Master Sciences, Technologies, Santé
Mention Santé Publique
Parcours Systèmes d'Informations et Technologies Informatiques pour la Santé
2019- 2020

Projet

Recherche d'Information

INF203

Recherche d'information, indexation et fouille de textes

Marc Fouqué marc.fouque@etu.u-bordeaux.fr

lien git : https://github.com/marcfouque/gvri.git

Introduction

Le projet consiste en l'élaboration d'un moteur de recherche se basant sur les données de l'UE INF202.
Le corpus de document correspondant à plusieurs fichiers semblables à des fichiers *CSV* (Comma separated value), ayant pour base ce type de fichier, le parti a été pris de transformer ces fichiers en un document *RDF* (Resource Description Framework) permettant une liaison à des ressources externes et une hiérarchisation et une connexion des concepts.

Méthodes/Résultats

Technologie

Le moteur de recherche a été conçu avec le langage Java (java 11.0.5), en utilisant le gestionnaire de package maven (Apache Maven 3.6.3) et fournissant une interface graphique grâce à JavaFX (javaFX 11).
La gestion du fichier RDF a été permise par l'utilisation de Jena (Apache Jena, *jena-core* et ses extensions *jena-arq* et *jena-text* permettant respectivement une large possibilité de requête Sparql et la prise en compte d'une recherche textuelle poussée) et de Fuseki (implémentation de Jena en tant que serveur), les requêtes effectuées aux travers de Jena ont été faites en SPARQL . Jena inclut en son sein Lucene utilisé, notamment pour l'indexation du fichier RDF.
Le fichier RDF a été créé à l'aide de Tarql, un outil permettant le passage de fichier CSV à un fichier RDF. Le développement effectué sur Windows 10, a demandé l'utilisation de script batch (.bat, équivalent shell Unix) permettant d'automatiser l'usage de Tarql.

Préparation

La première étape de ce projet est la constitution de la source de données, un fichier RDF. L'ensemble du corpus (*./corpus*) a été converti en un seul fichier rdf grâce à l'utilisation de la fonction *Construct* de *Sparql.* Chaque fichier *.txt* est associé avec un fichier *.sparql* (*./tarql/sparql*) permettant la génération d'une correspondance RDF, la structure du schéma (classe et propriété) du fichier RDF est décrite dans *./tarql/StrucutreTurtle.rdf*. Le fichier *./tarql/creation_gvri.bat* permet la création du fichier *gvri.ttl* contenant l'ensemble du schéma créé à l'aide des exécutions de lignes tarql/sparql et de la structure du schéma *StructureTurtle.rdf*.

Le fichier RDF constitué, est ensuite utilisé pour créer un TDB (base de données de triplets, moyen de Jena pour gérer les schémas) persistent avec *Fuseki*.
L'ensemble des commandes de cette "préparation" a été réuni dans le fichier *./init.bat* permettant l'initialisation de la base du projet (étape utile que pour changement de données, etc…, cf *./ReadMe.md*)

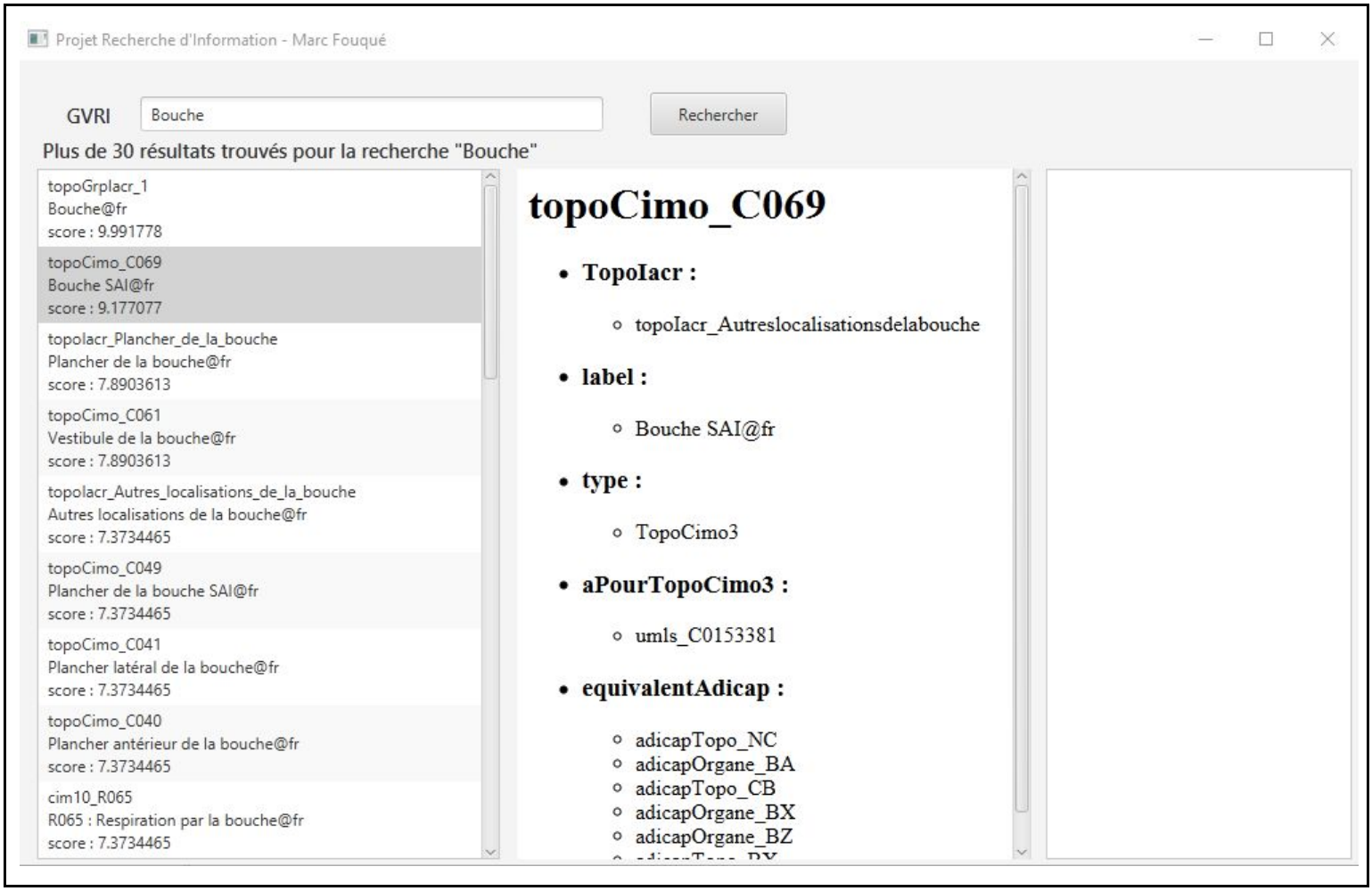
Indexation

L'indexation de notre base de connaissance se fait à l'aide d'un fichier assembleur (./source_java/src/main/resources/base/*assembleur.ttl*) regroupant l'ensemble des paramètres d'indexation et de recherche du projet. Ce fichier assembleur décrit différentes entités du projet (TDB, dataset, indexeur et analyseur) dont *indexLucene* décrivant les paramètres d'indexation via Lucene.

Cette indexation prend en compte 4 champs, les champs Nom, Prénom, DateNaissance et Labels, et est réalisée à l'aide de l'analyseur standard de Lucène (*StandardAnalyser*). Le champs Label utilisé avec la majorité des entités du modèle, est annoté d'un format de langue permettant un support multilingue du StandarAnalyser, avec l'utilisation de *FrenchAnalyser* (*lang=fr*) pour l'ensemble des ressources exceptée pour les labels des groupe IACR morphologique utilisant *EnglishAnalyser* (*lang=en*). DateNaissance composé exclusivement de chaîne de caractère représentant des dates, un analyseur spécifique est utilisé, embarquant le *LetterTokenizer* de Lucène permettant d'extraire les chaines de caractères et donc d'extraire les mois permettant leur recherche à travers le moteur.
Une fois l'index créé, les préparatifs préalables à l'utilisation du moteur de recherche sont effectués (cd *./ReadMe.md*).

Recherche

Au lancement de l'application, le programme récupère le TDB, et crée un *TextDataset* (*Jena*) à partir dudit TDB et de l'index grâce au fichier assembleur.
Le programme initie une interface graphique séparée en 4 partie: une barre de recherche, une liste de résultat, un affichage du résultat choisi (informations liées à la ressources sélectionnée) et une liste de résultats de codes potentielles (d'après un matching d'expressions régulières).



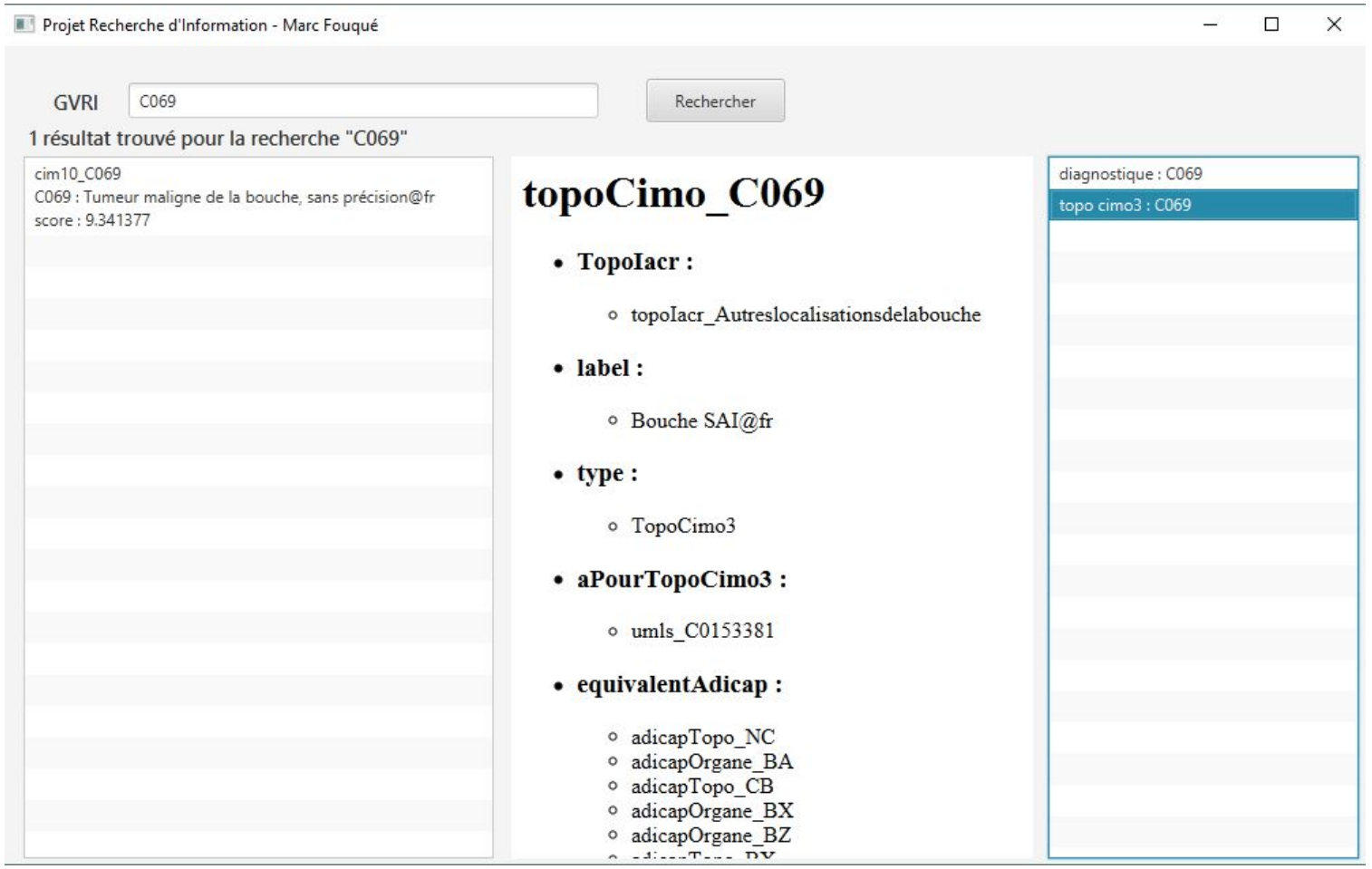


Figure 1 : Recherches exemples

La recherche s'effectue à l'aide d' une requête *Sparql* (*Select*) permettant grâce à l'extension *Jena-text* de faire une recherche sur l'index créé. Une recherche exact est lancée (pas de *fuzzy query*), le parseur de la recherche est le même que celui de l'indexation, la recherche permet de récupérer l'ensemble des sujets (dans un paradigme de triplets, *subject-predicate-object*), le score rattaché à chacun d'eux et la valeur matchée par la requête.

Conclusion / Discussion

Le projet et son application permettent une recherche basique sur les données du PMSI.
Cette recherche ne possède pas de procédés permettant une expansion sémantique, les bases pour la permettre sont présentes mais l'appliquer au sein d'une seule requête sparql malgré l'aide de l'*union* ne s'est pas avéré fructueux. L'ajout de cette recherche sémantique associée à une association et une pondération des scores retrouvés auraient pu permettre une finesse dans les résultats et leur hiérarchisation.
La recherche souffre d'une non indexation du champs *uri* qui aurait permit une recherche exacte sur l'intitulé de la ressources (bien que le label est censé faire office d'intitulé) et d'une non prise en compte des caractères accentués lors de l'analyse, l'utilisation d'un filtre tel que *ISOLatin1AccentFilter* aurait permis de gommer les différences entre les caractères

accentués et leurs pendants non accentués, cette manipulation aurait été faite à travers un analyseur configurable dans le fichier assembleur.
La nature du fichier RDF perds de sa valeur par le fait que le document ne contient pas de lien vers des ressources externes, les codes cimO3 auraient par exemple pu etre lié à la cim10 en faisant référence à l'icd 10 (http://purl.bioontology.org/ontology/ICD10/... au lieu de https://inf203.isped.fr/gvri/instances#topoCimo_...) mais sémantiquement cette liaison aurait pu être maladroite. De même les nom et prénom n'ont pas été décrit avec la ressource *FOAF*, par le non besoin (bien au contraire) de lié les patients à des ressources extérieures les définissant aussi.

Un approfondissement des possibilités offertes par le domaines des ontologies dans le cadre de la recherche d'information est donc nécessaire pour une évolution de ce projet.