RAPPORT PROJET INF203

I. Objectif du projet

Le projet a pour but de concevoir un moteur de recherche en se basant sur le thésaurus et les données exploités lors du projet d'intégration de données.

II. Outils utilisés

Ce travail a été réalisé sous Eclipse IDE avec le langage de programmation Java 1.8.
Le gestionnaire de dépendances Maven a été utilisé pour intégrer les librairies Apache Lucene 2.3.0 et OpenNLP au projet. L'utilisation de Neo4j, qui est un système de gestion de base de données basé sur les graphes, a également été nécessaire. Le framework JavaFx a permis de développer l'interface utilisateur.

III. Algorithme d'expansion de requête

L'expansion de requête consiste à reformuler une requête soumise par l'utilisateur en y rajouter des termes pour améliorer la performance de la recherche d'information. Il existe un bon nombre de techniques d'expansion parmi lesquelles :

- chercher les synonymes des termes de la requête et les ajouter
- chercher les termes sémantiquement relié (hyperonyme, hyponyme, méronyme …)
- chercher les autres variations morphologiques des termes
- corriger les erreurs de syntaxe

Pour ce projet le choix a été porté sur deux méthodes :

- la correction des termes en se basant sur un vocabulaire et un calcul de similarité
- la recherche de termes sémantiquement reliés

IV. Stratégie adoptée

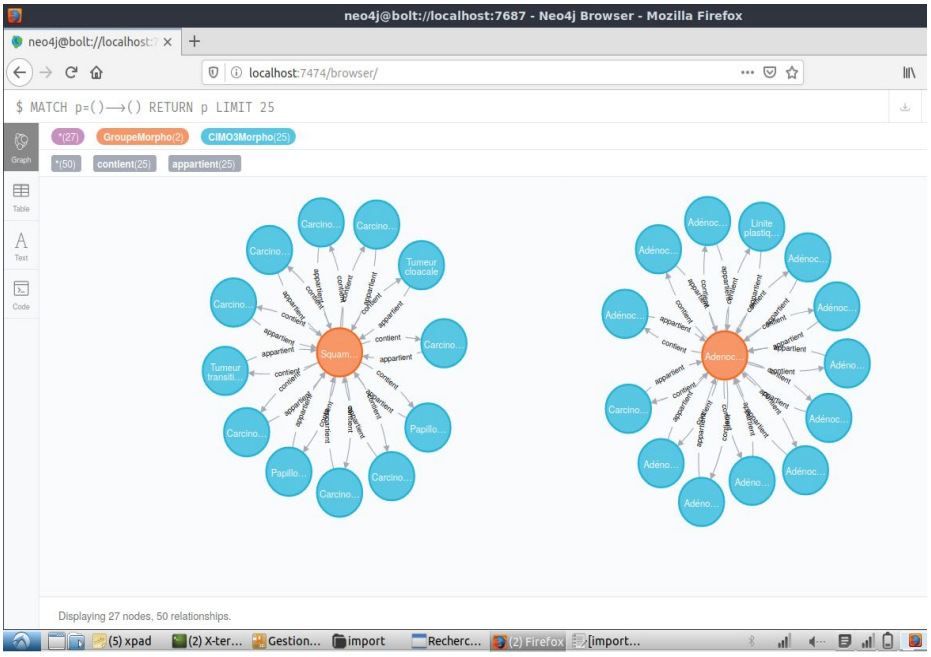
1. Confection d'une collection

Pour mettre en place la collection qui sera utilisée par l'algorithme de correction, une tokenisation des fichiers décrivant les différents cancers et les groupes morphologiques auxquels ils appartiennent et ceux décrivant les différentes localisations des cancers et leurs groupes topographiques a été faite. Les termes issus de cette tokenisation sont ensuite enregistrés dans les fichiers "vocabulaire.txt" contenant l'ensemble des termes, "morpho.txt", "topo.txt".

2. Création des relations avec Neo4j

Neo4j permet à partir du langage de requêtes Cypher d'importer des fichiers, notamment CSV, de créer des noeuds à partir des entités des tables CSV et des relations.
Exemple :

- Importation CSV et création de noeuds :
  LOAD CSV WITH HEADERS FROM "file:///CIMO3_TOPO.csv" AS row
  CREATE (:CIMO3Topo {codeTopoCIMO3: row.CodeTopoCIMO3, libelleTopoCIMO3: row.LibelleTopoCIMO3, codeGroupeTopo: row.CodeGroupeTopo});
- Création de relations CIMO3Topo - GroupeTopo :
  match (c:CIMO3Topo) match(g:GroupeTopo) where c.codeGroupeTopo in g.codeGroupeTopo merge(g)<-[:appartient]-(c)



Graphe obtenu sous Neo4j

3. Relation sémantique entre les termes

La définition de la relation sémantique entre les termes s'est fait grâce à un fichier RDF. Il a d'abord fallu définir les relations au niveau d'un graphe sur Neo4j en utilisant les fichiers CSV fournis dans le cadre du projet IME202. Une fois ces fichiers chargés, il est possible de définir les relations entre les entités : "appartient" pour dire qu'une entité appartient à une autre" et "contient" pour dire qu'une entité en contient une autre. Il s'agit de relations d'hyperonymie - hyponymie (morphologie) et de méronymie-holonymie (topographie)
Exemple des cancers : Adénocarcinome basocellulaire "appartient" Adenocarcinomas.
Exemple des organes : Bouche "contient" Palais dur.
Neo4j a permis ensuite de générer un fichier RDF ('AnatomieEtCancer.rdf') à partir du graphe :
:POST /rdf/cypher
{ "cypher" : "MATCH path = (n:Order { orderID : '10785'})-[:ORDERS]->()-[:PART_OF]->(:Category { categoryName : 'Beverages'}) RETURN path " , "format": "RDF/XML" , "mappedElemsOnly" : true }

4. Technique expansion

La soumission par l'utilisateur d'une chaîne de caractères déclenche une série d'actions :

- ➔ "split" de la chaîne de caractères et récupération des termes un à un
- ➔ comparaison de chaque terme avec le vocabulaire et calcul de la similarité avec l'algorithme "EditDistance"
  - ◆ si résultat < 3 : choisir le terme du vocabulaire comme nouveau terme de recherche
  - ◆ sinon : conserver le terme soumis par l'utilisateur
- ➔ recherche de chaque terme dans les collections "morpho.txt" et "topo.txt"
  - ◆ si le terme est retrouvé dans "morpho.txt", le groupe morphologique auquel il appartient est recherché et rajouté comme terme de recherche
  - ◆ si le terme est retrouvé dans "topo.txt" le groupe topographique auquel il appartient est recherché et rajouté comme terme de recherche
- ➔ tous ces nouveaux termes sont enregistrés dans une liste de String qui va définir la requête étendue

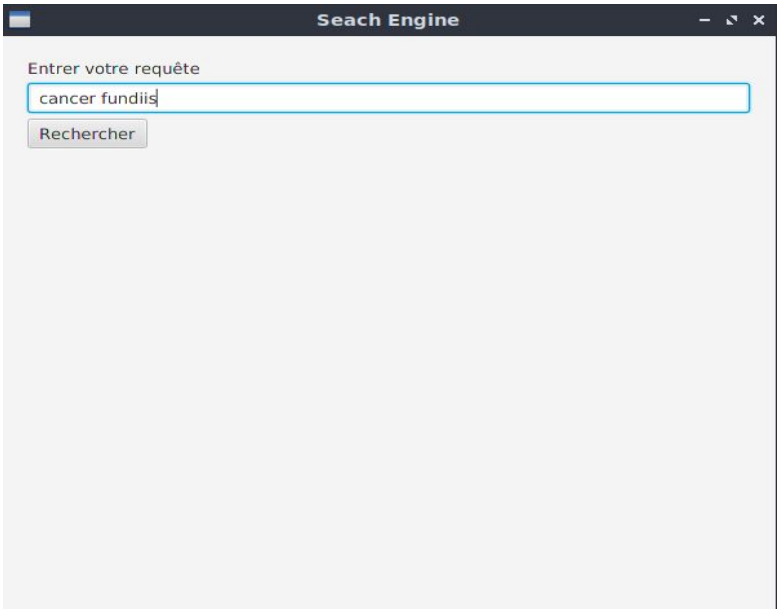
5. Indexation des documents et recherche

Un corpus contenant des documents abordant différents thèmes (anatomie organes et systèmes, cancers) a été constitué et est utilisé pour l'indexation et la recherche avec Apache Lucene.

La recherche renvoie une liste de documents classés par score de pertinence ainsi que les scores. Les documents sont classés en utilisant la classe TopDocs d'Apache Lucene qui renvoie les 'N' meilleurs résultats de la recherche.

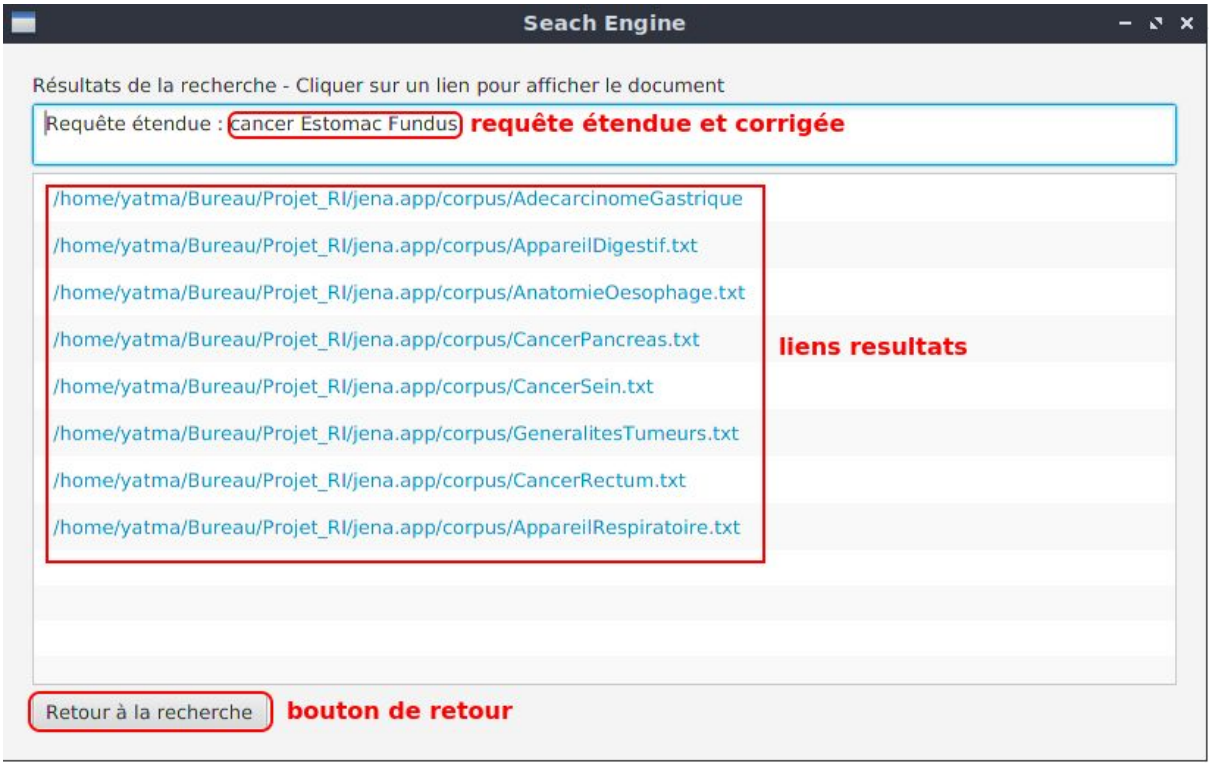
V. Guide d'utilisation

A l'exécution de l'application (fichier **Main.java** du package **siti.inf203.jena.javafx**), l'utilisateur est invité à renseigner les termes de sa requête dans la barre de recherche puis peut lancer la recherche avec le bouton 'Rechercher'.



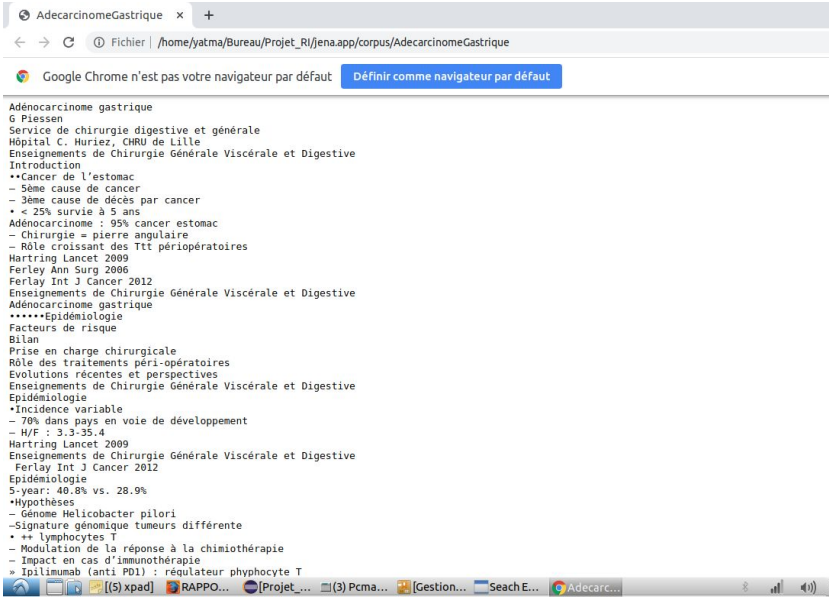
Menu principal

Les résultats pertinents par rapport à la recherche s'affichent par ordre de score décroissant et sous forme de liens.



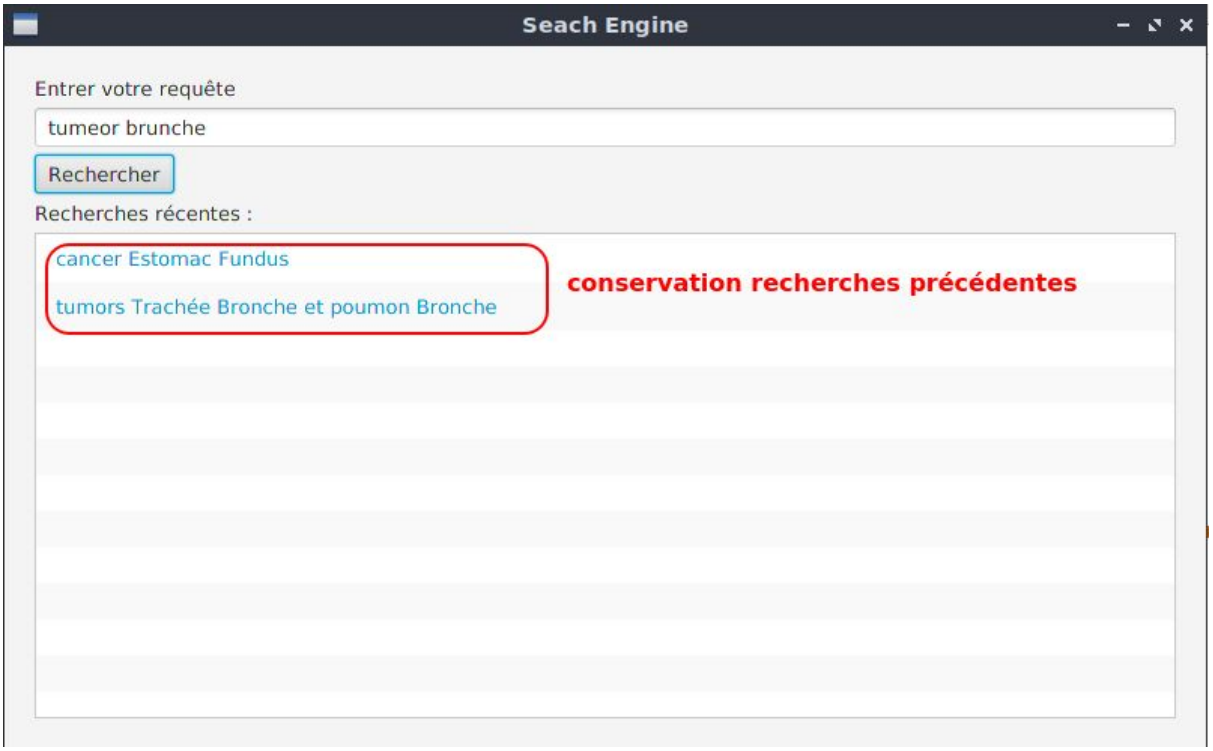
Résultats de la recherche

Il est ainsi possible de cliquer sur les différents liens pour afficher les documents retournés.



Affichage du document

En appuyant sur le bouton ‘Retour à la recherche’, une liste des requêtes précédentes s’affiche en plus dans le menu principal.



Retour au menu de recherche

VI. Discussion

Ce projet a permis d’utiliser la majorité des bases vues en cours : la syntaxe **RDF**, les requêtes **SPARQL**, indexation et rechercher avec Apache Lucene, calcul de **similarité** avec EditDistance (distance de **Levenshtein**) , l’indexation et la recherche de documents, les **relations sémantiques** (hyperonymie - hyponymie, méronymie-holonymie), le framework JavaFx, le gestionnaire de dépendances Maven entre autres.
Cependant le calcul de la similarité entre les termes avec “EditDistance” a des limites et peut parfois renvoyer des termes totalement différents d’un point de vue sémantique, mais cette limite est considérablement atténué en fixant un seuil au score obtenu.
Il aurait également été intéressant de rechercher des synonymes aux termes de recherche ou des termes fréquemment co-occurents pour améliorer encore plus la performance de ce moteur de recherche. Un package (**org.apache.lucene.wordnet**) utilisant les synonymes définis par WordNet dans une base de données prolog (fichier wn_s.pl) pour étendre les requêtes est disponible mais en version anglaise, ce qui limite le nombre de termes qu’on aurait pu être retrouvés en l’utilisant dans ce travail.