Rapport de projet INF 301

14/12/2023

Laura ARLES, Marwa BENAMOR, Lou CLAEYSSEN FABRIS

Introduction

Le projet a pour but de développer un moteur de recherche intuitif spécialisé dans l'exploration d'un corpus d'abstracts de littérature scientifique.

Scénario

Ce moteur de recherche doit permettre aux professionnels de santé de réaliser une recherche rapide dans un corpus d'articles scientifiques. Celle-ci doit retourner des articles pertinents à l'aide notamment d'une expansion de requête, et proposer des articles similaires afin d'approfondir le sujet.

Un scénario d'utilisation serait la recherche par un hématologue d'informations sur le traitement d'un type d'anémie : il peut requêter un corpus de résumés d'articles scientifiques portant sur l'anémie qui a été indexé selon 3 niveaux (mots simples, termes et concepts) afin de rechercher les articles les plus pertinents.

Structure

Le projet prend la forme d'une application Java Springboot qui expose des API Rest. Suivant la structure d'un projet Java Springboot, il comprend différents packages décrits ci-dessous :

- un package **controller** qui comprend deux Controller (CorpusController et SearcherController) avec leurs API respectives, reliés par une configuration CORS
- un package **service** comprenant les logiques de code de la création de corpus et de l'indexation d'une part dans CorpusService, et de la recherche dans les indexs d'autre part dans SearcherService. Ces services appellent des méthodes situées dans des packages spécifiques décrit ci-après.
- un package **domain** comprenant un DocumentDomain qui représente un résultat PubMed comprenant un identifiant, un titre, un résumé et un lien URL

Les autres packages comprennent des méthodes propres aux différentes fonctionnalités :

- un package **abstractExtractor** qui comprend les méthodes qui vont requêter la base PubMed et extrait les résultats pour les écrire dans un fichier en format texte avec les informations suivantes : identifiant de l'article, titre, résumé et lien
- un package **indexerSearcher** qui comprend les méthodes d'indexation et les méthodes de recherche pour chaque niveau (mot simple, terme et concept), ainsi qu'une méthode créant un CharArraySet à partir d'un fichier de "stopwords" en anglais qui est utilisé par le StandardAnalyzer

- un package **ontologyLoader** qui comprend les méthodes pour charger une ontologie existante en local, récupérer l'URI d'un concept à partir de son label, et récupérer les termes liés à un terme

Enfin, la racine des packages du projet comprend la classe exécutable **Application** et les paramètres configurables dans la classe **Parameters**.

Applications

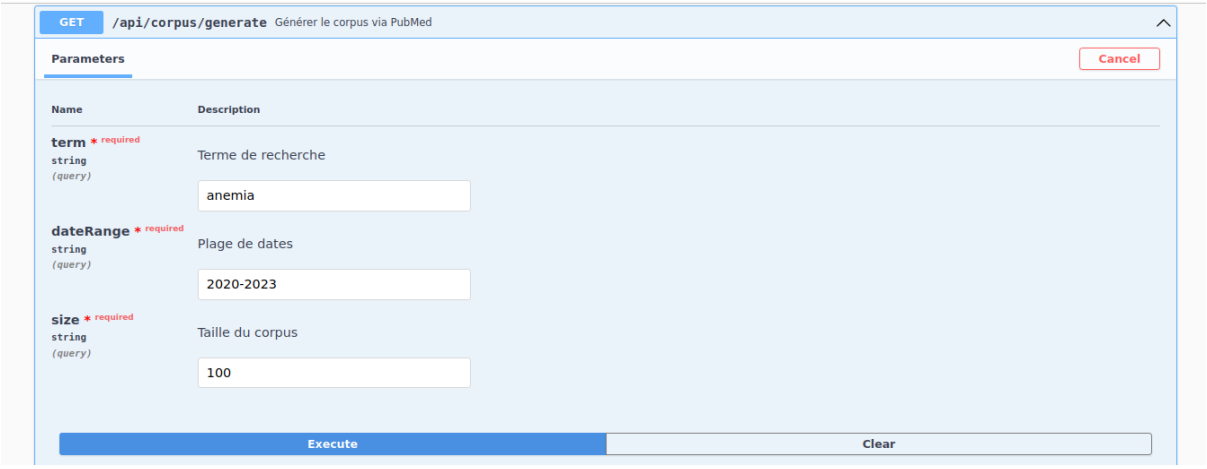
Les API Rest exposées sont les suivantes :

`CorpusController`

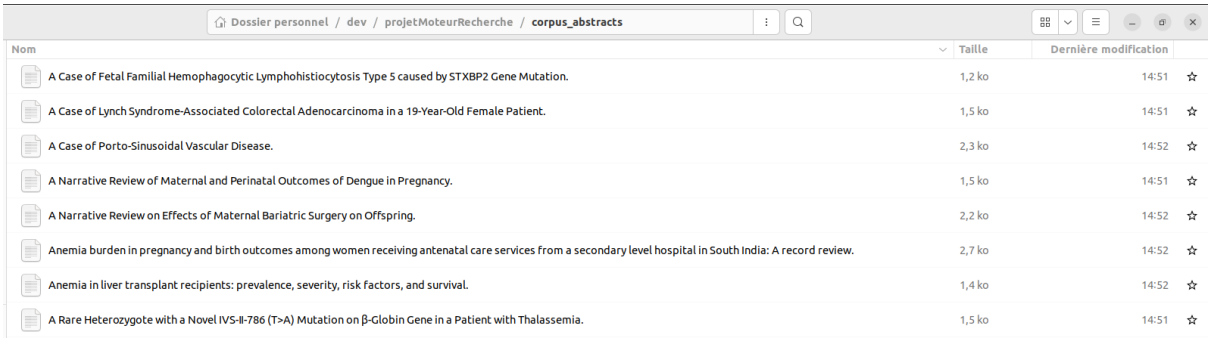
`generate` : création d'un corpus d'abstracts à partir d'une requête PubMed

La fonction `generate` permet d'extraire les résultats de la page HTML de résultats PubMed en fonction d'une requête spécifique.

Les paramètres d'entrée incluent le terme de recherche (`term`), la plage de dates (`dateRange`), et le nombre de résultats souhaité (`size`).



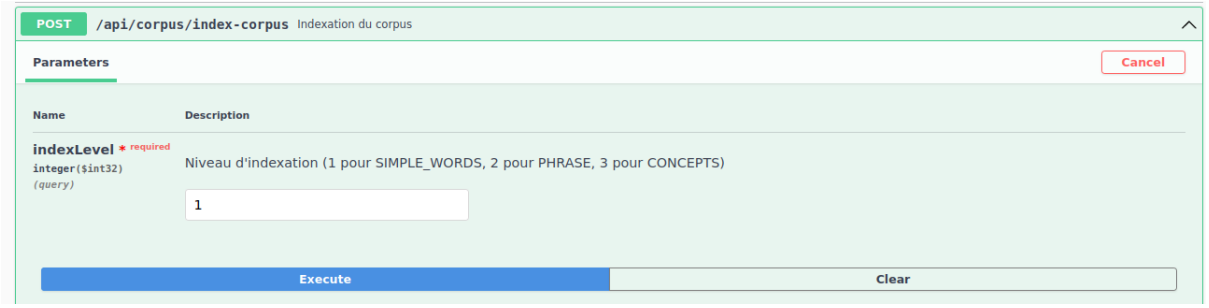
Les résultats sont ensuite stockés dans des fichiers texte (.txt) pour former le corpus d'abstracts (cf. capture d'écran ci-dessous).



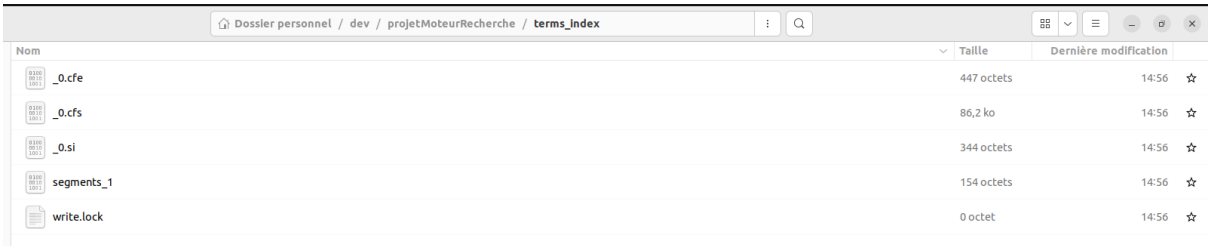
`index-corpus` : indexation d'un corpus de documents

La fonction `index-corpus` permet d'indexer le corpus créé précédemment selon trois niveaux : par mots simples, par termes, et par concepts.

Le paramètre d'entrée correspond au niveau d'indexation souhaité : 1 pour l'indexation par mots simples, 2 pour l'indexation par termes et 3 pour l'indexation par concepts.



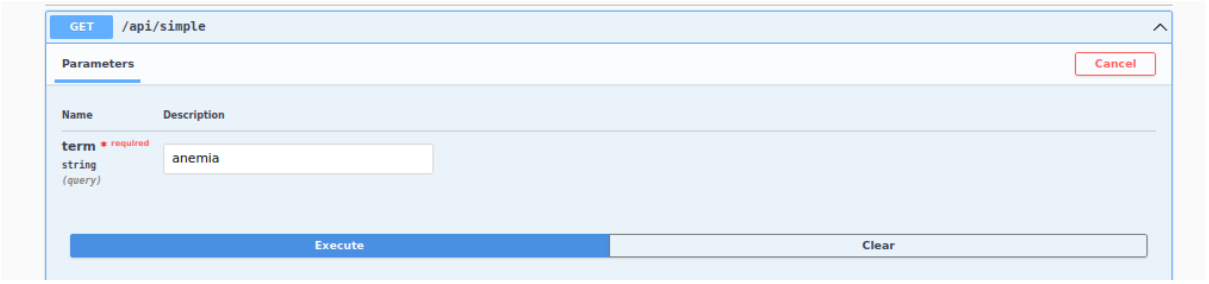
Ci-dessous un exemple d'index créé suite à une indexation par termes.



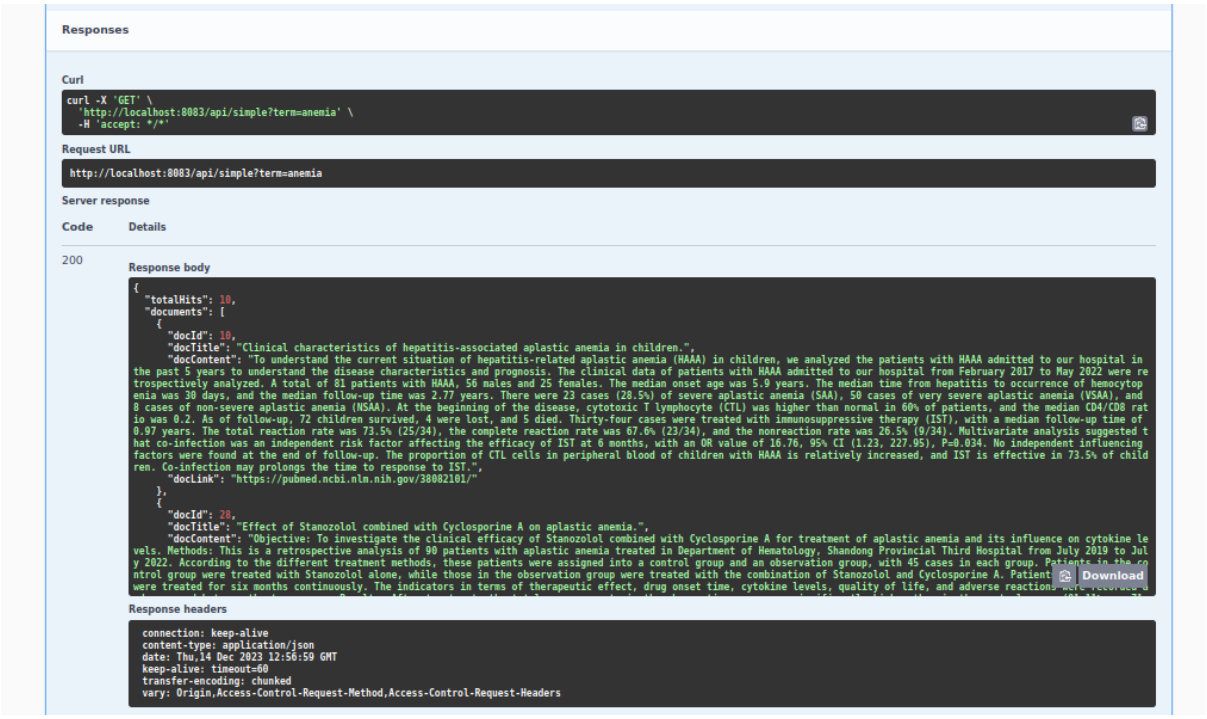
`SearcherController`

`simple` : recherche par mots simples dans le corpus de documents

La fonction `simple` réalise une recherche par mots simples dans le corpus de documents. Elle prend pour paramètre d'entrée un mot simple.



Elle retourne le nombre total de résultats (`totalHits`) ainsi qu'une liste d'informations sur les documents correspondants, comprenant l'identifiant (`docId`), le titre (`docTitle`), le résumé (`docContent`), et le lien URL sur PubMed (`docLink`) : voir l'exemple ci-dessous.

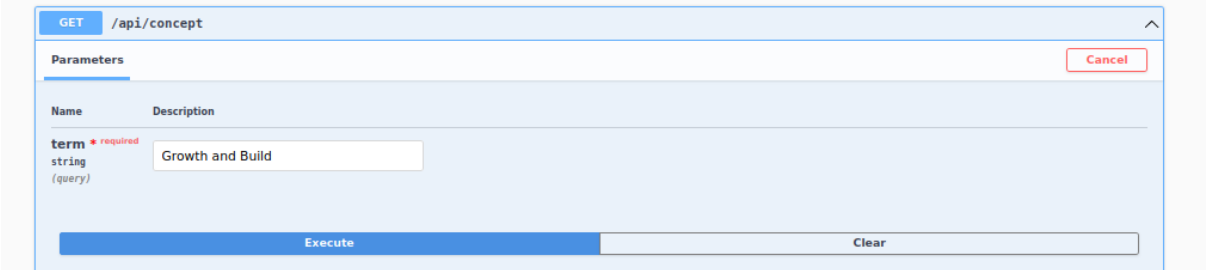


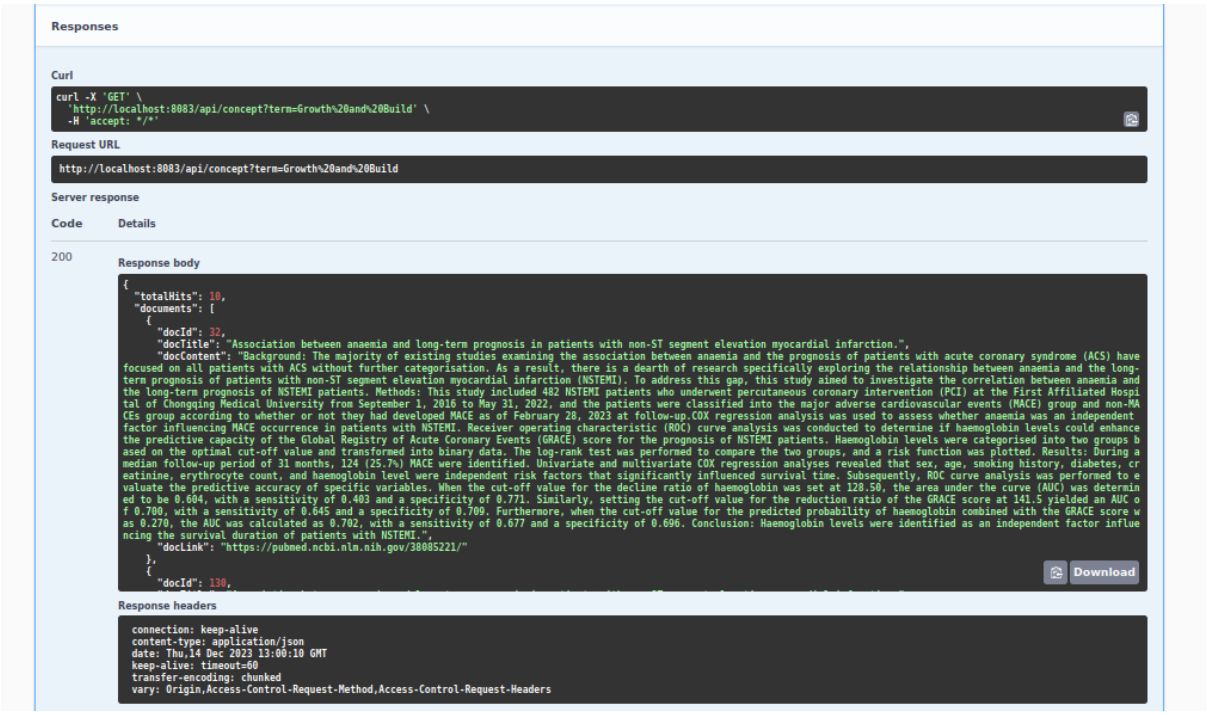
`phrase` : recherche par termes dans le corpus de documents

La fonction `phrase` effectue une recherche par termes dans le corpus de documents. Elle prend en paramètre d'entrée un terme, et elle retourne le nombre total de résultats (`totalHits`) ainsi qu'une liste d'informations sur les documents correspondants, comprenant l'identifiant (`docId`), le titre (`docTitle`), le résumé (`docContent`), et le lien URL sur PubMed (`docLink`), tel que montrée pour l'API `simple`.

`concept` : recherche par concepts dans le corpus de documents

La fonction `concept` permet une recherche par concepts dans le corpus de documents. Elle prend en paramètre d'entrée un concept, et elle retourne le nombre total de résultats (`totalHits`) ainsi qu'une liste d'informations sur les documents correspondants, comprenant l'identifiant (`docId`), le titre (`docTitle`), le résumé (`docContent`), et le lien URL sur PubMed (`docLink`), cf exemple ci-dessous.





Guide d'utilisation

Le guide d'utilisation est disponible dans le `README` à la racine du projet.

Difficultés rencontrées

Le projet ne comprend pas de techniques de proposition d'articles similaires, ni de système permettant d'indiquer la performance du système.
L'indexation par termes se fait à l'aide du WhitespaceAnalyzer plutôt que les concepts compris dans l'ontologie.