h e g

Haute école de gestion
Genève

Extraction des concepts biomédicaux des essais cliniques en utilisant le traitement automatique du langage naturel

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

David VICENTE ALVAREZ

Conseiller au travail de Bachelor :

Douglas TEODORO

Genève, le 08.08.19

Haute École de Gestion de Genève (HEG-GE)

Filière informatique de gestion

Hes·so GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre < … >.

L'étudiant a envoyé ce document par email à l'adresse remise par son conseiller au travail de Bachelor pour analyse par le logiciel de détection de plagiat URKUND, selon la procédure détaillée à l'URL suivante : http://www.urkund.fr/student_gorsahar.asp.

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul< e > le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 8 août 2019

David Vicente Alvarez

Remerciements

J’aimerais remercier mon directeur de mémoire, M. Teodoro Douglas pour m’avoir aidé et conseillé tout au long de mon travail. J’aimerais également remercier M. Humbert Jérôme pour m’avoir donné de son temps afin de réaliser mon travail.

J’aimerais également remercier tous mes relecteurs, sans qui le rendu/la finalisation de mon travail aurait été très compliqué/e)

Résumé

Les essais cliniques sont des études scientifiques qui permettent d'évaluer l'efficacité de certains médicaments, drogues ou nouvelles méthodes médicales, ainsi que leurs effets secondaires. La plupart du temps, ils se concluent sur un échec. Avoir un outil qui permet d'évaluer le risque d'échec est donc crucial.

Ces essais cliniques sont écrits en texte libre, ce qui rend le traitement automatique standard par ordinateur presque impossible. C'est pourquoi l'analyse du langage naturel est utilisée. Le but de ce travail est de créer une base de données qui contient les essais cliniques et les concepts qu'il est possible d'en extraire, pour permettre un traitement automatique dans le futur.

Table des matières

Liste des tableaux

Liste des figures

1. Introduction

1.1 Contexte

Des essais cliniques (EC) sont effectués chez des volontaires humains pour déterminer si un traitement expérimental fonctionne bien et sans danger pour obtenir une autorisation de mise sur le marché. Ils constituent la partie la plus longue et la plus coûteuse du développement de médicaments. En réalité, plus de 90% des EC échouent, ce qui impose une charge financière aux entreprises et expose les volontaires à des risques.

Indépendamment de leur résultat, les EC sont légalement déposés dans des bibliothèques numériques, telles que la base de données européenne sur les essais cliniques - EudraCT et le laboratoire américain ClinicalTrials.gov.

En raison de la complexité, de la taille et de la variété des données stockées dans ces bibliothèques numériques, les analyses manuelles ne permettent pas d'extraire des informations pertinentes, telles que les causes des échecs des EC. Comme les enregistrements EC sont normalement semi-structurés et contiennent des sections de texte libre, les méthodes classiques assistées par ordinateur ne sont pas non plus adaptées à cet usage.

2. But

Le but du projet est de transformer du texte en écriture libre, en données structurées et de fournir une base de données remplie de ces nouveaux fichiers structurés pour réaliser une future analyse.

Pour cela, l'analyse du langage naturel est utilisée. Chaque block de texte est coupé en phrases, puis les concepts sont extraits de chaque phrase et finalement réinsérés dans le document.

3. Méthodologie

3.1 Logiciels et outils utilisés

Les logiciels et outils utilisés sont :

- Programmation
    - Java SE/EE
        - Langage de programmation
    - Eclipse
        - Environnement de développement de code
    - Maven
        - Gestionnaire des dépendances et de libraires
- Base de données
    - Elasticsearch
        - Base de données principale
    - Kibana
        - Gestionnaire de Elasticsearch et visualisation de données
    - Logstash
        - Insertion en masse de données dans Elasticsearch
- Serveurs
    - Apache Tomcat 9
        - Serveur web principale exécutant le code
    - MetaMap
        - Serveur d'extraction de concepts

3.1.1 Java

Java est un langage de programmation ainsi qu'une plateforme qui a été créé par Sun Microsystems en 1995. Il peut être utilisé dans de nombreux environnements et est stable.

Dans ce projet, la presque totalité du code est écrite en Java. Le serveur possède un moteur Java (Java Runtime Environment) et exécute le code pour la transformation des fichiers.

La version de la JRE utilisée est Java 8.

3.1.1.1 Librairies Java utilisées

Les librairies principales utilisées dans ce projet sont :

- Jersey
- stanford-corenlp
- json-path
- metamap-api-2.0
- elasti httpclient
- elasticsearch-rest-high-level-client

3.1.1.1.1 Jersey

Il s'agit d'une série de librairies qui permet de lier du code Java à des URI pour que le code soit exécuté quand un utilisateur appelle le serveur avec l'URI. Il permet aussi d'injecter des variables web dans le code pour avoir entièrement le contrôle du comportement du serveur web. Il transforme aussi automatiquement des représentations aux formats xml et json vers des objets Java et inversement.

Cette librairie me permet de réaliser l'implémentation d'un service REST[1] et de gérer les protocoles http.

3.1.1.1.2 stanford-corenlp

Cette librairie fait du traitement automatique du langage naturel. Il permet notamment de séparer un bloc de texte en phrases. Il fait également de l'étiquetage morpho-syntaxique (ou « *part of speech tagging* ») qui associe à chaque mot son rôle grammatical.

C'est avec cette librairie que les phrases sont séparées et filtrées pour extraire les concepts.

[1] Representational State Transfer, voir à la page 18

3.1.1.1.3 json-path

Cette librairie permet d'analyser une chaîne de charactères et de la convertir en un objet json. Il me permet également de faire des requêtes de recherche pour trouver des nœuds. C'est ainsi que tous les nœuds contenant du texte sont recherchés.

3.1.1.1.4 metamap-api-2.0

Cette librairie permet d'envoyer une chaîne de caractères au serveur d'extraction de concepts MetaMap. Elle fournit une interface entre Java et le serveur.

3.1.1.1.5 elasticsearch-rest-high-level-client

Cette librairie permet de faire toutes les opérations de CRUD[2] sur Elasticsearch

3.1.1.1.6 httpclient

Une librairie conçue par la fondation Apache qui permet de faire des requêtes http en Java.

[2] Create, Retrieve, Update and Delete. Les fonctions de base des données persistantes

3.1.2 Maven

Le projet utilise Apache Maven pour la gestion des librairies et des dépendances. Maven est utilisé pour rechercher, installer et mettre à jour des jars[3], ce qui rend la documentation et la maintenance plus facile. Chaque librairie est explicitement décrite et stockée dans une librairie centrale. Si quelqu'un veut reprendre le code, il n'a qu'à réutiliser le code source ainsi que le fichier qui décrit toutes les dépendances que Maven installera automatiquement.

3.1.3 Base de données

La base de données utilisée est Elasticsearch 7.1.1. C'est un moteur de recherche RESTful qui est orienté document. Elasticsearch est notamment connu pour faire des recherches très flexibles et rapides, ce qui la rend idéale pour l'analyse et la comparaison de concepts.

Kibana et Logstash ont été utilisés en complément lors de l'insertion de données pour faciliter l'insertion et visualiser la progression. Kibana permet également de visualiser des données.

3.1.4 Serveurs

Le serveur web choisit est Apache Tomcat 9. C'est un serveur open source qui est le résultat d'une collaboration entre développeurs. Il implémente des technologies Java et est conçu pour exécuter du code Java.

MetaMap est un serveur conçu par Dr. Alan (Lan) Aronson au *National Library of Medicine*. Ce serveur analyse du texte et découvre des concepts biomédicaux qui correspondent au standard UMLS. La version utilisée est celle de 2018.

3.1.5 UMLS

Unified Medical Language System est un standard de terminologie biomédicale qui spécifie et hiérarchise des termes médicaux et des concepts. UMLS est utilisé pour transmettre des concepts entre différents acteurs ou, comme ici, reconnaître des termes dans du texte. C'est le standard utilisé par MetaMap pour exprimer les concepts trouvés.

[3] Java archive. Un fichier compressé contenant du code java compilé

3.2 Données de départ

Les données sont des essais cliniques qui ont été téléchargés à l'adresse suivante https://clinicaltrials.gov/. Ce sont des fichiers XML (Extended Markup Language) qui suivent le standard émis par le National Library of Medicine, National Institutes of Health[4]. Ce standard est un fichier qui régit la grammaire du fichier. Un exemple type d'un essai clinique se trouve à l'annexe 1.

Le nombre d'essais cliniques téléchargés est de 301'104 répartis sur 390 dossiers (le nombre de fichiers est trop grand pour qu'ils soient stockés dans un seul dossier).

3.2.1 Insertion dans une base de données

Les essais cliniques ont d'abord été stockés dans Elasticsearch. Ils sont convertis et stockés sous le format json avec l'aide de Logstash. Cela prendrait beaucoup trop de temps d'insérer manuellement les fichiers, car Logstash effectue une insertion en batch.

2 Indexes ont été créés au début dans Elasticsearch :

- risklick_fulldev : contient tous les fichiers téléchargés
- risklick_devtest : est un sous-ensemble de *riskclick_fulldev* mais il ne contient que 751 fichiers. Ce jeu de données sert au développement et aux tests.

Toutes les données stockées dans Elasticsearch le sont au format json. Il s'agit d'un des formats par défaut qu'utilise Elasticsearch.

[4] La grammaire utilisée se trouve à cette adresse : https://clinicaltrials.gov/ct2/html/images/info/public.xsd

3.3 Processus de transformation

Le texte suivant explique quelles sont les étapes qui ont permis de réaliser la base de données des essais cliniques transformés.

3.3.1 Lecture du fichier

Le traitement commence par la recherche et la lecture du fichier. Le serveur reçoit l'identifiant unique de l'essai clinique (exemple NCT0036022), puis le serveur demande le fichier correspondant chez Elasticsearch. Il récupère le fichier sous forme de chaîne de caractères. Cette chaîne de caractères est ensuite transformée en un objet json grâce à la librairie json-path.

3.3.2 Recherche des nœuds textuels

L'étape suivante consiste à chercher les textes à traiter. Chaque texte se trouve dans une feuille (un objet json peut être vu sous la forme d'un arbre). C'est donc grâce à la librairie json-path que cela est réalisé. En effet, chaque essai clinique a le même standard, mais n'a pas la même structure (certains nœuds peuvent ne pas être présents). Il est donc impossible d'avoir une connaissance exacte de la structure d'un essai clinique à l'avance.

Pour trouver les textes à traiter, le serveur cherche toutes les feuilles qui ont le chemin d'accès contenant « content ». En, effet quand les fichiers xml ont été insérés dans Elasticsearch, la base de données a analysé les fichiers pour les convertir en json et en ajoutant des métadonnées, ce qui fait que chaque valeur, texte etc. est « content » dans Elasticsearch

Exemple de traduction

Figure 1 : Exemple d'entrée

```
<clinical_study>
    <brief_title>The Safety and Effectiveness of Different Doses of Vitamin C in HIV-
Infected Patients</brief_title>
</clinical_study>
```

Figure 2 : Résultat dans Elasticsearch

```
{
    "clinical_study":{
        "brief_title":[{
            "content":"The Safety and Effectiveness of Different Doses of Vitamin C in HIV-
Infected Patients"
        }]
    }
}
```

3.3.2.1 Paramètre « sections »

Une des exigences est de proposer un formatage partiel du document. Cela permet de ne pas traiter des parties du fichier qui ne sont pas pertinentes ou qui posent un problème avec MetaMap.

Plusieurs parties du document peuvent être spécifiées. Le serveur fonctionne comme suit : le serveur cherche toutes les feuilles (donc texte) dont le chemin d'accès contient le bloc recherché, plus « content » (exemple : transformer toutes les feuilles dont le chemin contient « brief_summary » et « content »).

Les noms suivants ont été choisis pour la création de la base de données :

- brief_title
- official_title
- textblock
- title
- sub_title
- description

Ces parties de documents ont été choisies, car elles contenaient la majorité des textes pertinents. Le reste du document ralentissait l'extraction de concepts car il renfermait des informations non pertinentes, comme des noms d'auteurs, des noms de lieux ou trop de mots différents sans corrélation.

3.3.3 Séparation des phrases

Une fois que chaque bloc de texte a été trouvé, ce même texte est envoyé à la librairie *stanford coreNLP*. Cette librairie analyse le texte et crée un nouvel objet en interne contenant les annotations. C'est grâce aux annotations que les phrases peuvent être extraites.

3.3.3.1 Part of speech tagging

Stanford coreNLP propose également du *part of speech tagging* (POS). Cette fonction analyse chaque mot de la phrase et détermine sa fonction grammaticale (exemple : adjectif, nom, etc.)[5].

Dans ce cas, le *part of speech tagging* est utilisé pour filtrer les mots, chaque mot de chaque phrase est pris et comparé, si le tag correspondant fait partie des tags « accepté » le mot est gardé dans la phrase, sinon il est supprimé.

Le filtrage se fait pour éviter d'encombrer le serveur d'extraction de concepts, seuls les noms et adjectifs sont envoyés au serveur, car le serveur est conçu pour traiter avec efficacité les noms et adjectifs. Le reste des mots ne fait qu'ajouter du bruit pour le serveur. Ainsi, on obtient un meilleur résultat.

Les tags choisis pour le filtrage sont :

- JJ : Adjective
- NN : NounSingularOrMass
- NNP : ProperNounSingular
- NNS : NounPlural
- NNPS : ProperNounPlural
- FW : ForeignWord
- JJS : AdjectiveSuperlative
- JJR : AdjectiveComparative
- MD : Modal

[5] La liste des tags peut être consulté à : https://www.eecis.udel.edu/~vijay/cis889/ie/pos-set.pdf

Exemple :

« This sentence is a complex sentence. » on obtient les tags suivants :

- [DT, NN, VBZ, DT, JJ, NN, .]

Après filtrage on obtient la phrase suivante :

- « sentence complex sentence »

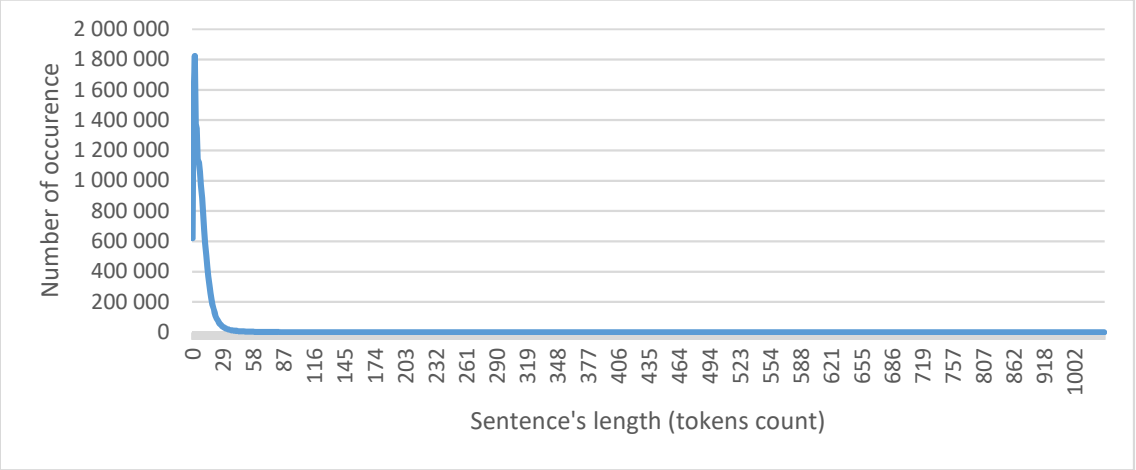
3.3.3.2 Limite des phrases

Stanford coreNLP délimite les phrases selon un algorithme entraîné. Comme tout algorithme, des erreurs peuvent se produire, ce qui fait que certaines phrases restent ensemble malgré tout. De plus, il est tout à fait possible que l'auteur de l'essai clinique ait écrit de très longues phrases.

D'après des tests empiriques, le serveur d'extraction a des performances quadratiques en fonction du nombre de mots envoyés au serveur, ce qui implique que plus la phrase est longue, plus elle est coûteuse. De plus, si la phrase est trop longue, le serveur se bloque sur cette phrase et n'avance plus du tout, ce qui demande un redémarrage du processus.

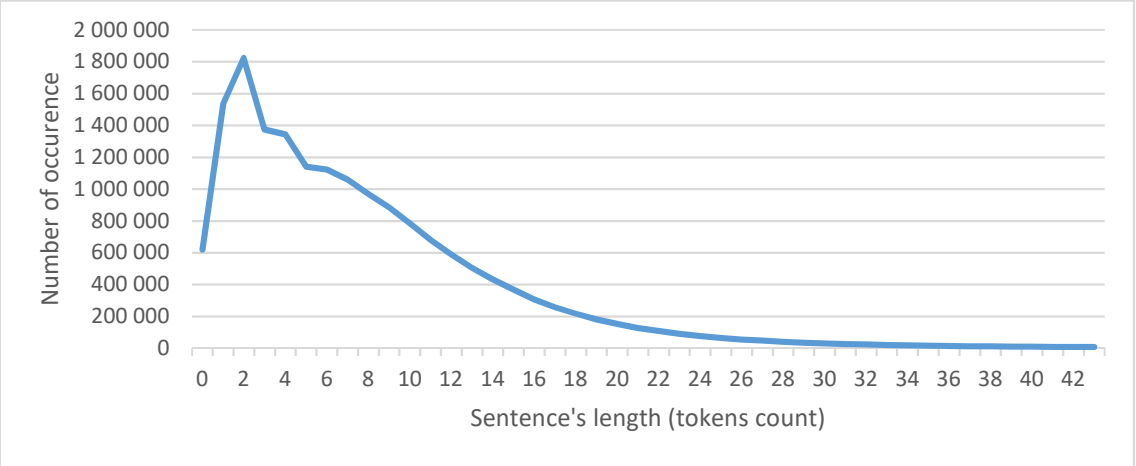
Pour avoir une idée plus précise du nombre de phrases jugées « extrêmes » présent, une classe a été créée qui compte le nombre de phrases pour chaque longueur, tout en répliquant le filtrage par section et par *Part of speech tagging* pour avoir une vision plus réaliste. Le résultat est le suivant :

Figure 3 : Nombre d'occurrences pour chaque longueur de phrase avec filtrage (POS tagging et les sections intéressantes) sur 301104 fichiers



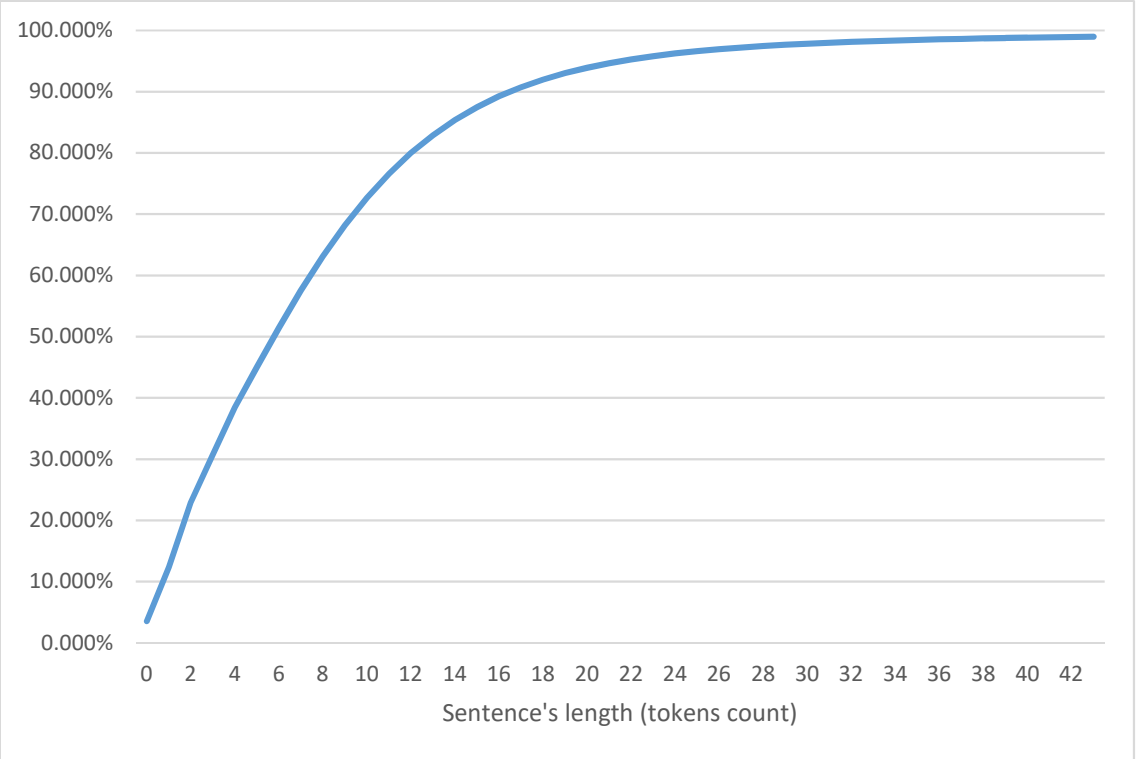
Comme on peut le constater, il existe des phrases très longues, mais elles ne forment pas la majorité des phrases.

Figure 4 : Nombre d'occurrences pour chaque longueur de phrase avec filtrage (POS tagging et les sections intéressantes) sur 301104 fichiers (99%)



Ce graphique est le même que le précédent, mais 1% des phrases les plus longues ont été enlevées.

Figure 5 : Répartition cumulative des longueurs de phrase (99%)



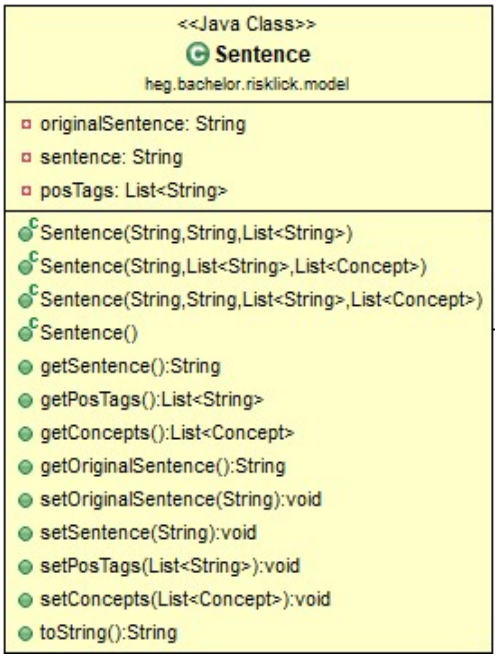
Ce graphique représente la répartition des phrases par leurs tailles. On peut constater que ¾ des phrases ont 11 mots ou moins et que 90% des phrases ont 17 mots ou moins.

Une limite a donc été fixée à 15 mots. Si la limite est dépassée, la phrase est juste abandonnée et n'est pas traitée. C'est un bon compromis, car cela ne supprime que 12,5% des phrases tout en préservant une bonne performance.

3.3.3.3 Création de l'objet « *Sentence* »

Une fois que toutes les phrases d'un bloc de texte ont été trouvées, l'objet *Sentence* est créé pour chaque phrase, la liste *Sentence* obtenue est retournée (un bloc de phrase peut avoir plusieurs phrases).

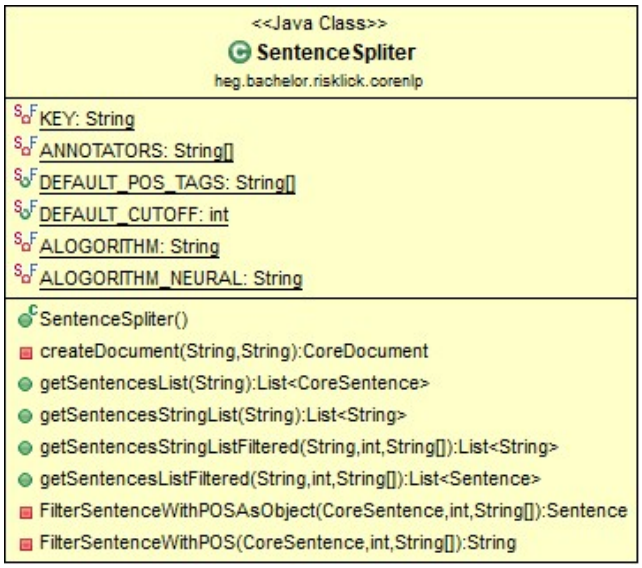
Figure 6 : La classe *Sentence*



Pour chaque phrase, l'objet stocke la phrase originale, tous les tags du *part of Speech tagging* et la phrase filtrée par le *part of speech tagging.* Si après le filtrage, la phrase dépasse les 15 mots, la phrase n'est pas ajoutée dans la liste retournée.

La classe qui implémente la division phrase est *SentenceSpliter* et la méthode qui retourne la liste de *Sentence* est *getSentencesListFiltered*.

Figure 7 : La classe *SentenceSpliter*



3.3.4 Extraction des concepts

Une fois que les phrases ont été délimitées et filtrées, il faut en extraire les concepts. Chaque phrase filtrée est donc envoyée au serveur d'extraction de concepts.

3.3.4.1 Ctakes

Ctakes est un moteur qui traite des fichiers développés par Apache. Il est très modulable et complexe. Il fonctionne sous la forme d'un pipeline avec 3 éléments, à savoir un *reader* qui s'occupe de lire les données, un ou plusieurs *annotateurs* et un *writer* pour choisir comment les résultats sont écrits.

Cependant, sa trop grande complexité et le manque de temps pour sa compréhension ont fait que cette solution a été abandonnée au profit de MetaMap.

3.3.4.2 MetaMap

MetaMap est le serveur d'extraction de concepts utilisé dans ce projet. Une fois la liste de *Sentence* obtenue, elle est envoyée à la classe qui s'occupe de les transmettre au serveur et de récupérer les données. Pour chaque objet *Sentence*, la classe prend le texte filtré et l'envoie au serveur, puis récupère les résultats et met à jour l'objet Sentence en ajoutant les concepts.

3.3.4.2.1 MetaMap API

Pour envoyer le texte au serveur, l'API (Application programming interface) offre une seule méthode : *processCitationsFromString*. Le serveur lit chaque chaîne de caractères transmise comme un seul document, donc une seule phrase peut être envoyée à la fois.

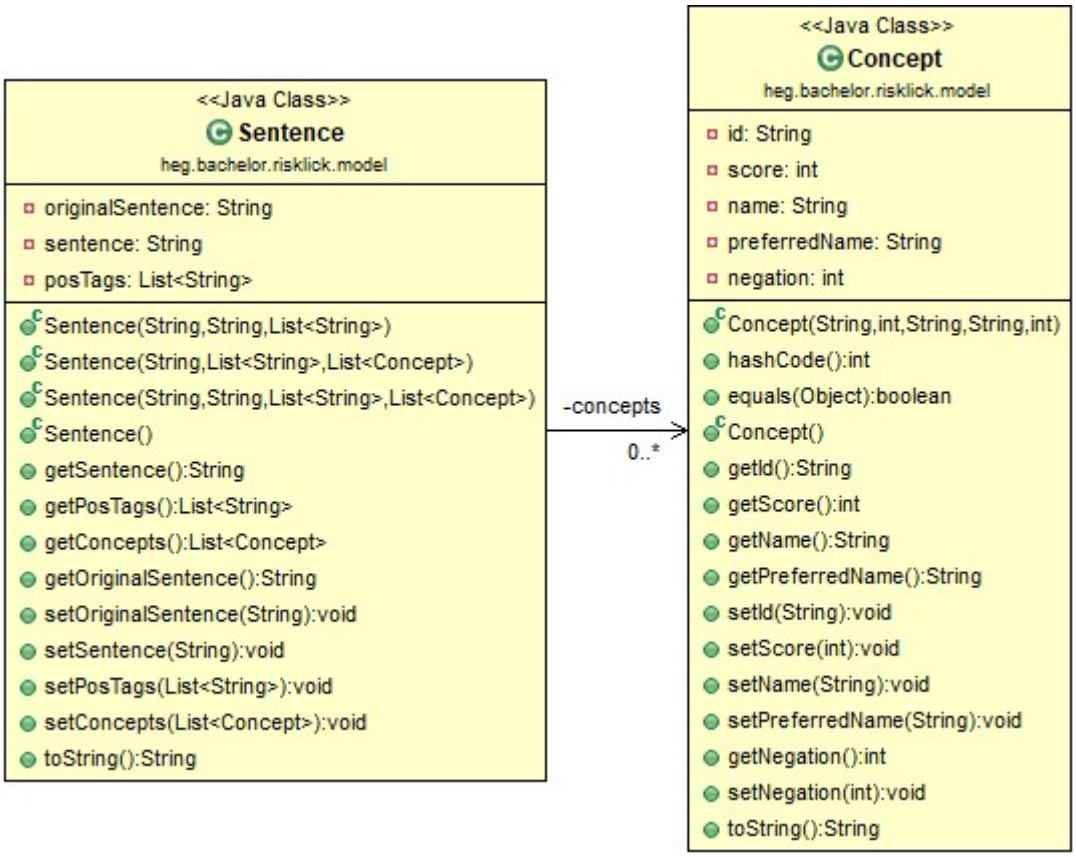
3.3.4.2.2 Objet retourné

L'objet retourné est assez complexe. Il contient beaucoup de listes imbriquées. MetaMap retourne une liste de *mapping*. Un *mapping* contient un score général de 0 à -1000 (-1000 était le meilleur) et une liste de concepts, qui ont chacun également un taux de fiabilité. MetaMap peut donc retourner plusieurs sets de concepts avec une fiabilité différente. Un exemple de tout ce qui peut être retiré de MetaMap se trouve en annexe.

Deux types de bases de données ont été créés. La première base de données contient tous les sets concepts trouvés, même les moins fiables, pour chaque phrase. Alors que la deuxième ne contient que le set de concept qui a obtenu le meilleur score pour chaque phrase.

3.3.4.2.3 Objet créé

Figure 8 : La classe *Concept* avec la classe *Sentence*



Pour chaque concept trouvé, les données suivantes sont enregistrées :

- L'identifient UMLS du concept
- Nom du concept
- Nom préféré du concept
- La négation (si le concept apparaît dans une négation)

Ensuite, l'objet *Concept* est créé contenant toutes ces informations. Puis pour chaque concept trouvé, il est inséré dans une liste. Finalement la liste de *concept* est ajoutée dans chaque objet *Sentence* envoyé initialement dans la classe.

3.3.5 Remplacement du nœud par l'objet

A partir de ce point, toutes les données nécessaires ont été trouvées, il faut remplacer le texte par la liste de *Sentence*. A cet effet, chaque bloc de texte est écrasé et remplacé par un tableau de *Sentence.*

La structure du document reste intacte, il n'y a que le texte qui est écrasé, les nœuds déjà existants restent présents. Uniquement des nœuds en lien avec l'objet *Sentence* sont créés et ajoutés. Cela permet de transformer un document tout en gardant une structure, mais également de conserver une trace de l'emplacement du texte original pour des analyse futures.

Voici un Extrait de document qui n'a pas encore été traité :

Figure 9 : Extrait d'un document non-traité


```
{
   "brief_title":[
    {
        "content":"The Safety and Effectiveness of Different Doses of Vitamin
C in HIV-Infected Patients"
        }
    ]
}
```


Avec cette entrée, le texte « The Safety and Effectiveness of Different Doses of Vitamin C in HIV-Infected Patients » est extrait, puis divisé en phrases (même si dans ce cas il n'y a qu'une phrase), pour ensuite être envoyé à MetaMap et, finalement, le texte va être écrasé et être remplacé par le résultat suivant :

Figure 10 : Résultat de l'exemple après transformation


```
{
  "brief_title": [{
    "content": [{
      "sentence": " Safety  Effectiveness  Different  Doses  Vitamin  C  HIV-
Infected  Patients",
      "originalSentence": "The Safety and Effectiveness of Different Doses of
Vitamin C in HIV-Infected Patients",
      "concepts": [
        {
          "score": -874,
          "negation": 0,
          "name": "Safety-patient",
          "id": "C1113679",
          "preferredName": "patient safety"
        },{
          "score": -624,
          "negation": 0,
          "name": "Effectiveness",
          "id": "C1280519",
          "preferredName": "Effectiveness"
        },{
          "score": -624,
          "negation": 0,
          "name": "Different",
          "id": "C1705242",
          "preferredName": "Different"
          },{
            "score": -624,
            "negation": 0,
            "name": "Doses",
            "id": "C0178602",
            "preferredName": "Dosage"
          },{
            "score": -641,
            "negation": 0,
            "name": "VITAMIN C",
            "id": "C0003968",
            "preferredName": "Ascorbic Acid"
          },{
            "score": -624,
            "negation": 0,
            "name": "HIV",
            "id": "C0019682",
            "preferredName": "HIV"
          },{
            "score": -624,
            "negation": 0,
            "name": "Infected",
            "id": "C0439663",
            "preferredName": "Infected"
          }
            ],
            "posTags":
["DT","NN","CC","NN","IN","JJ","NNS","IN","NN","NN","IN","JJ","NNS"
            ]
          }
        ]
      }
   ]}
```

4. Code

4.1 Architecture

L'architecture choisie est une architecture REST qui utilise un seul serveur. C'est une des contraintes imposées du projet.

4.1.1 REST

Representational State Transfer est un style d'architecture qui impose des contraintes. Tout objet (le résultat d'une fonction, des données etc.) est considéré comme ressource. Et afin que les acteurs (serveurs, base de données etc.) puissent avoir un protocole commun, le protocole HTTP a été choisi et est appliqué strictement. Chaque ressource possède donc un identificateur uniforme unique (URI). Lorsque des serveurs ou des utilisateurs cherchent une ressource, ils s'adressent au serveur via l'URI et le serveur leur envoie une représentation de la ressource (souvent sous la forme XML, Json …). L'utilisateur peut demander de recevoir une représentation, de la modifier, d'en créer une ou de la supprimer. Pour ce faire les « verbes » http sont utilisés afin de savoir quelles actions sont réalisées :

- GET : lecture d'une représentation
- POST : création d'une nouvelle ressource en donnant au serveur la représentation de la nouvelle ressource
- PUT : Modification d'une ressource en envoyant au serveur la représentation complète de la ressource
- DELETE : Suppression d'une ressource

C'est cette architecture qui a été choisie pour ce projet.

4.2 API

Presque toutes les interfaces ont 3 variables en commun :

- Sections : Spécification des parties du fichier à traiter, plusieurs sections peuvent être spécifiées, mais elles doivent toutes être séparées par « ; »
- cutoff : indique la limite de la longueur de la phrase maximale
- posTags : tous les tags du *part of speech tagging* qui doivent être utilisés, séparés par « ; »

4.2.1 API principale

4.2.1.1 Extraction de concept depuis du texte libre

Reçoit du texte libre en entrée, et retourne une liste de *Sentence* avec leurs concepts.

Tableau 1 : Contenu de la requête http

URI	/transformPlainText
Verbe	POST
Contenu	Du texte en écriture libre (en anglais)
Type de media	text/plain
Paramètre http « cutoff »	Définit la longueur maximale de la phrase
Paramètre http « tags »	Définit les tags du part of speech à utiliser

Tableau 2 : Contenu de la réponse http

Contenu	Une liste de *Sentence* représentant chaque phrase associée à ses concepts
Type de media	application/json

4.2.1.2 Transformation et stockage de document

Cette API cherche et lit les documents en lien avec leur identifiant, puis les transforme et les enregistre dans Elasticsearch.

Tableau 3 : Contenu de la requête http

URI	/storeConcepts
Verbe	POST
Contenu	Une liste des identifiants des documents à traiter
Type de media	application/json
Paramètre http « sections »	Définit toutes les parties du document à analyser. La nomenclature doit respecter celle de la libraire json-path (Ex :sections=$..title;$..link)
Paramètre http « cutoff »	Définit la longueur maximale de la phrase
Paramètre http « tags »	Définit les tags du part of speech tagging à utiliser

Tableau 4 : Contenu de la réponse http

Contenu	Une liste de *Sentence* représentant chaque phrase associée à ses concepts
Type de media	application/json

4.2.2 API json

L'api json lit les fichiers qui ont été stockés dans Elasticsearch et les transforme.

4.2.2.1 Séparation des phrases d'un seul document json

Cette API cherche le document selon son identifiant, le lit et sépare toutes les phrases en remplaçant le texte par un tableau de String. Finalement, il retourne le document transformé.

Tableau 5 : Contenu de la requête http

URI	/docs/*ID_du_document*/splitSentence
Verbe	POST
Contenu	Aucun
Type de media	application/json (le header content-type doit être présent)
Paramètre http « sections »	Définit toutes les parties du document à analyser. La nomenclature doit respecter celle de la libraire json-path (Ex :sections=$..title;$..link)
Paramètre http « cutoff »	Définit la longueur maximale de la phrase
Paramètre http « tags »	Définit les tags du part of speech tagging à utiliser

Tableau 6 : Contenu de la réponse http

Contenu	Un document formaté
Type de media	application/json

4.2.2.2 Séparation des phrases sur plusieurs documents json

Cette API cherche et lit les documents selon les identifiants envoyés et sépare toutes les phrases en remplaçant le texte par un tableau de String. Puis elle retourne tous les documents traités sous forme de liste.

Tableau 7 : Contenu de la requête http

URI	/docs/splitSentence
Verbe	POST
Contenu	Une liste des identifiants des documents à traiter
Type de media	application/json
Paramètre http « sections »	Définit toutes les parties du document à analyser. La nomenclature doit respecter celle de la libraire json-path (Ex :sections=$..title;$..link)
Paramètre http « cutoff »	Définit la longueur maximale de la phrase
Paramètre http « tags »	Définit les tags du part of speech tagging à utiliser

Tableau 8 : Contenu de la réponse http

Contenu	Une liste de document formaté
Type de media	application/json

4.2.2.3 Transformation des phrases en concept d'un seul document json

Cette API lit et cherche le document selon son identifiant, elle sépare toutes les phrases et cherche les concepts pour chaque phrase. Puis elle retourne le document formaté.

Tableau 9 : Contenu de la requête http

URI	/docs/*ID_du_document*/transformToConcepts
Verbe	POST
Contenu	Aucun
Type de media	application/json (le header content-type doit être présent)
Paramètre http « sections »	Définit toutes les parties du document à analyser. La nomenclature doit respecter celle de la libraire json-path (Ex :sections=$..title;$..link)
Paramètre http « cutoff »	Définit la longueur maximale de la phrase
Paramètre http « tags »	Définit les tags du part of speech tagging à utiliser

Tableau 10 : Contenu de la réponse http

Contenu	Un document formaté
Type de media	application/json

4.2.2.4 Transformation des phrases en concept sur plusieurs documents json

Cette API cherche et lit les documents selon les identifiants envoyé, et elle sépare toutes les phrases, puis elle cherche les concepts pour chaque phrase. Finalement, les documents sont retournés sous forme de liste de String.

Tableau 11 : Contenu de la requête http

URI	/docs/transformToConcepts
Verbe	POST
Contenu	Une liste des identifiants des documents à traiter
Type de media	application/json
Paramètre http « sections »	Définit toutes les parties du document à analyser. La nomenclature doit respecter celle de la libraire json-path (Ex :sections=$..title;$..link)
Paramètre http « cutoff »	Définit la longueur maximale de la phrase
Paramètre http « tags »	Définit les tags du part of speech tagging à utiliser

Tableau 12 : Contenu de la réponse http

Contenu	Une liste de document formaté
Type de media	application/json

4.2.3 API xml

L'API xml lit directement les essais cliniques au format xml et les transforme.

4.2.3.1 Séparation des phrases d'un seul document xml

Cette API lit le document passé dans la requête, et sépare toutes les phrases en remplaçant le texte par des nœuds <sentence> pour chaque phrase. Finalement, elle retourne le document transformé.

Tableau 13 : Contenu de la requête http

URI	/docs/*ID_du_document*/splitSentence
Verbe	POST
Contenu	L'essai clinique
Type de media	application/xml
Paramètre http « sections »	Définit toutes les parties du document à analyser. Chaque nom ajouté correspond au « tag name » du nœud racine et de ces enfants à traiter
Paramètre http « cutoff »	Définit la longueur maximale de la phrase
Paramètre http « tags »	Définit les tags du part of speech tagging à utiliser

Tableau 14 : Contenu de la réponse http

Contenu	Un document formaté
Type de media	Application/xml

4.2.3.2 Séparation des phrases sur plusieurs documents xml

Cette API lit les documents selon la liste des chemins d'accès des fichiers fournit, et elle sépare toutes les phrases en remplaçant le texte par des nœuds <sentence> pour chaque phrase. Puis elle retourne tous les documents traités sous forme de liste.

Tableau 15 : Contenu de la requête http

URI	/docs/splitSentence
Verbe	POST
Contenu	Une liste de chemins d'accès des fichiers
Type de media	application/xml
Paramètre http « sections »	Définit toutes les parties du document à analyser. Chaque nom ajouté correspond au « tag name » du nœud racine et de ces enfants à traiter
Paramètre http « cutoff »	Définit la longueur maximale de la phrase
Paramètre http « tags »	Définit les tags du part of speech tagging à utiliser

Tableau 16 : Contenu de la réponse http

Contenu	Une liste de document formaté
Type de media	application/xml

4.2.3.3 Transformation des phrases en concept d'un seul document xml

Cette API lit le document passé dans la requête, elle sépare toutes les phrases et cherche les concepts pour chaque phrase. Puis chaque bloc de texte est remplacé par des nœuds. Finalement, elle retourne le document formaté.

Tableau 17 : Contenu de la requête http

URI	/ docs/*ID_du_document*/transformToConcepts
Verbe	POST
Contenu	L'essai clinique
Type de media	application/xml
Paramètre http « sections »	Définit toutes les parties du document à analyser. Chaque nom ajouté correspond au « tag name » du nœud racine et de ces enfants à traiter
Paramètre http « cutoff »	Définit la longueur maximale de la phrase
Paramètre http « tags »	Définit les tags du part of speech tagging à utiliser

Tableau 18 : Contenu de la réponse http

Contenu	Un document formaté
Type de media	application/xml

4.2.3.4 Transformation des phrases en concept sur plusieurs documents xml

Cette API lit les documents selon la liste des chemins d'accès des fichiers fournit, elle sépare toutes les phrases et cherche les concepts pour chaque phrase. Puis chaque bloc de texte est remplacé par des nœuds. Finalement, elle retourne le document formaté.

Tableau 19 : Contenu de la requête http

URI	/docs/transformToConcepts
Verbe	POST
Contenu	Une liste d'identifiant de documents
Type de media	application/xml
Paramètre http « sections »	Définit toutes les parties du document à analyser. Chaque nom ajouté correspond au « tag name » du nœud racine et de ces enfants à traiter
Paramètre http « cutoff »	Définit la longueur maximale de la phrase
Paramètre http « tags »	Définit les tags du part of speech tagging à utiliser

Tableau 20 : Contenu de la réponse http

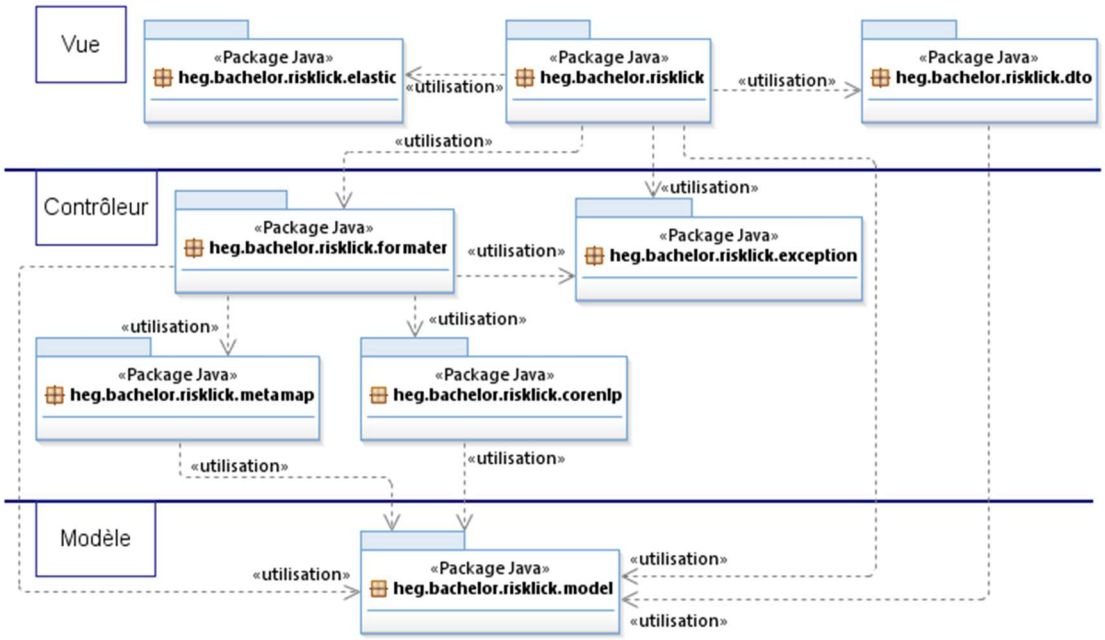
Contenu	Une liste de document formaté
Type de media	application/xml

4.3 Diagramme de classes du projet

Le projet est structuré comme suit (selon le modèle MVC) :

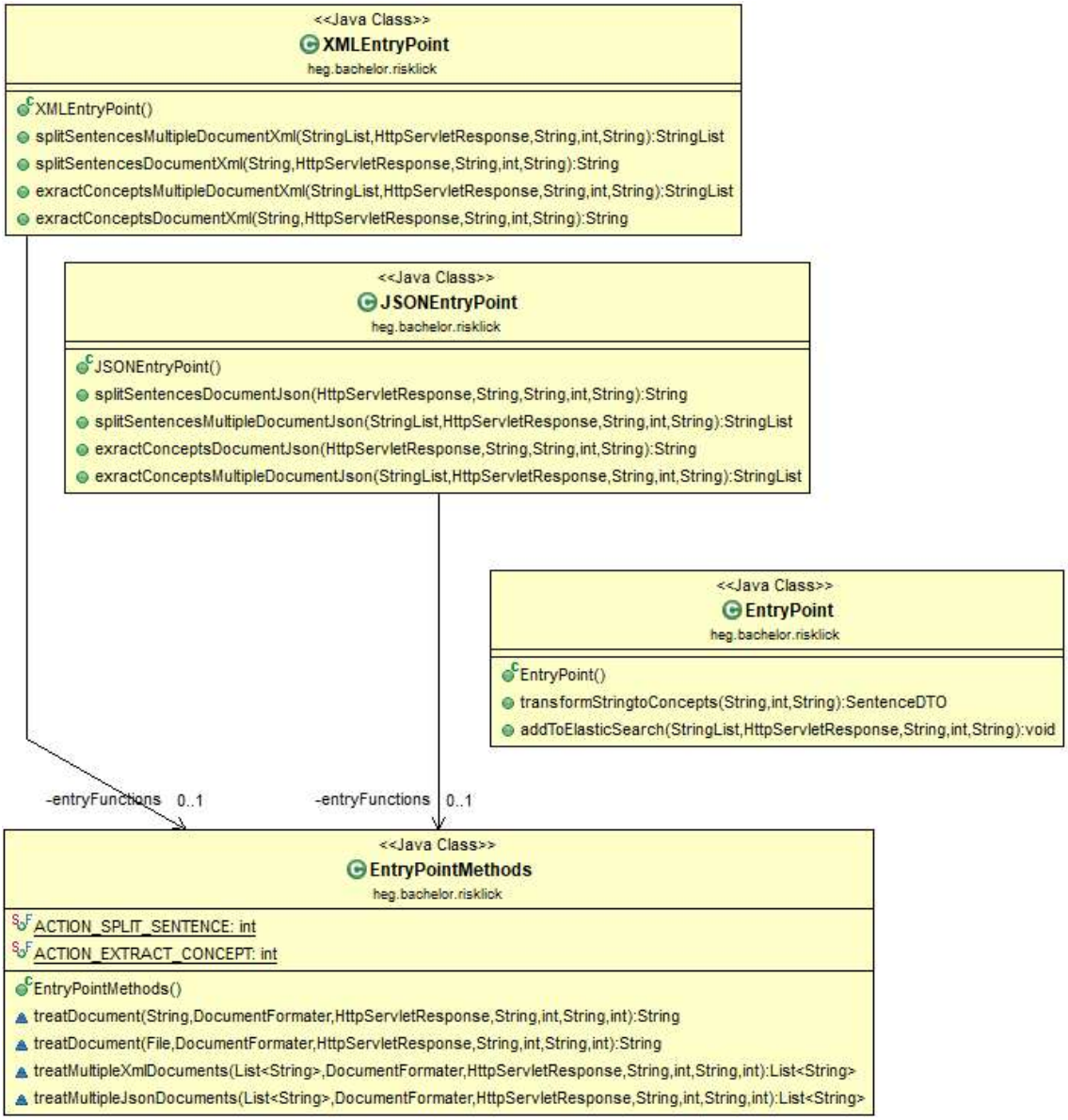
- Les classes qui offrent une interface REST font partie de la « vue »
- Les classes dans le package heg.bachelor.risklick.formater sont les contrôleurs. Elles délèguent la tâche de séparation des phrases et d'extraction de concepts.
- Le modèle est la classe *Sentence* qui est instancié par la classe *SentenceSpliter*. Cet objet est utilisé à tous les niveaux du projet.

Figure 11 : Diagramme de classe des packages Java



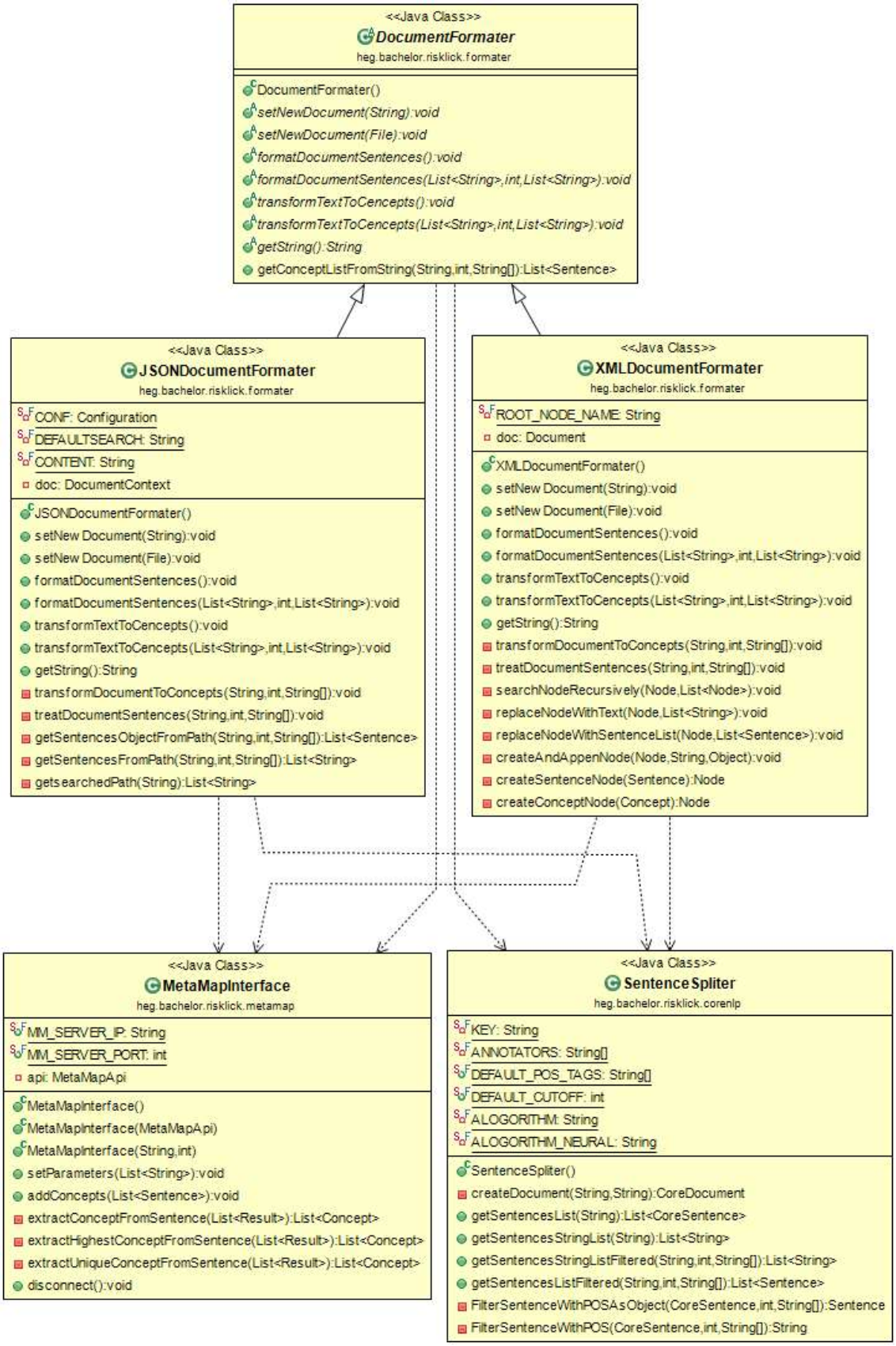
La partie « vue » est constituée de 4 classes. La classe *EntryPointMethods* contient toutes les méthodes qui sont en commun et utilisée par les autres classes. La classe *JSONEntryPoint* fournit les 4 interfaces REST pour les documents json. Quant à la classe *XMLEntryPoint*, elle fournit les 4 autres interfaces REST pour les fichiers xml. Finalement la Classe *EntryPoint* offre l'interface REST qui convertit du texte simple en concepts et la 2e qui transforme et stocke les fichiers dans une base de données persistante.

Figure 12 : Les classes REST



Il existe un contrôleur par type de fichier (xml et json). Le contrôleur est instancié par la « vue », puis il traîte le fichier courant. Il transforme le document sous forme de chaîne de caractères en un objet Java manipulable, puis il envoie le texte aux différents services pour effectuer le formatage. Une fois l'opération terminée, il retransforme le document en chaîne de caractères quand la « vue » le demande.

Figure 13 : Les classes du formatage



5. Résultat

5.1 Exemple de fichier transformé

Un exemple de fichier transformé json se trouve en annexe 4. Le fichier est beaucoup trop long pour être directement inséré dans document.

5.2 Base de données

Une base de données est constituée de 3 parties :

- La partie principale qui contient 80% des fichiers
- La partie *Dev* qui contient 10% des fichiers
- La partie *Test* qui contient 10% des fichiers

La partie principale constitue la partie qui va entraîner les algorithmes, la partie *dev* servira à paramétrer l'algorithme et la partie *test* testera la précision de l'algorithme.

5.2.1 Vue d'ensemble des bases de données

Deux jeux de données ont été créés, une version qui contient tous les sets de concepts trouvés, ce qui fait de très longues listes pour chaque phrase. La deuxième version contient les fichiers dont seul le meilleur set de résultat a été gardé par phrase. Ainsi, l'utilisateur a le choix entre une base de données plus restreinte, mais plus précise ou une base de données plus large.

Tableau 21 : Vue d'ensemble de la base de données

Nom de l'indexe	**Nombre de fichiers**	**Fichiers transformés**
risklick_formated_json	32'207	NCT00001070 - NCT00369993
risklick_formated_json_dev	3'011	NCT00370006 - NCT00409994
risklick_formated_json_test	3'012	NCT00410007 - NCT00449995

Tableau 22 : Vue d'ensemble de la base de données alternative

Nom de l'indexe	**Nombre de fichiers**	**Fichiers transformés**
risklick_formated_high_json	32'207	NCT00001070 - NCT00369993
risklick_formated_high_json_dev	3'011	NCT00370006 - NCT00409994
risklick_formated_high_json_test	3'012	NCT00410007 - NCT00449995

Le premier tableau décrit la base de données dont les fichiers contiennent tous les sets de concepts trouvés par MetaMap, alors que le deuxième tableau décrit la base de données alternative qui ne contient que le meilleur set de concepts par phrase.

5.2.2 Performance

La performance moyenne de transformation de fichiers est d'environ 800 fichiers par heure. Cette mesure a été prise lors de la création de la base de données. Un client REST a appelé la base de données et a calculé le temps que le serveur a pris pour répondre. Pour trouver ce résultat, le nombre de fichiers transformé a été pris, divisé par le temps écoulé en milliseconde, le tout multiplié par 3'600'000.

Ce nombre peut fortement varier en fonction des tailles et types de fichiers à traiter, les paramètres utilisés et les capacités de calcul de la machine. Lors des premiers essais de transformation, il n'y avait pas de limite de taille des phrases, le document était entièrement traité (chaque texte possible dans le fichier était traité). La performance était extrêmement faible (moins de 80 fichiers par heure) et très variable. Un fichier de 8Ko prenait 8 secondes, alors qu'un fichier de 100Ko pouvait prendre plus de 4 minutes.

De plus, si la phrase en question était trop longue, la durée de transformation dégénérait au point que le processus se bloquait sur une même phrase pendant plus de 30 minutes (avec une utilisation de la RAM et du processeur a 100%, contrairement à une utilisation de 30% du processeur habituellement). Les paramètres pour limiter la taille des phrases, ainsi que ceux qui indiquent les parties du document à traiter sont importants pour le projet.

6. Utilité future

Les buts de la base de données sont de faciliter le traitement des fichiers de manière automatique et d'entraîner des modèles ou des algorithmes pour permettre de prédire le risque d'échec d'un essai clinique.

6.1 Comparaison de documents pour la prédiction

La prochaine phase serait de pouvoir comparer les fichiers entre eux. Ainsi, on peut prendre un essai clinique dont on aimerait connaître le risque d'échec et trouver les essais cliniques qui sont « les plus proches ». L'étape suivante consiste à regarder la réussite ou non des N essais cliniques les plus proches et les comparer pour prédire le risque d'échec.

La base de données Elasticsearch possède un moteur de recherche. Elle est capable de trouver les documents les plus proches en fonction des paramètres envoyés. Cette fonction s'appelle « more_like_this »

Figure 14 : Exemple de requête « more_like_this » de base

```
{
  "query": {
      "more_like_this" : {
          "fields" : ["title", "description"],
          "like" : "Once upon a time",
          "min_term_freq" : 1,
          "max_query_terms" : 12
      }
  }
}
```

(Elasticsearch)

Elasticsearch va utiliser le moteur de recherche *Lucene* pour chercher dans tous les documents les textes des champs « title » et « description » qui ressemblent le plus à « Once upon a time ». Le moteur va émettre un poids à chaque document comparé qui correspond à un poids de confiance. Plus ce nombre est grand, plus il est « proche » de la recherche. Il suffit donc de prendre les X documents les plus proches et de les traiter.

Dans notre cas, il faudrait comparer tous les champs formatés en concepts avec les concepts que l'on a obtenu de l'essai clinique que l'on veut prédire, prendre les x résultats les plus proches et prédire en fonction des résultats.

6.1.1 Exemple d'utilisation de la requête « more_like_this »

Elasticsearch propose notamment la recherche de documents les plus proches à partir de documents venant d'autres indices.

Voici un Exemple de recherche pour notre cas :

Figure 15 : Exemple de requête « more_like_this » adaptée

```
{
    "query": {
        "more_like_this": {
            "like": [
                {
                    "_index": "risklick_formated_json_dev",
                    "_type": "_doc",
                    "_id": "NCT00385034"
                }
            ],
            "max_query_terms" : 25,
            "min_term_freq" : 1,
            "min_doc_freq":1
        }
    }
}
```

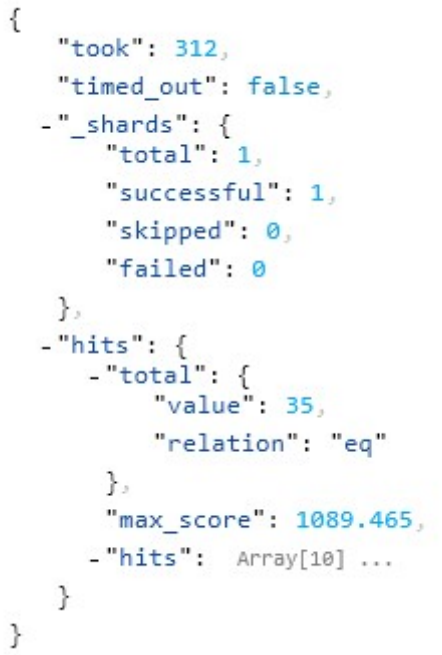
La requête va chercher le document se trouvant dans l'indice « riscklick_formated_json_dev » dont l'identifiant est « NCT00385034 ». Il va être analysé, puis le moteur de recherche va faire son travail et trouver les documents les plus « ressemblants ».

Il existe plusieurs paramètres qui contrôlent les résultats de la requête :

- max_query_terms : le nombre de termes maximal que la requête prend en compte pour la recherche. La valeur par défaut est de 25
- min_term_freq : le nombre minimal qu'un terme doit être présent du fichier source. La valeur par défaut est de 2
- min_doc_freq : la fréquence minimale de documents que le terme doit apparaître dans les documents. La valeur par défaut est de 5
- max_doc_freq : la fréquence maximale de documents que le terme doit apparaître dans les documents. La valeur par défaut est illimitée

Ces paramètres peuvent changer radicalement le résultat, il faut donc expérimenter pour trouver les meilleurs résultats. C'est à cet effet que la partie *dev* de la base de données existe.

Figure 16 : Résultat d'une requête « more_like_this »



Voici un exemple de résultat (les « hits » contenant les documents ont été omis, ils sont trop verbeux pour être afficher). La réponse affiche le nombre de documents qui a été trouvé (35 dans ce cas), ainsi que le poids de confiance le plus élevé trouvé (1089.465). Puis il affiche les 10 documents en entier les plus « proches » triés par *score* descendant.

Voici la liste des champs « overall_status » trouvés (l'ordre a été gardé) dans la réponse de la requête

1. "Completed"
2. "Completed"
3. "Completed"
4. "Completed"
5. "Completed"
6. "Completed"
7. "Completed",
8. "Unknown status"
9. "Terminated"
10. "Active, not recruiting"

On peut constater que les 7 documents les plus proches ont le champ « overall_status » *completed*, alors qu'un seul a le champ « overall_status » *terminated*. Donc, on peut conclure que le document NCT00385034 a de bonne chance d'avoir « overall_status » "Completed" (70%).

Ce traitement peut être automatisé. Une première méthode pourrait exécuter diverses requêtes avec des paramètres différents et retenir le set de paramètres qui obtient les meilleurs résultats. Puis une deuxième méthode qui prédit les essais cliniques en boucle avec les paramètres trouvés dans la première méthode.

7. Conclusion

Extraire des concepts à partir d'un serveur peut sembler facile, il faut juste envoyer du texte et prendre les concepts. Sauf que de la théorie à la pratique, il y a un autre monde. Il y a eu en premier la compréhension du fonctionnement du serveur qui n'était pas si simple, puis les soucis de performance ou de blocage et des comportements inattendus qui sont survenus. Néanmoins, j'ai pu surpasser tous ces problèmes et apprendre énormément sur le traitement du langage naturel. J'espère également que la base de données, ainsi que l'application, répondront à l'attente des différents acteurs et seront utiles pour atteindre le but final du projet.

Bibliographie

2019. *Essai clinique* [en ligne]. S.l. : s.n. [Consulté le 1 août 2019]. Disponible à l'adresse : https://fr.wikipedia.org/w/index.php?title=Essai_clinique&oldid=161416525.

2019. *Interface de programmation* [en ligne]. S.l. : s.n. [Consulté le 26 juillet 2019]. Disponible à l'adresse : https://fr.wikipedia.org/w/index.php?title=Interface_de_programmation&oldid=160979493.

2019. *Méthode des k plus proches voisins* [en ligne]. S.l. : s.n. [Consulté le 30 juillet 2019]. Disponible à l'adresse : https://fr.wikipedia.org/w/index.php?title=M%C3%A9thode_des_k_plus_proches_voisins&oldid=158322796.

2019. *Representational state transfer* [en ligne]. S.l. : s.n. [Consulté le 23 juillet 2019]. Disponible à l'adresse : https://fr.wikipedia.org/w/index.php?title=Representational_state_transfer&oldid=160222966.

THE APACHE SOFTWARE FOUNDATION. Apache Tomcat® - Apache Tomcat 9 Software Downloads. In : [en ligne]. [Consulté le 12 juin 2019 a]. Disponible à l'adresse : https://tomcat.apache.org/download-90.cgi.

THE APACHE SOFTWARE FOUNDATION. Apache Tomcat® - Welcome! In : [en ligne]. [Consulté le 26 juillet 2019 b]. Disponible à l'adresse : http://tomcat.apache.org/.

SOLID IT GMBH., [2019]. DB-Engines Ranking. In : *DB-Engines* [en ligne]. [Consulté le 26 juillet 2019 c]. Disponible à l'adresse : https://db-engines.com/en/ranking/search+engine.

ELASTICSEARCH B.V. Download Elasticsearch Free • Get Started Now | Elastic. In : [en ligne]. [Consulté le 12 juin 2019 d]. Disponible à l'adresse : https://www.elastic.co/fr/downloads/elasticsearch.

ELASTICSEARCH B.V. Download Kibana Free • Get Started Now | Elastic. In : [en ligne]. [Consulté le 12 juin 2019 e]. Disponible à l'adresse : https://www.elastic.co/fr/downloads/kibana.

ELASTICSEARCH B.V. Download Logstash Free • Get Started Now | Elastic. In : [en ligne]. [Consulté le 12 juin 2019 f]. Disponible à l'adresse : https://www.elastic.co/fr/downloads/logstash.

W3C, [2016]. Extensible Markup Language (XML). In : [en ligne]. [Consulté le 26 juillet 2019 g]. Disponible à l'adresse : https://www.w3.org/XML/.

Home - ClinicalTrials.gov. In : [en ligne]. [Consulté le 25 juillet 2019 h]. Disponible à l'adresse : https://clinicaltrials.gov/.

THE APACHE SOFTWARE FOUNDATION. Maven – Introduction. In : [en ligne]. [Consulté le 23 juillet 2019 i]. Disponible à l'adresse : https://maven.apache.org/what-is-maven.html.

ELASTICSEARCH B.V. More like this query | Elasticsearch Reference [7.2] | Elastic. In : [en ligne]. [Consulté le 30 juillet 2019 j]. Disponible à l'adresse : https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-mlt-query.html.

pos-set.pdf [en ligne]. S.l. : s.n. [Consulté le 26 juillet 2019 k]. Disponible à l'adresse : https://www.eecis.udel.edu/~vijay/cis889/ie/pos-set.pdf.

ASF INFRABOT, [2014]. PoweredBy - Apache Tomcat - Apache Software Foundation. In : [en ligne]. [Consulté le 26 juillet 2019 l]. Disponible à l'adresse : https://cwiki.apache.org/confluence/display/TOMCAT/PoweredBy.

Stanford CoreNLP – Natural language software | Stanford CoreNLP. In : [en ligne]. [Consulté le 12 juin 2019 m]. Disponible à l'adresse : https://stanfordnlp.github.io/CoreNLP/.

[2019] Unified Medical Language System (UMLS). In : [en ligne]. [Consulté le 26 juillet 2019 n]. Disponible à l'adresse : https://www.nlm.nih.gov/research/umls/.

Web Services Architecture. In : [en ligne]. [Consulté le 23 juillet 2019 o]. Disponible à l'adresse : https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest.

FOUNDATION, Eclipse. Eclipse Downloads | The Eclipse Foundation. In : [en ligne]. [Consulté le 12 juin 2019]. Disponible à l'adresse : https://www.eclipse.org/downloads/.

ORACLE. Qu'est-ce que Java et pourquoi en ai-je besoin ? In : [en ligne]. [Consulté le 23 juillet 2019]. Disponible à l'adresse : https://www.java.com/fr/download/faq/whatis_java.xml.

[2019]. MetaMap - A Tool For Recognizing UMLS Concepts in Text. In : [en ligne]. [Consulté le 8 août 2019 a]. Disponible à l'adresse : https://metamap.nlm.nih.gov/.

Using the MetaMap Java API. In : [en ligne]. [Consulté le 8 août 2019 b]. Disponible à l'adresse : https://metamap.nlm.nih.gov/Docs/README_javaapi.shtml.

MANNING, CHRISTOPHER D., MIHAI SURDEANU, JOHN BAUER, JENNY FINKEL, STEVEN J. BETHARD, AND DAVID MCCLOSKY, [2014]. The Stanford CoreNLP Natural Language Processing Toolkit In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60. [pdf] [bib]

BODENREIDER O., [2004] The Unified Medical Language System (UMLS): integrating biomedical terminology. Nucleic Acids

Annexe 1 : Exemple d'un essai clinique

```
<clinical_study>
  <!-- This xml conforms to an XML Schema at:
    https://clinicaltrials.gov/ct2/html/images/info/public.xsd -->
  <required_header>
    <download_date>ClinicalTrials.gov processed this data on March 26, 2019</download_date>
    <link_text>Link to the current ClinicalTrials.gov record.</link_text>
    <url>https://clinicaltrials.gov/show/NCT00001070</url>
  </required_header>
  <id_info>
    <org_study_id>DATRI 017</org_study_id>
    <nct_id>NCT00001070</nct_id>
  </id_info>
  <brief_title>The Safety and Effectiveness of Different Doses of Vitamin C in HIV-Infected
Patients</brief_title>
  <official_title>A Dose Escalation Study to Evaluate Tolerance and Antiviral Effect of Oral
Vitamin C in Two Groups of HIV-Infected Adults</official_title>
  <sponsors>
    <lead_sponsor>
      <agency>National Institute of Allergy and Infectious Diseases (NIAID)</agency>
      <agency_class>NIH</agency_class>
    </lead_sponsor>
  </sponsors>
  <source>National Institute of Allergy and Infectious Diseases (NIAID)</source>
  <brief_summary>
    <textblock>
      This is a study to evaluate the tolerance and antiviral effect of oral vitamin C in
      HIV-infected.
    </textblock>
  </brief_summary>
  <overall_status>Withdrawn</overall_status>
  <phase>N/A</phase>
  <study_type>Interventional</study_type>
  <has_expanded_access>No</has_expanded_access>
  <study_design_info>
    <primary_purpose>Treatment</primary_purpose>
  </study_design_info>
  <enrollment type="Actual">0</enrollment>
  <condition>HIV Infections</condition>
  <intervention>
    <intervention_type>Drug</intervention_type>
    <intervention_name>Oral Vitamin C</intervention_name>
  </intervention>
  <eligibility>
    <gender>All</gender>
    <minimum_age>18 Years</minimum_age>
    <maximum_age>N/A</maximum_age>
    <healthy_volunteers>No</healthy_volunteers>
  </eligibility>
  <verification_date>December 2012</verification_date>
  <study_first_submitted>November 2, 1999</study_first_submitted>
  <study_first_submitted_qc>August 30, 2001</study_first_submitted_qc>
  <study_first_posted type="Estimate">August 31, 2001</study_first_posted>
  <last_update_submitted>December 28, 2012</last_update_submitted>
  <last_update_submitted_qc>December 28, 2012</last_update_submitted_qc>
  <last_update_posted type="Estimate">December 31, 2012</last_update_posted>
  <condition_browse>
    <!-- CAUTION:  The following MeSH terms are assigned with an imperfect algorithm           -
->
    <mesh_term>HIV Infections</mesh_term>
  </condition_browse>
  <intervention_browse>
    <!-- CAUTION:  The following MeSH terms are assigned with an imperfect algorithm           -
->
    <mesh_term>Vitamins</mesh_term>
    <mesh_term>Ascorbic Acid</mesh_term>
  </intervention_browse>
  <!-- Results have not yet been posted for this study                                        -
->
</clinical_study>
```

Annexe 2 : API complète de MetaMap

```
static void process(String terms, PrintStream out) throws Exception {
  MetaMapApi api = new MetaMapApiImpl();
  List<Result> resultList = api.processCitationsFromString(terms);
  for (Result result : resultList) {
    if (result != null) {
      out.println("input text: ");
      out.println(" " + result.getInputText());
      List<AcronymsAbbrevs> aaList = result.getAcronymsAbbrevsList();
      if (aaList.size() > 0) {
        out.println("Acronyms and Abbreviations:");
        for (AcronymsAbbrevs e : aaList) {
          out.println("Acronym: " + e.getAcronym());
          out.println("Expansion: " + e.getExpansion());
          out.println("Count list: " + e.getCountList());
          out.println("CUI list: " + e.getCUIList());
        }
      }
      List<Negation> negList = result.getNegationList();
      if (negList.size() > 0) {
        out.println("Negations:");
        for (Negation e : negList) {
          out.println("type: " + e.getType());
          out.print("Trigger: " + e.getTrigger() + ": [");
          for (Position pos : e.getTriggerPositionList()) {
             out.print(pos + ",");
           }
           out.println("]");
           out.print("ConceptPairs: [");
           for (ConceptPair pair : e.getConceptPairList()) {
             out.print(pair + ",");
           }
           out.println("]");
           out.print("ConceptPositionList: [");
           for (Position pos : e.getConceptPositionList()) {
             out.print(pos + ",");
            }
            out.println("]");
          }
        }
        for (Utterance utterance : result.getUtteranceList()) {
          out.println("Utterance:");
          out.println(" Id: " + utterance.getId());
          out.println(" Utterance text: " + utterance.getString());
          out.println(" Position: " + utterance.getPosition());
          for (PCM pcm : utterance.getPCMList()) {
            out.println("Phrase:");
            out.println(" text: " + pcm.getPhrase().getPhraseText());
            out.println("Minimal Commitment Parse:"+pcm.getPhrase().getMincoManAsString());
            out.println("Candidates:");
            for (Ev ev : pcm.getCandidatesInstance().getEvList()) {
              out.println(" Candidate:");
              out.println("  Score: " + ev.getScore());
              out.println("  Concept Id: " + ev.getConceptId());
              out.println("  Concept Name: " + ev.getConceptName());
              out.println("  Preferred Name: " + ev.getPreferredName());
              out.println("  Matched Words: " + ev.getMatchedWords());
              out.println("  Semantic Types: " + ev.getSemanticTypes());
              out.println("  MatchMap: " + ev.getMatchMap());
              out.println("  MatchMap alt. repr.: " + ev.getMatchMapList());
              out.println("  is Head?: " + ev.isHead());
              out.println("  is Overmatch?: " + ev.isOvermatch());
              out.println("  Sources: " + ev.getSources());
              out.println("  Positional Info: " + ev.getPositionalInfo());
              out.println("  Pruning Status: " + ev.getPruningStatus());
              out.println("  Negation Status: " + ev.getNegationStatus());
            }
```

```
            out.println("Mappings:");
            for (Mapping map : pcm.getMappingList()) {
              out.println(" Map Score: " + map.getScore());
              for (Ev mapEv : map.getEvList()) {
                out.println("   Score: " + mapEv.getScore());
                out.println("   Concept Id: " + mapEv.getConceptId());
                out.println("   Concept Name: " + mapEv.getConceptName());
                out.println("   Preferred Name: " + mapEv.getPreferredName());
                out.println("   Matched Words: " + mapEv.getMatchedWords());
                out.println("   Semantic Types: " + mapEv.getSemanticTypes());
                out.println("   MatchMap: " + mapEv.getMatchMap());
                out.println("   MatchMap alt. repr.: " + mapEv.getMatchMapList());
                out.println("   is Head?: " + mapEv.isHead());
                out.println("   is Overmatch?: " + mapEv.isOvermatch());
                out.println("   Sources: " + mapEv.getSources());
                out.println("   Positional Info: " + mapEv.getPositionalInfo());
                out.println("   Pruning Status: " + mapEv.getPruningStatus());
                out.println("   Negation Status: " + mapEv.getNegationStatus());
              }
            }
          }
        }
      }else {
        out.println("NULL result instance! ");
      }
    }
  api.resetOptions();
  api.disconnect()
}
```

Annexe 3 : exemple de résultat de MetaMap

input text:
high blood pressure
Utterance:
Id: 00000000.tx.1
Utterance text: high blood pressure
Position: (0, 19)
Phrase:
text: high blood pressure
Minimal Commitment Parse: [head([lexmatch([high blood pressure]),inputmatch([high,blood,pressure]),tag(noun),tokens([high,blood,pressure])])]
Candidates:
Mappings:
Map Score: -1000
Score: -1000
Concept Id: C0020538
Concept Name: Blood Pressure, High
Preferred Name: Hypertensive disease
Matched Words: [high, blood, pressure]
Semantic Types: [dsyn]
MatchMap: [[[1, 3], [1, 3], 0]]
MatchMap alt. repr.: [[[phrase start: 1, phrase end: 3], [concept start: 1, concept end: 3], lexical variation: 0]]
is Head?: true
is Overmatch?: false
Sources: [AOD, CCS, CCS_10, CHV, COSTAR, CSP, CST, DXP, HPO, ICD10CM, ICD9CM, LCH, LCH_NW, LNC, MEDLINEPLUS, MSH, MTH, MTHICD9, NCI, NCI_CTCAE, NCI_FDA, NCI_NCI-GLOSS, NCI_NICHD, NDFRT, NLMSubSyn, OMIM, SNM, SNMI, SNOMEDCT_US, SNOMEDCT_VET]
Positional Info: [(0, 19)]
Pruning Status: 0
Negation Status: 0
Map Score: -1000
Score: -1000
Concept Id: C2926615
Concept Name: High blood pressure
Preferred Name: Ever told by doctor or nurse that you have high blood pressure:Finding:Point in time:^Patient:Ordinal
Matched Words: [high, blood, pressure]
Semantic Types: [clna]
MatchMap: [[[1, 3], [1, 3], 0]]
MatchMap alt. repr.: [[[phrase start: 1, phrase end: 3], [concept start: 1, concept end: 3], lexical variation: 0]]
is Head?: true
is Overmatch?: false
Sources: [LNC, MTH, NLMSubSyn]
Positional Info: [(0, 19)]
Pruning Status: 0
Negation Status: 0

Annexe 4 : Exemple de résultat de transformation


```
{
  "_index": "risklick_formated_high_json",
  "_type": "_doc",
  "_id": "NCT00001070",
  "_version": 1,
  "_seq_no": 6911,
  "_primary_term": 3,
  "found": true,
  "_source": {
    "id": ["NCT00001070"],
    "type": "clinical_study",
    "@version": "1",
    "clinical_study": {
      "has_expanded_access": [
        { "content": "No"}
      ],
      "verification_date": [
        {
          "content": "December 2012"
        }
      ],
      "sponsors": [
        {
          "lead_sponsor": [
            {
              "agency_class": [
                {
                  "content": "NIH"
                }
              ],
              "agency": [
                {
                  "content": "National Institute of Allergy and Infectious Diseases
(NIAID)"
                }
              ]
            }
          ]
        }
      ],
      "study_first_posted": [
        {
          "content": "August 31, 2001",
          "type": "Estimate"
        }
      ],
      "study_type": [
        {
          "content": "Interventional"
        }
      ],
      "intervention_browse": [
        {
          "mesh_term": [
            {
              "content": "Vitamins"
            },
            {
              "content": "Ascorbic Acid"
            }
          ]
        }
      ],
      "required_header": [
        {
          "download_date": [
```

```
        {
          "content": "ClinicalTrials.gov processed this data on March 26, 2019"
        }
      ],
      "link_text": [
        {
          "content": "Link to the current ClinicalTrials.gov record."
        }
      ],
      "url": [
        {
          "content": "https://clinicaltrials.gov/show/NCT00001070"
        }
      ]
    }
  ],
  "id_info": [
    {
      "nct_id": [
        {
          "content": "NCT00001070"
        }
      ],
      "org_study_id": [
        {
          "content": "DATRI 017"
        }
      ]
    }
  ],
  "source": [
    {
      "content": "National Institute of Allergy and Infectious Diseases (NIAID)"
    }
  ],
  "last_update_posted": [
    {
      "content": "December 31, 2012",
      "type": "Estimate"
    }
  ],
  "official_title": [
    {
      "content": []
    }
  ],
  "condition": [
    {
      "content": "HIV Infections"
    }
  ],
  "last_update_submitted": [
    {
      "content": "December 28, 2012"
    }
  ],
  "study_design_info": [
    {
      "primary_purpose": [
        {
          "content": "Treatment"
        }
      ]
    }
  ],
  "study_first_submitted_qc": [
```

```
        {
          "content": "August 30, 2001"
        }
      ],
      "enrollment": [
        {
          "content": "0",
          "type": "Actual"
        }
      ],
      "intervention": [
        {
          "intervention_name": [
            {
              "content": "Oral Vitamin C"
            }
          ],
          "intervention_type": [
            {
              "content": "Drug"
            }
          ]
        }
      ],
      "eligibility": [
        {
          "healthy_volunteers": [
            {
              "content": "No"
            }
          ],
          "maximum_age": [
            {
              "content": "N/A"
            }
          ],
          "gender": [
            {
              "content": "All"
            }
          ],
          "minimum_age": [
            {
              "content": "18 Years"
            }
          ]
        }
      ],
      "study_first_submitted": [
        {
          "content": "November 2, 1999"
        }
      ],
      "brief_title": [
        {
          "content": [
            {
              "sentence": " Safety  Effectiveness  Different  Doses  Vitamin  C  HIV-
Infected  Patients",
              "originalSentence": "The Safety and Effectiveness of Different Doses of
Vitamin C in HIV-Infected Patients",
              "concepts": [
                {
                  "score": -874,
                  "negation": 0,
                  "name": "Safety-patient",
```

```
                  "id": "C1113679",
                  "preferredName": "patient safety"
                },
                {
                  "score": -624,
                  "negation": 0,
                  "name": "Effectiveness",
                  "id": "C1280519",
                  "preferredName": "Effectiveness"
                },
                {
                  "score": -624,
                  "negation": 0,
                  "name": "Different",
                  "id": "C1705242",
                  "preferredName": "Different"
                },
                {
                  "score": -624,
                  "negation": 0,
                  "name": "Doses",
                  "id": "C0178602",
                  "preferredName": "Dosage"
                },
                {
                  "score": -641,
                  "negation": 0,
                  "name": "VITAMIN C",
                  "id": "C0003968",
                  "preferredName": "Ascorbic Acid"
                },
                {
                  "score": -624,
                  "negation": 0,
                  "name": "HIV",
                  "id": "C0019682",
                  "preferredName": "HIV"
                },
                {
                  "score": -624,
                  "negation": 0,
                  "name": "Infected",
                  "id": "C0439663",
                  "preferredName": "Infected"
                }
              ],
              "posTags": ["DT","NN","CC", "NN","IN","JJ","NNS","IN",
"NN","NN","IN","JJ","NNS"]
            }
          ]
        }
      ],
      "overall_status": [
        {
          "content": "Withdrawn"
        }
      ],
      "last_update_submitted_qc": [
        {
          "content": "December 28, 2012"
        }
      ],
      "condition_browse": [
        {
          "mesh_term": [
            {
```

```
            "content": "HIV Infections"
          }
        ]
      }
    ],
    "brief_summary": [
      {
        "textblock": [
          {
            "content": []
          }
        ]
      }
    ],
    "phase": [
      {
        "content": "N/A"
      }
    ]
  },
  "@timestamp": "2019-06-05T22:28:29.054Z",
  "tags": [
    "multiline"
  ],
  "message": "<clinical_study>\r\n  <!-- This xml conforms to an XML Schema at:\r\n
https://clinicaltrials.gov/ct2/html/images/info/public.xsd -->\r\n  <required_header>\r\n
<download_date>ClinicalTrials.gov processed this data on March 26, 2019</download_date>\r\n
<link_text>Link to the current ClinicalTrials.gov record.</link_text>\r\n
<url>https://clinicaltrials.gov/show/NCT00001070</url>\r\n  </required_header>\r\n
<id_info>\r\n    <org_study_id>DATRI 017</org_study_id>\r\n
<nct_id>NCT00001070</nct_id>\r\n  </id_info>\r\n  <brief_title>The Safety and Effectiveness
of Different Doses of Vitamin C in HIV-Infected Patients</brief_title>\r\n
<official_title>A Dose Escalation Study to Evaluate Tolerance and Antiviral Effect of Oral
Vitamin C in Two Groups of HIV-Infected Adults</official_title>\r\n  <sponsors>\r\n
<lead_sponsor>\r\n      <agency>National Institute of Allergy and Infectious Diseases
(NIAID)</agency>\r\n      <agency_class>NIH</agency_class>\r\n    </lead_sponsor>\r\n
</sponsors>\r\n  <source>National Institute of Allergy and Infectious Diseases
(NIAID)</source>\r\n  <brief_summary>\r\n    <textblock>\r\n      This is a study to
evaluate the tolerance and antiviral effect of oral vitamin C in\r\n      HIV-infected.\r\n
</textblock>\r\n  </brief_summary>\r\n  <overall_status>Withdrawn</overall_status>\r\n
<phase>N/A</phase>\r\n  <study_type>Interventional</study_type>\r\n
<has_expanded_access>No</has_expanded_access>\r\n  <study_design_info>\r\n
<primary_purpose>Treatment</primary_purpose>\r\n  </study_design_info>\r\n  <enrollment
type=\"Actual\">0</enrollment>\r\n  <condition>HIV Infections</condition>\r\n
<intervention>\r\n    <intervention_type>Drug</intervention_type>\r\n
<intervention_name>Oral Vitamin C</intervention_name>\r\n  </intervention>\r\n
<eligibility>\r\n    <gender>All</gender>\r\n    <minimum_age>18 Years</minimum_age>\r\n
<maximum_age>N/A</maximum_age>\r\n    <healthy_volunteers>No</healthy_volunteers>\r\n
</eligibility>\r\n  <verification_date>December 2012</verification_date>\r\n
<study_first_submitted>November 2, 1999</study_first_submitted>\r\n
<study_first_submitted_qc>August 30, 2001</study_first_submitted_qc>\r\n
<study_first_posted type=\"Estimate\">August 31, 2001</study_first_posted>\r\n
<last_update_submitted>December 28, 2012</last_update_submitted>\r\n
<last_update_submitted_qc>December 28, 2012</last_update_submitted_qc>\r\n
<last_update_posted type=\"Estimate\">December 31, 2012</last_update_posted>\r\n
<condition_browse>\r\n    <!-- CAUTION:  The following MeSH terms are assigned with an
imperfect algorithm           -->\r\n    <mesh_term>HIV Infections</mesh_term>\r\n
</condition_browse>\r\n  <intervention_browse>\r\n    <!-- CAUTION:  The following MeSH
terms are assigned with an imperfect algorithm           -->\r\n
<mesh_term>Vitamins</mesh_term>\r\n    <mesh_term>Ascorbic Acid</mesh_term>\r\n
</intervention_browse>\r\n  <!-- Results have not yet been posted for this study
-->\r\n</clinical_study>\r\n\r",
  "path": "D:/HEG/Bachelor/Data/data/NCT0000xxxx/NCT00001070.xml",
  "host": "DESKTOP-B47ONOS"
 }
}
```