Web Sémantique - Utilisation d'OWL et SHACL

Par : Ambre Correia & Jeannes Théo

Pour ce projet, nous sommes reparties de notre projet de Web Of Linked Data, modélisant des rapports médicaux, auquel nous avons ajouté des concepts SKOS. Nous avons ensuite tiré parti d'OWL pour créer une ontologie, ainsi que de SHACK pour valider les données.

La première étape a été de corriger les différents problèmes de notre premier projet. Nous avons tout d'abord changé le type attendu pour la valeur de `s:hasDoctorNumber`, pour mettre un nombre à la place d'une chaine de caractère. Nous avons ensuite changé les données pour utiliser des ressources anonymes pour créer les maladies ou problèmes, puisque celles-ci ne sont utilisées qu'une seule fois.

Nous avons ensuite fini par retirer les définitions `skos` des classes utilisées pour typer. Par exemple, nous avons retiré les relations `skos` de la classe `Medecin`, afin de pouvoir définir des docteurs.

I. Concepts SKOS

Nous avons ajouté des concepts `skos` pour définir les spécialités médicales, comme `Cardiology` ou `Surgery`. Les spécialités sont définies comme des concepts, car elles ne sont pas tangibles, ce ne sont pas des objets physiques, mais sont susceptibles d'être utilisées plusieurs fois dans d'autres cas, pour classer des papiers scientifiques par exemple.

Puisque les spécialités ont été définies comme des concepts, nous avons utilisé les relations `skos:broaderTransitive` et `skos:narrowerTransitive` pour créer une hiérarchie parmi les concepts. Par exemple, la spécialité `s:GeneraleMedicine` est plus générale que les spécialitées `s:ENTField` et `s:Rheumatology`.

II. Modélisation OWL

Nous avons commencé par changer les propriétés pour utiliser les propriétés owl `DatatypeProperty` et `ObjectProperty`. Nous avons ensuite défini de nouvelles

classes, qui ont la possibilités depuis les données.

Nous avons commencé par créer la classe *s*:Person comme une union des médecins et des patients. Cela nous a permis d'établir une équivalence avec la classe foaf:Person. Nous avons ensuite défini la classe *s*:ThreatenedPatient, ce qui permet d'identifier de potentielles personnes fragiles dans le graph sans qu'elles soient déclarées explicitement. Nous avons donc défini un patient à risque grâce à une restriction owl, comme un patient avec trois maladies héréditaires, ou deux maladies de longue durée.

En utilisant des restrictions, nous pouvons aussi identifier plus facilement certains types de traitements. En effet, nous avons mis en place deux nouvelles classes OWL en utilisant des restrictions sur la propriété s:givenTreatment. La première classe regroupe tous les rapports médicaux qui ont soigné un problème uniquement avec des médicaments, alors que la seconde classe regroupe les rapports qui soignent des problèmes avec d'autres méthodes, comme des séances de balnéothérapie. Ces classes pourraient être utilisées pour identifier rapidement les problèmes qui peuvent être soignés en utilisant moins de médicaments.

Nous avons ensuite créé la classe Gender, pour mieux définir les sexes des patients, ce qui est primordial pour adapter le suivi médical. Nous avons donc défini la classe Gender avec une disjonction, à l'aide de owl:disjointUnionOf.

Nous avons également utilisé OWL pour préciser les relations. Dans un premier temps, nous avons utilisé owl:hasKey pour préciser que les numéros de sécurités sociales, les identifiants de docteur et les numéros de rapports médicaux permettent d'identifier des ressources de manière certaine en cas de confusion. Nous avons renforcé cette contrainte sur les numéro de sécurité sociale, en définissant la relation *s*:hasSocialSecurityNumber comme une relation fonctionnelle et inversement fonctionnelle, puisque chaque numéro identifie une personne, et chaque personne est identifiée par un unique numéro de sécurité sociale.

Pour finir, nous avons créé des relations pour relier les individus dans une même famille, ce qui est nécessaire pour identifier les problèmes génétiques, ou pour les détecter quand ceux-ci ne sont pas connus. Nous avons donc ajouté la relation *s*:sibling, qui est symétrique et transitive, et la relation *s*:hasChild
qui est nécessairement asymétrique. Nous avons ensuite défini *s*:hasParent, comme la relation inverse à *s*:hasChild. Cela nous a permis d'ajouter que la relation *s*:sibling pouvait également être inférée comme l'enfant d'un parent, en écrivant *owl*:propertyChainAxiom (*s*:hasParent *s*:hasChild). Nous avons également ajouté la relation *s*:notDirectlyRelated, comme le frère ou la sœur d'un

ascendant. Un ascendant est défini à l'aide d'une relation transitive, comme une personne dans la ligne généalogique direct d'une personne. Cela permet également de surveiller les gens proches d'un patient, mais dont l'ancêtre commun n'est pas renseigné dans les données.

Le fichier *queries.md* contient quelques requêtes qui permettent de mettre en avant les relations ajoutées grâce à l'inférence OWL.

III. Contraintes SHACL

Nous avons utilisé SHACL pour définir des contraintes. Chaque jeu de contrainte est dans un dossier avec un jeu de données permettant de vérifier que les contraintes s'appliquent correctement. Le fichier *summary.md* résume les contraintes, et précise les nodes qui sont valides.

Nous avons commencé par imposer des restrictions liées au patient, dans le dossier 1. Nous avons commencé par préciser qu'un patient a forcément un nom, et au moins un prénom. Nous avons ensuite ajouter des contraintes sur le poids et le sexe. Le sexe doit nécessairement être l'une des trois valeurs parmi Male, Female ou Other. Pour le poids, nous avons stipulé que le poids était nécessaire pour former un patient correct, et qu'il devait être supérieur à 0, inférieur ou égal à 250 kilos. Nous avons configuré la sévérité à `sh:Info` puisque cette information n'est pas critique, mais est importante pour le dossier du patient. Nous avons également défini le message en cas d'erreur, pour le rendre lisible pour un opérateur humain.

Nous nous sommes ensuite penché sur les numéros de sécurité sociale, dans le dossier 2. Nous avons contraint tous les patients à avoir exactement un numéro, puisqu'il sert d'identifiant. Nous avons ensuite utilisé un regex, avec *`sh:pattern`*, pour s'assurer que les chaînes de caractères étaient bien des suites de chiffre, de 13 chiffres de long. Ces dispositions permettent de s'assurer que les numéros de sécurité sociale sont des numéros valides.

Nous avons ensuite ajouté dans le dossier 3 des contraintes sur les problèmes médicaux. En effet, celles-ci étant définies avec des ressources anonymes, il est primordial de définir des contraintes pour s'assurer qu'elles comportent les informations nécessaires. Nous avons commencé par imposer qu'une date soit présente, avec le format *`xs`*`:date`. Par souci de cohérence, nous avons imposé que la date soit antérieure au 31 décembre 2023. Pour un système réel, cette date devrait être changée, mais cela permet de s'assurer que les dates des événements soient cohérentes. Nous avons également imposé qu'une maladie soit précisée pour chaque problème médical. Chaque maladie doit être définie par une classe du thésaurus NCI, par souci de cohérence. Pour cela, chaque valeur doit être

une node de type IRI, ce qui est défini en écrivant `sh:nodeKind sh:IRI`. Nous avons également ajouté une contrainte pattern, pour s'assurer que les ressources sont bien des classes qui proviennent de NCIT.

Les dernières contraintes ajoutées concernent les rapports médicaux et les suivis, présentes dans le dossier 4. Pour commencer, nous avons utilisé des séquences pour nous assurer que les rapports sont écrits par une personne avec un identifiant de médecin, ce qui permet de garantir qu'il s'agit d'un membre du personnel médical. Nous avons également ajouté une contrainte précisant que le médecin et le patient doivent avoir exactement 2 noms de familles. Cette propriété seule n'est pas suffisante, puisqu'une personne pourrait avoir 2 noms de famille. Toutefois, combiné aux restrictions dans le premier dossier, cela peut assurer que le patient et le médecin ne peuvent avoir qu'un seul nom chacun pour qu'un rapport médical soit valide. Nous avons également ajouté des contraintes sur les suivis. Pour être utile, un suivi doit avoir un champ précisant si le patient est en bonne santé. Nous avons donc imposé que ce champ soit présent, une seule fois par suivi seulement, et qu'il soit un booléen. Pour permettre plus de détails dans le suivi du patient, nous avons également imposé qu'au moins un commentaire soit fait. Pour éviter que le champ de texte ne soit compréhensible, nous avons ajouté une contrainte pour que le commentaire ne puisse être qu'en français, ou en anglais, et une seule fois par langue.