeux de données ait un *identifiant unique persistant (PID)*, notion qui recouvre par exemple les DOI et ARK (données), ORCID (auteurs), RAiD (projets), PURL et w3id (localisations dans le web). Les plateformes regroupant les données d'un domaine, par exemple UniProt, génèrent automatiquement un PID au moment du dépôt du jeu de données. Ensuite, pour que des logiciels puissent trouver, accéder, interopérer et réutiliser des données, le deuxième principe, **F2**, est d'associer aux données des *métadonnées* (elles-mêmes munies d'un PID). Celles-ci sont donc omniprésentes dans les quinze principes. Le troisième, **F3**, est que les métadonnées contiennent explicitement une référence aux données qu'elles décrivent. Le quatrième, **F4**, est d'enregistrer les (méta)données dans un répertoire public (comme BioPortal ou Zenodo) où elles puissent être recherchées. Les cinquième à septième, **A1**, **A1.1** et **A1.2**, sont qu'elles puissent être récupérées via un protocole de communication standard, ouvert et qui permettent un contrôle des droits d'accès. Le huitième, **A2**, est que les métadonnées restent récupérables mêmes si les données ne le sont pas. Les neuvième et dixième, **I1** et **I2**, portent sur les vocabulaires et ontologies décrivant les (méta)données, qui doivent être aussi standard que possible et eux-mêmes respecter les principes FAIR. Le onzième, **I3**, est que les (méta)données doivent inclure des références à d'autres (méta)données. Les douzième à quinzième, **R1**, **R1.1**, **R1.2** et **R1.3**, recommandent un choix d'attributs permettant une réutilisation aussi informée que possible, adapté aux usages dans le domaine propres aux données, incluant une license d'utilisation et des informations de provenance.

Dans (Lamprecht et al., 2020) les auteurs dressent un bilan des réflexions de différents groupes sur l'adaptation des quinze principes FAIR que nous venons de brièvement rappeler aux *logiciels* de la recherche, conçus pour générer et analyser des données et résultats, que ce soient des codes sources, des exécutables ou des services web. Si un logiciel est un objet digital au même titre qu'un jeu de données (on peut lui attacher un DOI, lui affecter une licence d'utilisation, etc.), c'est aussi un *outil*, qui a un comportement et des interfaces, nécessite un environnement d'exécution, peut utiliser d'autres outils, et requiert en général des données en entrée, selon un certain workflow. Un logiciel est donc associé à un ensemble de *dépendances*. De plus, chaque partie peut évoluer, la notion de *version* est donc très importante. La publication d'un logiciel peut se faire via des plateformes comme Github, des sites web, des répertoires spécifiques à un langage comme par exemple PyPI, etc. Le mouvement Open Source côté logiciel rejoint sur certains points le mouvement FAIR et si, côté données, FAIR n'implique pas forcément licence ouverte, côté logiciel on peut aisément argumenter pour l'ouverture des codes. Enfin, il faut bien noter que la question de la qualité d'un logiciel dépasse largement les principes FAIR, qui ne traitent que de la façon dont le logiciel est fourni.

1. https ://op.europa.eu/en/publication-detail/-/publication/7769a148-f1f6-11e8-9982-01aa75ed71a1/language-en
2. Repris ici : `https://www.go-fair.org/fair-principles/`

	Principe	C3M	Résultats
F	1) identifiant unique persistant , 2) métadonnées, 3) références aux données, 4) accès public	oui	oui
A	1) protocole ouvert, 2) disponibilité des métadonnées	oui	oui
I	1) vocabulaire standard pour les données et 2) les métadonnées, 3) références à d'autres métadonnées, 4) dépendances logicielles informées	oui	oui
R	licences compatibles, réutilisation informée et adaptée	partiel	oui

TAB. 2 – *Principes FAIR pour le prototype C3M et les résultats produits*

Tout ceci étant dit, les auteurs de Lamprecht et al. (2020) revisitent et complètent les quinze principes FAIR pour les logiciels. En résumé : **F1** et **F3** doivent s'appliquer à chaque *version* du logiciel ; pour **I2** il est précisé que les descriptions du logiciel, mais aussi celle des données qu'il utilise ou qu'il produit, utilisent des vocabulaires ou ontologies standards qui respectent eux-même les principes FAIR ; un **I4** est introduit pour que les dépendances du logiciel soient documentées et qu'il y ait un moyen d'y accéder ; enfin, pour **R1.1** il est précisé que la licence associée au logiciel doit être compatible avec celles des autres codes qu'il utilise. De plus, certaines des spécificités du logiciel, parmi celles rappelées précédemment, nécessitent des outils pour réaliser les principes FAIR : ainsi l'interopérabilité, qui est une des quatre dimensions la plus complexe à assurer, doit prendre en compte le problème de la portabilité, c'est-à-dire la disponibilité de tout *l'environnement d'exécution* nécessaire. Cet environnement peut être fourni de manières très diverses, de la machine virtuelle Java aux containers Docker, en passant par une encapsulation dans un site Web et/ou une API Web, l'exécution étant alors prise en charge par le serveur Web. Des efforts remarquables en ce sens peuvent être cités dans le domaine de recherche du web sémantique, comme SPARQL Generate [3] ou SAGE [4]. Cette démo s'inscrit modestement dans leur sillage, en permettant de tester en ligne C3M.

4 C3M plus FAIR

Hébergements sur le Web du code source et des résultats La première initiative pour faciliter la reproductibilité scientifique et plus généralement satisfaire les principes FAIR est de mettre l'ensemble du code source, des données utilisées et des résultats produits à disposition sur le Web. De cette manière, le prototype C3M satisfait le principe **F1** car l'URL du GitHub [5] identifie le prototype de manière unique. Par ailleurs, le jeu de données a aussi été enregistré sous la plateforme Zenodo pour disposer d'un DOI [6] correspondant au prototype C3M. Sous cette même plateforme, plusieurs métadonnées sont disponibles notamment par rapport à la publication scientifique et la licence d'utilisation qui réfèrent explicitement au prototype (satisfaction des critères **F2** et **F3**). Des métadonnées sont également présentes dans le GitHub, exprimées avec les vocabulaires standard DCAT, DCterms, VOID, PROV-O et schema.org. L'hébergement Web garantit également un accès public (critère **F4**) facilité notamment par les

3. https://ci.mines-stetienne.fr/sparql-generate/
4. http://soyez-sage.univ-nantes.fr/
5. https://github.com/asoulet/C3M/tree/v1.0
6. http://doi.org/10.5281/zenodo.4106070

moteurs de recherche des plateformes GitHub et Zenodo, reposant sur des protocoles standardisés du Web (critère **A1**). La disponibilité de ces plateformes de référence est très haute et garantit aussi une très haute disponibilité des (méta)données (critère **A2**). Les résultats générés à partir de C3M reposent sur des vocabulaires et des ontologies standardisés du Linked Open Data satisfaisant pleinement le critère **I**. Le prototype lui-même est décrit à l'aide du vocabulaire schema.org dans le GitHub et dans le site Web à l'aide de RDFa. Concernant la réutilisation, le code source Java du prototype est disponible sur GitHub avec l'ensemble des librairies nécessaires (par exemple, Jena). Néanmoins, la documentation du prototype n'est pas suffisante actuellement, c'est un point d'amélioration future. Un autre axe d'amélioration prévu, important pour les principes **I** et **R**, est la mise à disposition du prototype via une API REST Web, qui sera elle-même décrite avec Open API pour faciliter son utilisation dans des programmes clients. Pour l'heure, seule l'interface Web pour les humains est opérationnelle. Elle permet à tout un chacun de vérifier si une propriété présente dans une base de connaissances du Web a des contraintes de cardinalité maximale dans certaines classes de cette base.

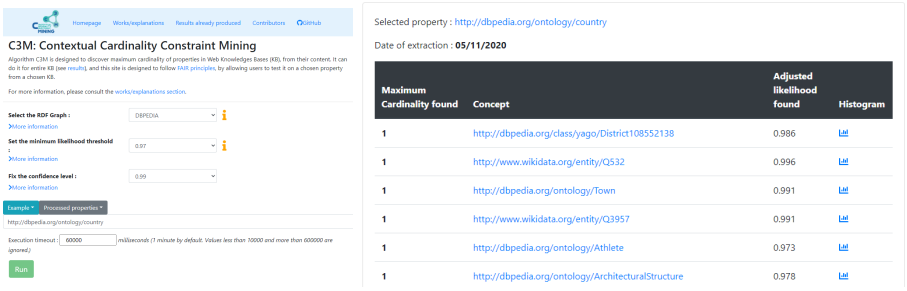

FIG. 1 – *Captures d'écran des entrées et de la sortie pour la propriété* `dbo:country`

Prototype en ligne La difficulté de déploiement d'un prototype (clonage, compilation, exécution) est parfois un frein à sa réutilisation. Par conséquent, nous proposons le site Web `http://c3m.univ-tours.fr/` où une interface simple permet de tester C3M en quelques clics. Toujours dans un souci d'accessibilité, cette application Web n'effectue pas une extraction des contraintes de cardinalité pour toutes les propriétés d'une base de connaissances. Pour avoir un résultat rapide, l'extraction se focalise sur une seule propriété. Néanmoins, le fonctionnement reste similaire et reprend le code source de C3M en Java. Pour cette raison, l'application web a également été codée en Java en utilisant l'architecture J2EE 11.0.7 ainsi que le conteneur web Apache Tomcat 9.0. Nous utilisons MySQL 8.0.20 comme SGBD.

Sur la page principale (cf. la figure 1, gauche), l'utilisateur indique les paramètres suivants :

— La base de connaissances $\mathcal{K}$: Pour le moment le nombre de bases de connaissances disponibles est fixe. Nous envisageons d'ajouter une fonctionnalité permettant à l'utilisateur de rentrer son propre point d'accès SPARQL.

— Le seuil minimal du taux de cohérence min_τ et le niveau de confiance $1 - \delta$: Le seuil min_τ spécifie le taux de cohérence pessimiste minimale pour qu'une contrainte soit significative. De plus, $1 - \delta$ intervient dans le calcul du taux de cohérence pessimiste. Des valeurs raisonnables sont suggérées pour ces deux paramètres.

— La propriété R : Pour chaque base de connaissances des exemples de propriétés sont disponibles pour pouvoir, sans connaître de noms de propriété, utiliser le prototype. L'utilisateur peut également sélectionner une propriété déjà traitée.

— Temps d'arrêt de l'exécution : La valeur par défaut est une minute. L'énorme quantité de données de certaines bases de connaissances peut engendrer un temps de traitement long, même pour une seule propriété.

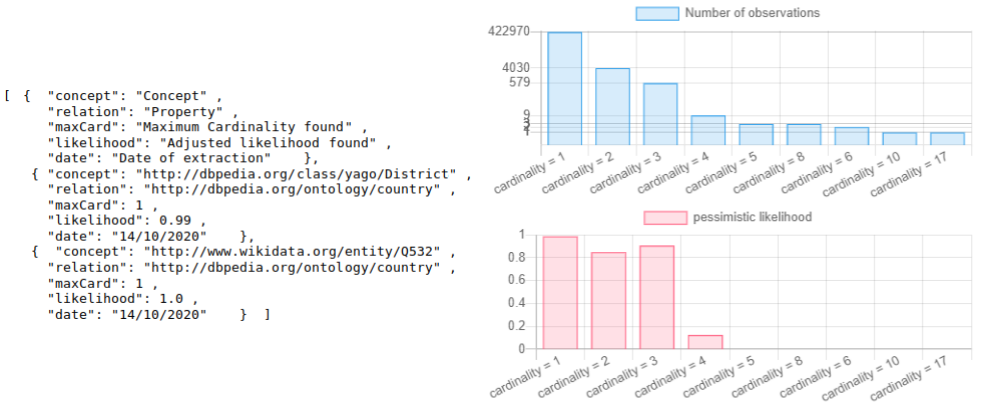

FIG. 2 – *Liste des contraintes au format JSON et représentation visuelle d'une contrainte*

Lorsque l'utilisateur lance l'exécution, le serveur vérifie que la propriété est bien présente dans la base de connaissances cible. Si ce n'est pas le cas, un message d'erreur est retourné à l'utilisateur, sinon l'implémentation partielle de l'algorithme C3M effectue son traitement. L'application gère aussi différents types d'erreurs tels que les erreurs provenant des points d'accès SPARQL ou encore s'il n'y a aucune contrainte significative détectée car pas assez d'instances pour cela dans la base. Une fois l'exécution terminée le serveur retourne une liste, au format JSON, de contraintes de cardinalité maximale chacune associée à une classe avec ses paramètres associés (voir la partie gauche de la figure 2 qui présente un extrait du résultat pour la propriété `dbo:country`). Ces résultats sont affichés à l'utilisateur sous forme d'une liste (voir la figure 1 à droite) et l'utilisateur peut avoir le détail pour une contrainte. Par exemple, la figure 2 (à droite) présente, pour chaque cardinalité, l'histogramme des individus (en bleu) et le taux de cohérence pessimiste (en rouge).

A noter que les contraintes des propriétés déjà traitées sont mises en cache dans une base de données pour limiter le nombre de requêtes sur les points d'accès SPARQL pendant une séquence de tests. Un inconvénient de cette optimisation est que les informations d'une base de connaissances varient. Cette approche ne garantit pas la fraîcheur des contraintes déjà extraites. C'est pour cela que la date d'extraction est renseignée dans les résultats.

Enfin, le site dispose également de quatre autres pages pour référencer la publication scientifique qui décrit C3M, indiquer les contributeurs, mettre à disposition les résultats de la publication et le code source du prototype, en redirigeant vers le dépôt GitHub.

5 Conclusion

Cette démonstration vise non seulement à montrer les résultats que produit C3M, un algorithme qui calcule l'ensemble des contraintes contextuelles de cardinalité maximale significatives qui existent dans une base de connaissances du Web, mais aussi la démarche entammée pour le rendre plus facile à trouver, accessible, intéropérable et réutilisable, en particulier grâce à un site Web conçu pour pouvoir le tester (dans une version partielle) très simplement. Le site Web en lui-même peut encore être amélioré, la documentation du prototype également. Un autre axe d'amélioration prévu, important pour les principes I et R, est la mise à disposition du prototype via une API REST Web, qui sera elle-même décrite avec Open API pour faciliter son utilisation dans des programmes clients.

Remerciements. Ce travail a été en partie financé par le projet ANR-18-CE38-0009 ("SESAME").

Références

Baader, F., D. Calvanese, D. L. McGuinness, D. Nardi, et P. F. Patel-Schneider (Eds.) (2003). The Description Logic Handbook : Theory, Implementation, and Applications. New York, NY, USA : Cambridge University Press.

Giacometti, A., B. Markhoff, et A. Soulet (2019). Mining significant maximum cardinalities in knowledge bases. In International Semantic Web Conference, pp. 182–199. Springer.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association 58(310), 13–20.

Lamprecht, A.-L., L. Garcia, M. Kuzak, C. Martinez, R. Arcila, E. Martin Del Pico, V. Dominguez Del Angel, S. van de Sandt, J. Ison, P. A. Martinez, P. McQuilton, A. Valencia, J. Harrow, F. Psomopoulos, J. L. Gelpi, N. Chue Hong, C. Goble, et S. Capella-Gutierrez (2020). Towards FAIR principles for Research Software. Data Science, 37–59.

Soutou, C. (1998). Relational database reverse engineering : algorithms to extract cardinality constraints. Data & Knowledge Engineering 28(2), 161–207.

Tanon, T. P., D. Stepanova, S. Razniewski, P. Mirza, et G. Weikum (2017). Completeness-aware rule learning from knowledge graphs. In ISWC, pp. 507–525. Springer.

Wilkinson, M. D. et al (2016). The FAIR guiding principles for scientific data management and stewardship. Scientific Data 3.

Yeh, D., Y. Li, et W. Chu (2008). Extracting entity-relationship diagram from a table-based legacy database. Journal of Systems and Software 81(5), 764–771.

Summary

The FAIR principles carried by the open science movement also apply to softwares developed in research. Although these prototypes often perform remarkably well, they rarely reach a reusable state. This demo focuses on a FAIR improvement approach for C3M dedicated to the discovery of contextual maximum cardinality constraints in Web knowledge bases. We present how we improved FAIR features by making its usage simpler with a Web site.