Web of Linked Data - Choix de modélisation

Par : Ambre Correia & Jeannes Théo

I. Introduction

Le but de ce projet est de créer un schéma RDFS pour représenter des *case reports*, soit des rapports médicaux. Dans ce rapport, nous expliquerons la modélisation liée à notre modèle RDFS pour décrire des observations médicales, ainsi que les choix de conception que nous avons fait.

Dans ce modèle, nous avons utilisé plusieurs ontologies usuelles, dont FOAF pour décrire le patient et le médecin, et Dublin Core pour décrire les rapports médicaux ainsi que leurs relations, afin de garder un standard pour la gestion des documents. Enfin, nous utilisons également SKOS, cela nous permet de labelliser, définir et donner des exemples de relations ou de classes dans un standard compris de tous.

Nous avons choisi d'utiliser uniquement SKOS pour décrire et labelliser les documents, pour éviter la duplication d'informations. Nous aurions également pu utiliser Dublin Core pour cela, avec la propriété `dcterms:description` par exemple.

Nous avons également utilisé NCIT, ou National Cancer Institute Thesaurus, tout au long de la modélisation. Ce thésaurus est très utilisé pour les modèles qui traitent de médecine, et il contient de nombreuses classes et relations, tant sur les médicaments, que sur les symptômes ou les maladies. Nous nous sommes donc appuyés dessus pour que notre schéma puisse être utilisé ou intégré facilement par quelqu'un qui serait familier avec le NCIT.

II. Modélisation des personnes

Nous avons commencé par modéliser les patients et les médecins, en étendant la classe `foaf:Person` pour conserver une ontologie connue. Les noms, prénoms et date de naissance sont représentés en utilisant foaf.

Pour représenter le médecin, nous avons décidé d'associer le statut de médecin provenant de NCIT grace a `foaf:status`. Le médecin possède un identifiant unique défini grâce à la classe identifiant du Dublin Core pour être facilement identifié, et une spécialité qui correspond à une spécialité liée à NCIT.

Nous avons ensuite défini des relations pour permettre de relier les patients à leurs informations essentielles. Nous avons commencé par la taille, le poids, le sexe, le numéro de sécurité sociale et l'adresse comme des relations entre une personne et l'attribut décrit. Cela permet de garder des relations génériques qui peuvent être appliquées à d'autres classes qu'un Patient en cas de réutilisation. Pour des raisons de respect de la vie privée, les adresses sont définies par deux relations, une pour le code postal et une pour le pays.

Nous avons ensuite défini les antécédents médicaux, les maladies et les syndromes préexistants comme des relations entre un patient et une maladie ou une blessure, concept défini par NCIT. Nous avons également défini les conditions médicales dans la famille, qui peuvent être importantes en cas de problèmes héréditaires. Pour cela, nous avons créé la propriété <knownHereditaryCondition> qui peut s'appliquer à la fois d'un problème médical et a un problème héréditaire, autre concept tiré de NCIT.

III. Modélisations des rapports médicaux

Pour modéliser le cœur de notre schéma, c'est-à- dire les rapports médicaux, nous avons commencé par créer la classe <CaseReport>.

Nous avons ensuite défini une relation pour donner un identifiant unique à la classe *ncit*:C19498, qui correspond au concept de document dans NCIT, en étendant la relation *dc*:identifier du Dublin Core. Nous avons ensuite établi des propriétés entre un rapport et des classes de notre propre ontologie, comme par exemple avec des relations qui décrivent l'auteur du rapport ou bien qui est le patient concerné par le rapport. Nous avons fait de même avec des classes NCIT, pour pouvoir lier les symptômes, le diagnostic ou d'éventuels traitements complémentaires au rapport. Nous avons ajouté une relation pour relier la date de rédaction, étendue de la relation *dc*:date de Dublin Core, pour compléter les informations administratives.

La relation *skos*:*related* peut être utilisée pour lier deux rapports concernant la même personne, tandis que *rdfs*:seeAlso pourra être utilisé pour lier deux rapports qui présentent des similitudes.

Nous avons également créé des propriétés pour lier des traitements ou des follow-up à un rapport médical. Le follow-up permet de lier l'état du patient à une date ultérieure au rapport, ce qui permet de mesurer l'efficacité du traitement donné par exemple.

Le follow-up est une classe qui descend de la classe *ncit*:C19498, ce qui permet de réutiliser les propriétés définies précédemment, que ce soit pour savoir qui écrit le document, qui est le sujet et à quelle date le document est écrit. Nous avons créé deux

propriétés de plus pour les <Follow-Up>. La première permet de lier un booléen au follow-up, pour s'assurer que le patient est en bonne santé, tandis qu'une deuxième relation lie une chaîne de caractère au Follow-up, ce qui permet à un médecin d'inscrire plus de détails sur l'état du patient. Grâce à ce système de Follow-up, il est possible de suivre l'évolution d'un patient dans un rapport médical, simplement en liant plusieurs Follow-Up avec différentes dates.

IV. Modélisation des traitements

Pour finir, nous avons modélisé des traitements. Pour cela, nous avons créé la classe <Treatment>. Nous avons ensuite créé la classe <Drug> et la classe <MedicalActivity>. Les deux classes étendent la classe <Treatment>, ce qui permet de les utiliser pour les relations désignant des traitements. Nous avons lié les classes précédentes dans la classe <Treatment> à l'aide de *skos*:*narrower* pour que ces concepts soient plus faciles à identifier pour quelqu'un qui voudrait s'approprier notre schéma.

Ensuite, nous avons développé deux relations : l'une ayant pour sujet le concept NCIT d'un médicament, et l'autre ayant comme sujet le concept NCIT d'activité médicale, ce qui englobe les opérations et les traitements chez un professionnel ou dans un lieu spécifique, tels que les massages, la balnéothérapie, les séances de kinésithérapie. Ces deux relations permettent de spécifier le contenu d'un traitement, tout en séparant clairement les relations servant à définir un médicament des relations servant à définir une activité.

Nous avons ajouté des relations dans le traitement, pour indiquer différentes informations. Nous avons plusieurs relations utiles pour les médicaments, comme la durée du traitement et le nombre de jours d'écart entre deux prises. Ces deux relations n'acceptent que des attributs de type *xs*:dayTimeDuration pour standardiser l'écriture et éviter des problèmes d'interprétation. Nous avons également ajouté une relation pour définir le nombre de prises par jour, notamment d'un médicament.

Afin de pouvoir définir les informations liées à des activités médicales, nous avons ajouté plusieurs relations avec la classe <Treatment> comme sujet. Nous avons construit une relation pour expliquer ce qui doit être fait précisément lors d'une opération, et une relation pour indiquer le nombre de séances nécessaires, qui est simplement définie par un entier, ainsi qu'une relation pour pouvoir donner des informations complémentaires.

Cette modélisation nous permet d'indiquer à la fois un médicament ou une opération comme traitement, et de donner toutes les informations nécessaires liées à la bonne exécution du traitement.