Création d'un annotateur à l'aide de composantes de cTAKES pour le traitement de questions cliniques et la recherche sémantique dans des articles médicaux

Rapport final présenté à
M. Michal Iglewski,
superviseur et coordonnateur

par
Valérie Levasseur (LEVV10608406)

dans le cadre du cours
INF4173- Projet Synthèse

Département d'informatique et d'ingénierie
Université du Québec en Outaouais
15 août 2014

TABLE DES MATIÈRES

LISTE DES TABLEAUX ET FIGURES

Tableaux

Figures

Création d'un annotateur à l'aide de composantes de cTAKES pour le traitement de questions cliniques et la recherche sémantique dans des articles médicaux

1 Introduction

Lors de la formation des étudiants en sciences de la santé, plusieurs sont introduits à la médecine factuelle (*evidence-based medecine*). La médecine factuelle, qui est un concept datant du milieu du 19e siècle [1], vise l'utilisation des données probantes par le clinicien, en plus de son jugement et de son expertise, lorsqu'il prend une décision à propos des soins d'un patient [1].

Un des problèmes reliés à la médecine factuelle est la recherche de ces données probantes; c'est-à-dire la recherche des meilleurs articles médicaux présentant le meilleur niveau de preuve. Une application, EBMPICO, visant, entre autres, à aider le clinicien à effectuer cette recherche des articles médicaux les plus pertinents, est d'ailleurs en cours de développement par le département d'informatique et d'ingénierie de l'Université du Québec en Outaouais, en collaboration avec l'Université McGill, l'Agence de la santé et des services sociaux de l'Outaouais et le Centre de santé et de services sociaux de Gatineau [2].

Dans le cadre d'une application telle que EBMPICO, il s'agit non seulement de trouver les meilleures sources d'information, mais aussi de les trouver le plus rapidement possible. Or, selon les lignes directrices de la médecine factuelle, l'utilisation du cadre PICO facilite la recherche de données par le clinicien [3]. Ce cadre propose la formulation d'une question clinique en termes de problème et/ou de population (élément P), d'intervention visée (élément I), de comparaison (éléments C) et de résultats (élément O – pour *outcome*). Non seulement il rend la recherche plus aisée pour le clinicien, mais il permet aussi de dégager certains concepts dans les questions cliniques. Dans le cadre d'une application visant à assister la recherche de données probantes, le dégagement de concepts sémantiques dans les questions cliniques permet d'améliorer la qualité de la recherche en ne retournant au clinicien que les articles correspondants aux concepts clés de la question. Le dégagement de ces concepts peut être réalisé de diverses façons, et

quelques outils sont disponibles pour ce faire. L'un d'eux, cTAKES, qui a été employé dans le cadre de ce projet, contient un ensemble d'annotateurs spécialisés dans le domaine médical et est basé sur l'infrastructure d'application UIMA [4]. C'est que les annotateurs permettent d'identifier des entités sémantiques dans des documents.

Le projet réalisé dans le cadre de ce cours a donc pour but d'améliorer la recherche d'articles en tentant d'établir si un article trouvé correspond à la question du clinicien.

Ce rapport décrit donc les objectifs de ce travail, les outils utilisés, de même qu'une description des étapes conduisant à sa réalisation ainsi que résultats obtenus, suivis des conclusions et des recommandations en découlant.

2 Description du projet

2.1 Objectifs

Le projet consiste à créer un annotateur à l'aide de composantes de cTAKES pour le traitement de questions cliniques et la recherche sémantique dans des articles médicaux afin d'améliorer la recherche d'articles, et ce, en tentant d'établir si un article trouvé correspond à la question du clinicien, tel que mentionné précédemment. C'est qu'avant que cette correspondance ne soit ou non établie, la question et l'article (ou du moins son résumé) sont analysés sémantiquement, en tirant profit des capacités de cTAKES et du cadre PICO.

L'objectif principal du projet est donc la création d'un annotateur qui permet d'analyser la question clinique et d'identifier des concepts clés dans des articles. Les autres objectifs du travail, découlant de l'objectif principal, comprennent l'identification de concepts fréquemment présents dans les questions cliniques, l'analyse des composantes de cTAKES et l'identification correcte des éléments communs à la question et à l'article choisi.

2.2 Notions préliminaires et outils utilisés

Avant de pouvoir construire l'annotateur, une partie non négligeable du projet consistait à se familiariser avec la terminologie propre au traitement du langage naturel, l'architecture UIMA, le système unifié de langage médical (UMLS®) et le système cTAKES, par l'entremise de leur documentation.

2.2.1 Apache Unstructured Information Management Architecture (UIMA)

L'Apache Unstructured Information Management (UIMA) est non seulement une architecture, mais aussi une infrastructure d'application et un système de composantes où les interfaces sont décrites par des fichiers XML. Plusieurs outils sont aussi fournis dans le cadre du projet Apache. L'un d'eux, qui permet de transformer les composantes en services, a d'ailleurs été utilisé lors de la réalisation du prototype. À des fins de familiarisation avec cette architecture, le suivi d'un tutoriel disponible dans la documentation ainsi que l'installation locale de l'infrastructure d'application et de la trousse de développement logiciel Java UIMA furent effectués.

2.2.2 Unified Medical Language System® (UMLS®)

L'Unified Medical Language System® (UMLS®) est une initiative de l'U.S. National Library of Medicine (NIH) et est un système visant, entre autres, l'interopérabilité entre les systèmes d'information biomédicaux [5]. Ce système, constitué de termes, de systèmes de classification et de standards de codification comprend des millions de concepts issus d'une centaine de sources qui sont disponibles à travers le Metathesaurus. Chaque terme peut correspondre à plusieurs concepts et chacun de ces concepts appartient à au moins un type sémantique. Chaque terme est identifié par un identifiant unique intitulé LUI (*Term Unique Identifier*) et chacune de ses formes par un SUI (*String Unique Identifier*) [6]. Tous les concepts et tous les types sémantiques sont uniques et identifiés par un CUI (*Concept Unique Identifier*) et un TUI (*Type Unique Identifier*), respectivement. D'autres systèmes de codifications sont aussi présents dans le Metathesaurus, mais ne sont pas explicitement utilisés dans le cadre de ce projet.

2.2.3 Apache clinical Text Analysis and Knowledge Extraction System (cTAKES)

L'Apache clinical Text Analysis and Knowledge Extraction System (cTAKES) est un système de traitement du langage naturel destiné à extraire l'information des dossiers médicaux informatisés qui emploie des méthodes basées sur des règles et des algorithmes d'apprentissage et qui est basé sur l'architecture UIMA [4]. Ce système contient de nombreuses composantes et quelques problèmes furent rencontrés lors de l'installation de cet outil, particulièrement avec l'utilisation de l'outil Maven qui ne m'était pas familier : l'installation locale de cTAKES prit donc plus de temps que prévu. Une description des composantes de cTAKES employées dans le cadre de ce projet est incluse dans la section décrivant la réalisation de l'annotateur comme tel.

2.3 Approche choisie

La création d'un annotateur permettant d'analyser une question clinique et d'identifier des concepts clés dans des articles étant une tâche complexe, certaines décisions ont été prises afin de rendre sa réalisation plus aisée.

La restriction à des textes anglophones, tant pour les questions que pour les articles permet déjà une bonne simplification du problème. C'est que bien que des ressources existent pour le traitement de textes non structurés en d'autres langues que l'anglais, celles-ci sont beaucoup moins nombreuses.

Bien qu'au début du projet le type de question clinique devait être traité, cet aspect a été mis de côté en cours de route, afin, entre autres, de consacrer plus de temps à la réalisation du prototype. Le fait d'ignorer le type de question clinique a aussi permis de garder une définition plutôt intuitive des éléments P,I,C et O (tel que décrite dans l'introduction) et en a donc facilité le traitement. C'est que certains types sémantiques n'appartiennent pas toujours aux mêmes éléments P,I,C et O, dépendamment du type de question posée [3].

L'utilisation d'un maximum de composantes de cTAKES et du dictionnaire d'UMLS pour identifier des types sémantiques a permis de consacrer plus de temps à l'annotation des éléments P,I,C et O, comme la majorité des autres concepts importants était déjà

annotée à l'aide de ces outils. En effet, l'utilisation seule des fonctionnalités existantes permet déjà de détecter les phrases, les mots, et les symboles, à départager les verbes des noms et à transformer les mots sous forme canonique [7] en plus de détecter des traitements, des symptômes et des maladies.

3 Réalisation de l'annotateur

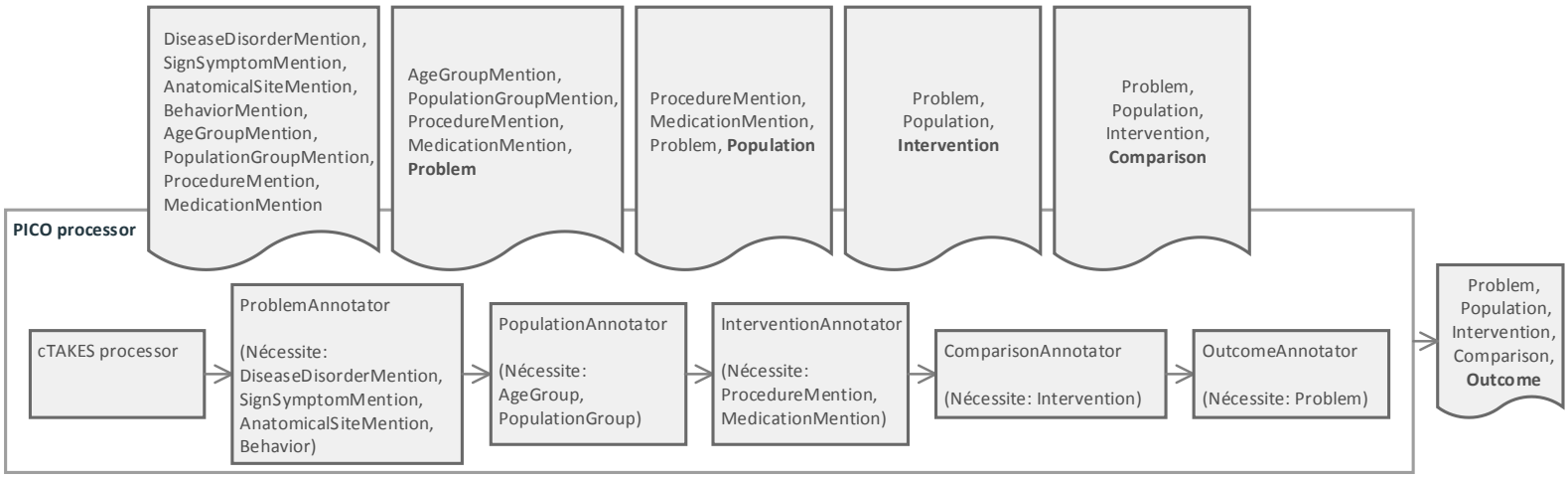
3.1 Recherche et analyse de questions cliniques

Trois banques de questions ont été parcourues afin d'identifier des questions types et de repérer certains patrons : *Cochrane Clinical Answers* [8], *The Journal of Family Practice* [9] et *Parkhurst Exchange* [10]. Comme le type de question n'a finalement pas été étudié dans le cadre de ce projet, aucun patron n'a vraiment été observé. Cependant, le parcours d'une centaine de questions a permis d'identifier certains termes permettant de repérer des comparaisons et des issues cliniques.

3.2 Structure globale

Afin de pouvoir mettre à jour facilement les composantes de cTAKES, il a été choisi de placer toutes ces composantes dans un annotateur agrégé nommé cTAKES processor. Ainsi, peu importe les annotateurs simples employés dans cet annotateur agrégé, il est attendu que certains types sémantiques soient identifiés à sa sortie. On pourra donc éventuellement optimiser ces composantes sans être obligé de modifier en même temps les annotateurs qui correspondent au cadre PICO (et inversement). La structure globale de l'annotateur développé est présentée ci-dessous à la figure 1. Il est à noter qu'il est nécessaire d'utiliser une version additionnée de l'annotateur *DB_consumer* (qui provient aussi de cTAKES), à la toute fin de l'annotateur *PICO processor*, pour pouvoir automatiquement enregistrer les documents annotés, ainsi que leurs annotations associées, dans la base de données locale.

Figure 1: Structure de l'annotateur développé



3.3 Description des types utilisés

Le système de types utilisé par l'annotateur développé, défini dans le fichier XML *resources/pico/typesystem/types/TypeSystem.xml*, utilise celui créé dans cTAKES, auquel sont ajoutés quelques types. D'abord, les concepts de groupe d'âge, de comportement et de groupe de population ont été introduits, sous les noms *AgeGroupMention*, *BehaviorMention* et *PopulationGroupMention*, respectivement. L'âge et le comportement sont quelques-uns des concepts proposés par Xiaoli Huang [3] alors que la classe sémantique «groupe», quant à elle, est directement tirée des travaux de Demner-Fushman et Lin [11] et de Cohen et Demner-Fushman [12]. Les différents éléments du cadre PICO y ont aussi été ajoutés, tout en divisant l'élément P en deux sous aspects : la population et le problème, tel qu'également proposé par Xiaoli Huang [3]. Les types créés correspondant aux éléments PICO sont donc *Population, Problem, Intervention, Comparison, et Outcome*. Un sixième type employé pour le cadre PICO, *PicoElement*, se veut un type abstrait duquel tous les éléments PICO sont issus. Ceci facilite entre autres la conversion d'un type d'annotation à un autre et à ne définir qu'une fois les attributs caractéristiques de tous les éléments PICO.

Ces attributs caractéristiques de tous les éléments PICO sont dénommés aspect, tui et cui. Les attributs tui et cui sont simplement les TUI et CUI du concept trouvé dans le dictionnaire UMLS® qu'ils contiennent. L'attribut aspect contient un mot décrivant

brièvement le type sémantique du concept contenu dans l'annotation. Par exemple, une annotation de type *Intervention* qui remplace une annotation de type *MedicationMention* aura une valeur de "*drug*" pour l'attribut aspect.

3.4 Choix des composantes de cTAKES

Les composantes choisies de cTAKES se trouvent dans l'annotateur agrégé *cTAKES processor* (mis à part l'annotateur permettant de stocker les documents et annotations dans la base de données locale – *DBConsumer.*) Après quelques tests, la liste de composantes comprises dans l'annotateur *AggregatePlaintextUMLSProcessor*, de la composante *ctakes-ytex-uima* a été adoptée presque qu'intégralement. Seulement certains paramètres ont été changés, dont ceux concernant l'enregistrement des annotations générées et les types d'annotations à retourner. C'est que cette liste de composantes, à la différence de celle comprise dans la composante *ctakes-clinical-pipeline* permet de chercher dans des dictionnaires insérés dans une base de données locale et inclut un annotateur permettant de choisir le concept UMLS® le plus approprié lorsque plusieurs concepts correspondent à l'expression cherchée.

3.5 Modifications apportées aux composantes choisies

3.5.1 Recherche dans le dictionnaire UMLS® local

Afin de pouvoir chercher les nouveaux concepts d'âge, de comportement et de groupe de population dans le dictionnaire UMLS® installé localement, le fichier *resources/org/apache/ctakes/ytex/dictionary/lookup/LookupDesc_SNOMED.xml* a dû être modifié. C'est que des propriétés ont été ajoutées au *lookupConsumer* dénommé *UmlsToSnomedDbConsumerImpl* à l'aide de ce fichier. Des propriétés correspondant aux TUI et CUI ont été ajoutées afin de placer ces informations comme attributs des éléments PICO qui sont créés plus tard à partir des types sémantiques. D'autres attributs, qui correspondent aux concepts ajoutés, servent à spécifier quels types sémantiques du dictionnaire correspondent à ces concepts. Le tableau 1 ci-dessous présente les attributs ajoutés dans ce fichier qui correspondent aux concepts introduits.

Tableau 1 : Concepts ajoutés à la recherche dans le dictionnaire UMLS® local

Concept (type de l'annotation)	**Nom de la propriété associée**	**Types sémantiques**
Comportement (BehaviorMention)	behaviorTuis	T053 (*Behavior*) T054 (*Social Behavior*) T055 (*Individual Behavior*) T056 (*Daily or Recreational Activity*)
Groupe d'âge (AgeGroupMention)	ageGroupTuis	T100 (*Age Group*)
Groupe de la population (PopulationGroupMention)	populationGroupTuis	T098 (*Population Group*) T101 (*Patient or Disabled Group*)

Le choix des types sémantiques correspondants aux divers concepts n'a pas fait l'objet d'une analyse poussée et pourrait être revu dans le but d'améliorer l'annotateur. Il est aussi à noter que les TUI correspondants aux médicaments ont aussi été ajoutés dans ce même fichier et qu'ils ont simplement été copiés du fichier *resources/org/apache/ctakes/dictionary/lookup/LookupDesc_Db.xml*.

Afin de pouvoir effectuer la recherche dans le dictionnaire local de ces nouveaux types sémantiques, ceux-ci ont aussi dû être ajoutés à la table *v_snomed_fword_lookup* (voir l'annexe «ajout de types sémantiques à la recherche»).

3.5.2 Utilisation de la base de données YTEX

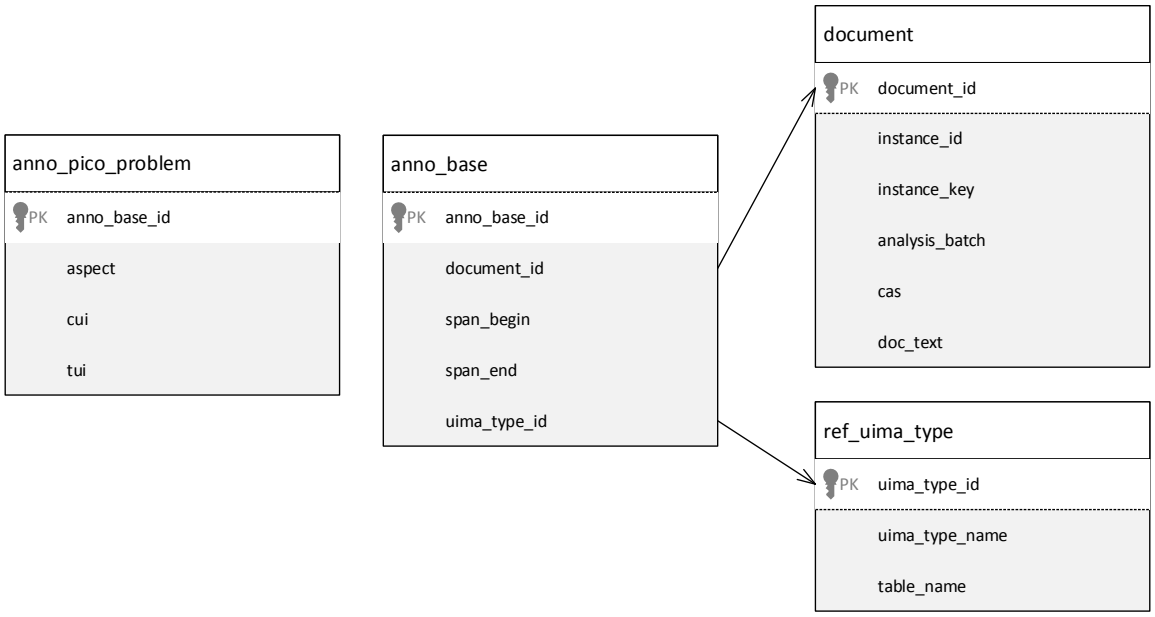
Afin de pouvoir enregistrer les nouveaux types d'annotations créés dans le cadre de ce projet, la table *ref_uima_type,* de la base de données locale YTEX, a d'abord été modifiée et des tables correspondants aux éléments P,I,C et O furent créées. Le tableau 2 suivant présente les entrées ajoutées dans la table *ref_uima_type*, de même que le nom des tables créées.

Tableau 2 : Entrées ajoutées à la table «ref_uima_type» de la base de données locale «ytex»

Type créé	**uima_type_id**	**uima_type_name**	**table_name**
Problem	500	pico.typesystem.type.picoelements.Problem	anno_pico_problem
Population	501	pico.typesystem.type.picoelements.Population	anno_pico_population
Intervention	502	pico.typesystem.type.picoelements.Intervention	anno_pico_intervention
Comparison	503	pico.typesystem.type.picoelements.Intervention	anno_pico_comparison
Outcome	504	pico.typesystem.type.picoelements.Outcome	anno_pico_outcome

La figure 2 ci-dessous présente non seulement la structure de la table *anno_pico_problem*, mais aussi celle des tables utilisées pour l'enregistrement des documents analysés et de leurs annotations associées.

Figure 2 : Structure des tables utilisées par l'annotateur pour l'enregistrement des documents et annotations dans la base de données locale



Il est à noter que les tables *anno_pico_population*, *anno_pico_intervention, anno_pico_comparison* et *anno_pico_outcome* ont la même structure que *anno_pico_problem*. De plus, la structure et la nomenclature des tables ainsi que cette façon de sauvegarder les attributs des annotations ont été choisies afin de s'harmoniser avec la façon de faire de cTAKES.

Le fichier *resources/org/apache/ctakes/ytex/uima/beans-uima-mapper.xml* a aussi dû être modifié afin de sauvegarder correctement les annotations représentant les concepts *Problem, Population, Intervention, Comparison* et *Outcome* dans la base de données locale. C'est ce fichier qui indique quel attribut de l'annotation devrait se trouver dans quelle colonne de quelle table dans la base de données.

3.6 Annotateurs simples développés

3.6.1 Problème

Cet annotateur simple, *ProblemAnnotator*, sert à reconnaître le problème visé par la question clinique. Pour ce faire, les types sémantiques correspondants aux maladies, aux symptômes, aux parties anatomiques et aux comportements du patient sont directement annotés comme décrivant le problème, à l'aide d'un itérateur pour chacun des types visés.

3.6.2 Population

L'annotateur *PopulationAnnotator* fonctionne de la même façon que l'annotateur servant à identifier le problème, à la différence qu'il recherche et utilise les annotations de types *AgeGroupMention* et *PopulationGroupMention*.

3.6.3 Intervention

L'annotateur *InterventionAnnotator* est construit de la même façon que les deux annotateurs précédents. Il emploie les annotations de types *MedicationMention* et *ProcedureMention*.

3.6.4 Comparaison

L'annotateur *ComparisonAnnotator* parcourt les annotations *Intervention* en cherchant deux annotations consécutives de ce type à l'intérieur d'un intervalle de caractères passé en paramètre. Lorsque deux interventions consécutives répondent à ce critère, on cherche alors dans l'espace entre les deux interventions un terme ou une expression indiquant une comparaison. Lorsque cette expression est trouvée, la deuxième annotation *Intervention* devient alors une annotation *Comparison*, de même que toutes les annotations du texte renvoyant au même concept. C'est qu'il est peu probable qu'à chaque fois que l'on parle de la comparaison dans le texte, qu'une expression indiquant la comparaison soit aussi présente. Il est à noter que cet annotateur ne fonctionne bien qu'avec une comparaison entre deux interventions et qu'il faudra éventuellement le modifier pour prendre en compte plus de deux interventions comparées.

3.6.5 «Outcome» (issue clinique)

Afin d'identifier les issues cliniques mentionnées dans un article ou une question, la liste des annotations de type *Problem* est parcourue et, pour chaque annotation, un intervalle de caractères précédant l'annotation est vérifié afin de déterminer s'il contient un terme ou une expression modulant l'annotation *Problem*. La largeur de cet intervalle est passée en paramètre à l'annotateur et pourrait être optimisée dans le futur.

Lorsqu'une annotation *Problem* est précédée d'une expression la modulant, l'ensemble composé de l'expression et de l'annotation devient une nouvelle annotation *Outcome*. Les attributs de l'annotation *Problem* visée deviennent les attributs de la nouvelle annotation *Outcome*. Ainsi, lors de l'établissement de la correspondance entre une question et un article, on pourra déterminer qu'on mentionne la modification des mêmes symptômes, par exemple. Il pourrait être intéressant éventuellement de voir s'il ne serait pas préférable d'établir la correspondance non seulement sur le problème modifié, mais aussi sur la nature de la modification.

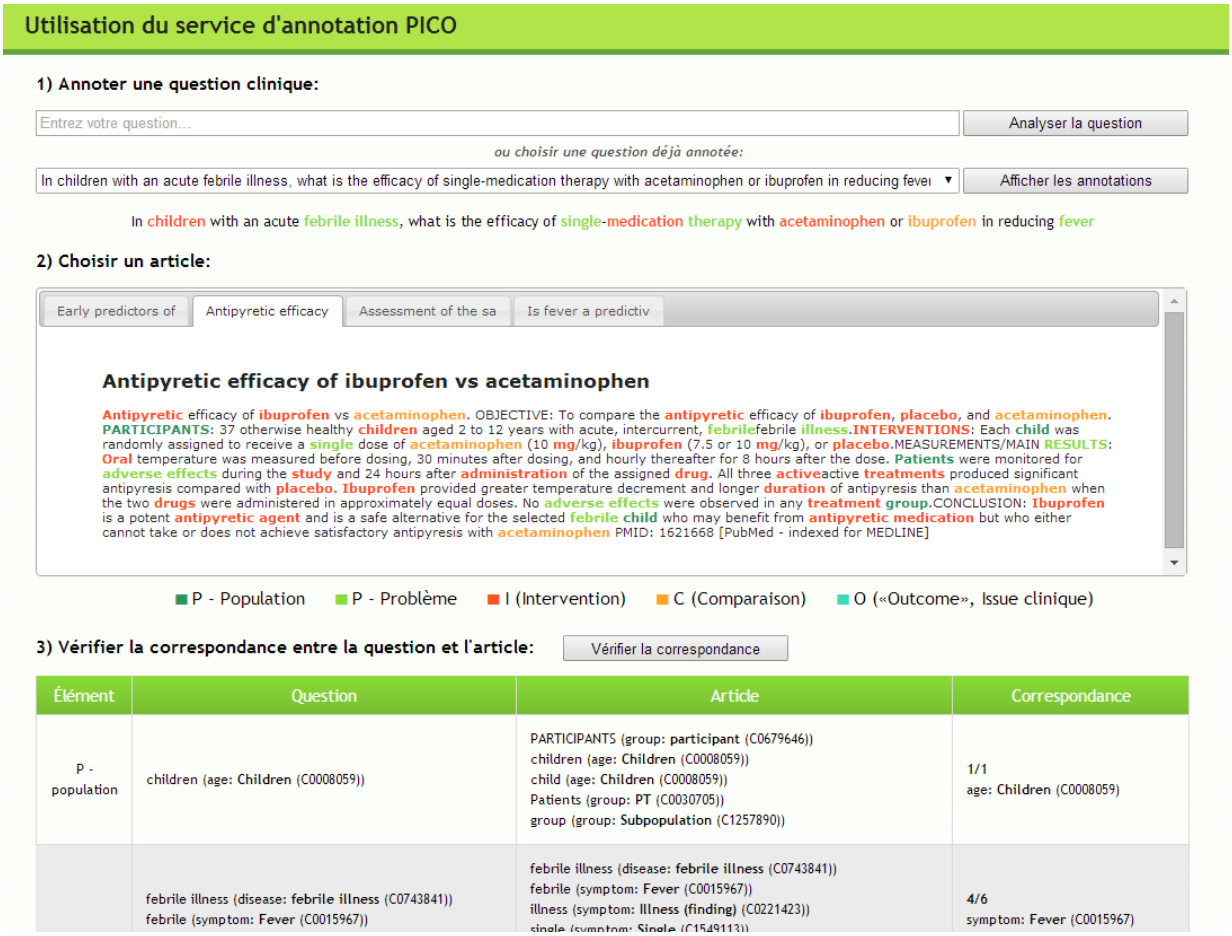
4 Réalisation du prototype

Afin de démontrer les capacités de l'annotateur développé, un prototype constitué d'une interface web et d'un service d'annotation fut réalisé. Bien qu'il ait été prévu au départ d'utiliser l'outil de recherche sémantique mentionné dans le tutoriel d'utilisation de l'UIMA [13] pour établir la correspondance entre une question et un article annotés, cet outil n'étant plus disponible au moment de la création de l'annotateur, de simples requêtes SQL sont plutôt employées à cette fin. L'utilisation de requêtes SQL limite cependant à une correspondance exacte ou nulle entre les concepts évalués.

Le service d'annotation est de type REST et utilise l'UIMA Simple Server [14]. Il est appelé via un appel Ajax de l'interface lorsque l'utilisateur demande l'annotation d'une question. Les résultats du service d'annotation sont alors retournés dans format XML avant d'être transformés et affichés dans un format plus visuel où chaque type d'annotation est représenté par une couleur différente.

Dans cette interface, les articles sont, quant à eux, déjà annotés, étant donné le temps d'exécution de l'annotateur. Lorsque l'utilisateur demande la correspondance entre l'article choisi et la question annotée, tous les concepts identifiés dans les documents sont alors affichés, de même que ceux présents dans les deux documents. La figure 3 ci-dessous présente une capture d'écran présentant l'interface web après une analyse de correspondance.

Figure 3 : Capture d'écran du prototype



5 Résultats obtenus

La figure 3 présentée à la page précédente montre des résultats plutôt typiques pour l'annotateur. C'est-à-dire que la plupart des concepts importants et facilement discernables dans des textes relativement simples sont correctement identifiés. Cependant, plusieurs concepts erronés sont aussi identifiés, tel qu'un dosage mesuré en mg identifié comme la présence de magnésium. Il est à noter que dans l'exemple présenté à la page précédente la comparaison («or ibuprofen») et l'issue clinique («reducing fever») avaient été correctement identifiées dans la question (tel que reflété dans le tableau de correspondance), mais qu'étant donné que plusieurs annotations couvrent les mêmes mots, seules les annotations intervention et problème sont visibles dans ce cas. Il semble donc que l'annotateur réalisé dans le cadre de ce projet présente des résultats acceptables à cette étape-ci de son développement, même s'il demande encore beaucoup de travail avant d'être intégré dans une application telle EBMPICO.

6 Recommandations

Afin d'améliorer l'annotateur, il est entre autres recommandé de traiter les différents types de questions cliniques et d'évaluer l'annotation de plusieurs articles et questions afin de reconnaître d'éventuels patrons.

Dans le but d'implémenter l'annotateur dans une application visant la recherche d'articles médicaux pertinents à une problématique clinique, un algorithme de tri d'articles basé sur le degré de correspondance entre une question et une série d'articles serait probablement fort utile.

7 Conclusion

En conclusion l'objectif principal du travail voulant la création d'un annotateur qui permet d'analyser la question clinique et d'identifier des concepts clés dans des articles a somme toute été atteint bien que ce projet soit relié à un problème très complexe qui mérite encore beaucoup de travail.

8 Bibliographie

[1] D. L. Sackett, W. M. Rosenberg, J. A. Gray, R. B. Haynes, and W. S. Richardson, "Evidence based medicine: what it is and what it isn't," *BMJ,* vol. 312, pp. 71-2, Jan 13 1996.

[2] Département d'informatique et d'ingénierie and U. d. Q. e. Outaouais. (2014, 14 mai 2014). *PICO*. Disponible: http://larip.uqo.ca/ebmpico/index.php

[3] M. Xiaoli Huang, Jimmy Lin, Ph.D., and Dina Demner-Fushman, M.D., Ph.D., "Evaluation of PICO as a Knowledge Representation for Clinical Questions," in *AMIA 2006 Symposium Proceedings*, 2006, p. 359.

[4] The Apache Software Foundation. (2013, 9 mai 2014). *cTAKES*. Disponible: https://ctakes.apache.org/index.html

[5] U.S. National Library of Medicine. (2013, 16 mai 2014). *Unified Medical Language System® (UMLS®) - UMLS Quick Start Guide*. Disponible: http://www.nlm.nih.gov/research/umls/quickstart.html

[6] U.S. National Library of Medicine. (2009, 16 mai 2014). *Unified Medical Language System® (UMLS®) - UMLS® Reference Manual [Internet].* Disponible: http://www.ncbi.nlm.nih.gov/books/NBK9684/

[7] J. Masanz. (2013, 22 mai 2014). *cTAKES 3.1 Component Use Guide*. Disponible: https://cwiki.apache.org/confluence/display/CTAKES/cTAKES+3.1+Component+Use+Guide

[8] The Cochrane Library. (2014, 25 juin 2014). *Cochrane Clinical Answers*. Disponible: http://cochraneclinicalanswers.com/

[9] The Journal of Family Practice. (2014, 25 juin 2014). *Clinical Inquiries*. Disponible: http://www.jfponline.com/articles/clinical-inquiries.html

[10] Parkhurst Publishing. (2014, 25 juin 2014). *Parkhurst Exchange*. Disponible: http://www.parkhurstexchange.com/searchQA

[11] D. Demner-Fushman and J. Lin, "Answering Clinical Questions with Knowledge-Based and Statistical Techniques," *Computational Linguistics,* vol. 33, pp. 63-103, 2007/03/01 2007.

[12] K. B. Cohen and D. Demner-Fushman, *Biomedical natural language processing*. Amsterdam: John Benjamins Publishing Company, 2014.

[13] The Apache UIMA Development Community. (2007, 14 mai 2014). *UIMA Tutorial and Developers' Guides*. Disponible: https://uima.apache.org/downloads/releaseDocs/2.1.0-incubating/docs/html/tutorials_and_users_guides/tutorials_and_users_guides.html

[14] The Apache UIMA Development Community. (2011, 13 juillet 2014). *UIMA Simple Server User Guide*. Disponible: http://uima.apache.org/d/uima-addons-current/SimpleServer/simpleServerUserGuide.html

9 Annexes

9.1 Description de l'annotateur développé

9.1.1 PICOprocessor.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<analysisEngineDescription xmlns="http://uima.apache.org/resourceSpecifier">
  <frameworkImplementation>org.apache.uima.java</frameworkImplementation>
  <primitive>false</primitive>
  <delegateAnalysisEngineSpecifiers>
    <delegateAnalysisEngine key="cTAKES processor">
      <import location="../analysis_engine_ctakes/cTAKES processor.xml"/>
    </delegateAnalysisEngine>
  <delegateAnalysisEngine key="ProblemAnnotator">
      <import location="ProblemAnnotator.xml"/>
    </delegateAnalysisEngine>
  <delegateAnalysisEngine key="PopulationAnnotator">
      <import location="PopulationAnnotator.xml"/>
    </delegateAnalysisEngine>
  <delegateAnalysisEngine key="InterventionAnnotator">
      <import location="InterventionAnnotator.xml"/>
    </delegateAnalysisEngine>
  <delegateAnalysisEngine key="ComparisonAnnotator">
      <import location="ComparisonAnnotator.xml"/>
    </delegateAnalysisEngine>
  <delegateAnalysisEngine key="OutcomeAnnotator">
      <import location="OutcomeAnnotator.xml"/>
    </delegateAnalysisEngine>
  </delegateAnalysisEngineSpecifiers>
  <analysisEngineMetaData>
    <name>PICO processor</name>
    <description/>
    <version>1.0</version>
    <vendor/>
    <configurationParameters searchStrategy="language_fallback"/>
    <configurationParameterSettings/>
    <flowConstraints>
      <fixedFlow>
      <node>cTAKES processor</node>
      <node>ProblemAnnotator</node>
      <node>PopulationAnnotator</node>
      <node>InterventionAnnotator</node>
      <node>ComparisonAnnotator</node>
      <node>OutcomeAnnotator</node>
      </fixedFlow>
    </flowConstraints>
    <typePriorities/>
    <fsIndexCollection/>
    <capabilities>
      <capability>
        <inputs/>
        <outputs>
          <type
allAnnotatorFeatures="true">pico.typesystem.type.picoelements.Problem</type>
```

```
          <type
allAnnotatorFeatures="true">pico.typesystem.type.picoelements.Population</type>
          <type
allAnnotatorFeatures="true">pico.typesystem.type.picoelements.Intervention</type>
          <type
allAnnotatorFeatures="true">pico.typesystem.type.picoelements.Outcome</type>
          <type
allAnnotatorFeatures="true">pico.typesystem.type.picoelements.Comparison</type>
        </outputs>
        <languagesSupported/>
      </capability>
    </capabilities>
  <operationalProperties>
      <modifiesCas>true</modifiesCas>
      <multipleDeploymentAllowed>true</multipleDeploymentAllowed>
      <outputsNewCASes>false</outputsNewCASes>
    </operationalProperties>
  </analysisEngineMetaData>
  <resourceManagerConfiguration/>
</analysisEngineDescription>
```

9.1.2 cTAKES processor.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<analysisEngineDescription xmlns="http://uima.apache.org/resourceSpecifier">
  <frameworkImplementation>org.apache.uima.java</frameworkImplementation>
  <primitive>false</primitive>
  <delegateAnalysisEngineSpecifiers>
    <delegateAnalysisEngine key="SimpleSegmentAnnotator">
      <import location="SimpleSegmentAnnotator.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="SentenceDetectorAnnotator">
      <import location="SentenceDetectorAnnotator.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="TokenizerAnnotator">
      <import location="TokenizerAnnotator.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="LvgAnnotator">
      <import location="LvgAnnotator.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="ContextDependentTokenizerAnnotator">
      <import location="ContextDependentTokenizerAnnotator.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="POSTagger">
      <import location="POSTagger.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="Chunker">
      <import location="Chunker.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="AdjustNounPhraseToIncludeFollowingNP">
      <import location="AdjustNounPhraseToIncludeFollowingNP.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="AdjustNounPhraseToIncludeFollowingPPNP">
      <import location="AdjustNounPhraseToIncludeFollowingPPNP.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="DrugLookupWindowAnnotator">
      <import location="DrugLookupWindowAnnotator.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="DrugMentionAnnotator">
      <import location="DrugMentionAnnotator.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="DictionaryLookupAnnotator">
      <import location="DictionaryLookupAnnotator.xml"/>
    </delegateAnalysisEngine>
  <delegateAnalysisEngine key="ClearNLPDependencyParserAE">
      <import location="ClearNLPDependencyParserAE.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="ClearNLPSemanticRoleLabelerAE">
      <import location="ClearNLPSemanticRoleLabelerAE.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="ExtractionPrepAnnotator">
      <import location="ExtractionPrepAnnotator.xml"/>
    </delegateAnalysisEngine>
    <delegateAnalysisEngine key="LookupWindowAnnotator">
      <import location="LookupWindowAnnotator.xml"/>
    </delegateAnalysisEngine>
  <delegateAnalysisEngine key="AssertionMiniPipelineAnalysisEngine">
```

```
    <import location="AssertionMiniPipelineAnalysisEngine.xml"/>
  </delegateAnalysisEngine>
<delegateAnalysisEngine key="SenseDisambiguatorAnnotator">
    <import location="SenseDisambiguatorAnnotator.xml"/>
  </delegateAnalysisEngine>
</delegateAnalysisEngineSpecifiers>
<analysisEngineMetaData>
  <name>cTAKES processor</name>
  <description/>
  <version>1.0</version>
  <vendor/>
  <configurationParameters searchStrategy="language_fallback">
    <configurationParameter>
      <name>ChunkCreatorClass</name>
      <type>String</type>
      <multiValued>false</multiValued>
      <mandatory>true</mandatory>
      <overrides>
        <parameter>Chunker/ChunkCreatorClass</parameter>
      </overrides>
    </configurationParameter>
    </configurationParameters>
  <configurationParameterSettings>
    <nameValuePair>
      <name>ChunkCreatorClass</name>
      <value>
        <string>org.apache.ctakes.chunker.ae.PhraseTypeChunkCreator</string>
      </value>
    </nameValuePair>
  </configurationParameterSettings>
  <flowConstraints>
    <fixedFlow>
      <node>SimpleSegmentAnnotator</node>
      <node>SentenceDetectorAnnotator</node>
      <node>TokenizerAnnotator</node>
      <node>LvgAnnotator</node>
      <node>ContextDependentTokenizerAnnotator</node>
      <node>POSTagger</node>
      <node>Chunker</node>
      <node>AdjustNounPhraseToIncludeFollowingNP</node>
      <node>AdjustNounPhraseToIncludeFollowingPPNP</node>
      <node>LookupWindowAnnotator</node>
      <node>DictionaryLookupAnnotator</node>
      <node>SenseDisambiguatorAnnotator</node>
    <node>ClearNLPDependencyParserAE</node>
      <node>ClearNLPSemanticRoleLabelerAE</node>
      <node>AssertionMiniPipelineAnalysisEngine</node>
    <node>ExtractionPrepAnnotator</node>
    </fixedFlow>
  </flowConstraints>
  <typePriorities>
    <priorityList>
      <type>org.apache.ctakes.typesystem.type.textspan.Segment</type>
      <type>org.apache.ctakes.typesystem.type.textspan.Sentence</type>
      <type>org.apache.ctakes.typesystem.type.syntax.BaseToken</type>
```

```
      </priorityList>
      <priorityList>
        <type>org.apache.ctakes.typesystem.type.textspan.Sentence</type>

<type>org.apache.ctakes.typesystem.type.textsem.IdentifiedAnnotation</type>
      </priorityList>
    </typePriorities>
    <fsIndexCollection/>
    <capabilities>
      <capability>
        <inputs/>
        <outputs>
          <type allAnnotatorFeatures="true">
org.apache.ctakes.typesystem.type.textsem.AnatomicalSiteMention</type>
          <type allAnnotatorFeatures="true">
org.apache.ctakes.typesystem.type.textsem.SignSymptomMention</type>
          <type allAnnotatorFeatures="true">
org.apache.ctakes.typesystem.type.textsem.ProcedureMention</type>
          <type allAnnotatorFeatures="true">
org.apache.ctakes.typesystem.type.textsem.MedicationMention</type>
          <type allAnnotatorFeatures="true">
org.apache.ctakes.typesystem.type.textsem.DiseaseDisorderMention</type>
        <type allAnnotatorFeatures="true">
org.apache.ctakes.typesystem.type.textsem.EntityMention</type>
        <type allAnnotatorFeatures="true">
pico.typesystem.type.textsem.BehaviorMention</type>
        <type allAnnotatorFeatures="true">
pico.typesystem.type.textsem.PopulationGroupMention</type>
          <type allAnnotatorFeatures="true">
pico.typesystem.type.textsem.AgeGroupMention</type>
          <type allAnnotatorFeatures="true">
org.apache.ctakes.typesystem.type.textspan.Sentence</type>
          <type allAnnotatorFeatures="true">
org.apache.ctakes.typesystem.type.syntax.BaseToken</type>
          <type allAnnotatorFeatures="true">
org.apache.ctakes.typesystem.type.textsem.IdentifiedAnnotation</type>
          <type allAnnotatorFeatures="true">
org.apache.ctakes.typesystem.type.syntax.WordToken</type>
        </outputs>
        <languagesSupported/>
      </capability>
    </capabilities>
  <operationalProperties>
      <modifiesCas>true</modifiesCas>
      <multipleDeploymentAllowed>true</multipleDeploymentAllowed>
      <outputsNewCASes>false</outputsNewCASes>
    </operationalProperties>
  </analysisEngineMetaData>
  <resourceManagerConfiguration/>
</analysisEngineDescription>
```

9.2 Ajout de types sémantiques à la recherche

9.2.1 Fichier LookupDesc_SNOMED.xml : UmlsToSnomedDbConsumerImpl

```
<lookupConsumer
        className="org.apache.ctakes.dictionary.lookup.ae.UmlsToSnomedDbConsumerImpl">
  <properties>
    <property key="codingScheme" value="SNOMED" />
    <property key="codeMetaField" value="cui" />
    <property key="cuiMetaField" value="cui" />
    <property key="tuiMetaField" value="tui" />
    <!-- Ajoutés -->
    <property key="behaviorTuis" value="T053,T054,T055,T056" />
    <property key="ageGroupTuis" value="T100" />
    <property key="populationGroupTuis" value="T098, T101" />
    <!-- Par défaut, lookupDesc de Ytex -->
    <!-- the following values are used as TUIs for testing: T_AS, T_PR, T_DD, T_SS -->
    <property key="anatomicalSiteTuis"
        value="T021,T022,T023,T024,T025,T026,T029,T030,T_AS" />
    <property key="procedureTuis" value="T059,T060,T061,T_PR" />
    <property key="disorderTuis"
        value="T019,T020,T037,T046,T047,T048,T049,T050,T190,T191,T_DD" />
    <property key="findingTuis"
        value="T033,T034,T040,T041,T042,T043,T044,T045,T046,T056,T057,T184,T_SS" />
    <!-- Par défaut, lookupDesc de drug -->
    <property key="medicationTuis"
        value="T073,T103,T109,T110,T111,T115,T121,T122,T123,T130,T168,T192,T195,
        T197,T200,T203" />
    <property key="dbConnExtResrcKey" value="DbConnection" />
    <property key="mapPrepStmt" value="select cui from mrconso where cui=?" />
  </properties>
</lookupConsumer>
```

9.2.2 Requête SQL

```
/*
Cette requête insère de nouveaux types sémantiques dans
v_snomed_fword_lookup à partir du dictionnaire UMLS local. C'est une
requête utilisée lors de l'installation de YTEX qui a été modifiée.
*/
INSERT IGNORE INTO
v_snomed_fword_lookup (cui, tui, fword, fstem, tok_str, stem_str)
SELECT mrc.cui, t.tui, c.fword, c.fstem, c.tok_str, c.stem_str
FROM umls_aui_fword c
INNER JOIN umls.MRCONSO mrc
ON c.aui = mrc.aui and mrc.SAB
      INNER JOIN
      (SELECT cui, min(tui) tui
        FROM umls.MRSTY sty
        GROUP BY cui) t ON t.cui = mrc.cui
```

9.3 Enregistrement des nouveaux types dans la base de données

9.3.1 Fichier beans-uima-mapper.xml: ajouts

```
<bean class="org.apache.ctakes.ytex.uima.mapper.AnnoMappingInfo">
  <property name="annoClassName"
    value="pico.typesystem.type.picoelements.Problem" />
  <property name="tableName" value="anno_pico_problem" />
  <property name="columnMappingInfos">
    <set>
      <bean
        class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
        <property name="annoFieldName" value="aspect" />
        <property name="columnName" value="aspect" />
        <property name="jxpath" value="value" />
      </bean>
      <bean
        class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
        <property name="annoFieldName" value="cui" />
        <property name="columnName" value="cui" />
        <property name="jxpath" value="value" />
      </bean>
      <bean
        class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
        <property name="annoFieldName" value="tui" />
        <property name="columnName" value="tui" />
        <property name="jxpath" value="value" />
      </bean>
    </set>
  </property>
</bean>
<bean class="org.apache.ctakes.ytex.uima.mapper.AnnoMappingInfo">
  <property name="annoClassName"
    value="pico.typesystem.type.picoelements.Population" />
  <property name="tableName" value="anno_pico_population" />
  <property name="columnMappingInfos">
    <set>
      <bean
        class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
        <property name="annoFieldName" value="aspect" />
        <property name="columnName" value="aspect" />
        <property name="jxpath" value="value" />
      </bean>
      <bean
        class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
        <property name="annoFieldName" value="cui" />
        <property name="columnName" value="cui" />
        <property name="jxpath" value="value" />
      </bean>
      <bean
        class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
        <property name="annoFieldName" value="tui" />
        <property name="columnName" value="tui" />
        <property name="jxpath" value="value" />
      </bean>
    </set>
```

```
  </property>
</bean>
<bean class="org.apache.ctakes.ytex.uima.mapper.AnnoMappingInfo">
  <property name="annoClassName"
    value="pico.typesystem.type.picoelements.Intervention" />
  <property name="tableName" value="anno_pico_intervention" />
  <property name="columnMappingInfos">
    <set>
      <bean
        class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
        <property name="annoFieldName" value="aspect" />
        <property name="columnName" value="aspect" />
        <property name="jxpath" value="value" />
      </bean>
      <bean
        class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
        <property name="annoFieldName" value="cui" />
        <property name="columnName" value="cui" />
        <property name="jxpath" value="value" />
      </bean>
      <bean
        class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
        <property name="annoFieldName" value="tui" />
        <property name="columnName" value="tui" />
        <property name="jxpath" value="value" />
      </bean>
    </set>
  </property>
</bean>
<bean class="org.apache.ctakes.ytex.uima.mapper.AnnoMappingInfo">
  <property name="annoClassName"
    value="pico.typesystem.type.picoelements.Comparison" />
  <property name="tableName" value="anno_pico_comparison" />
  <property name="columnMappingInfos">
    <set>
      <bean
        class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
        <property name="annoFieldName" value="aspect" />
        <property name="columnName" value="aspect" />
        <property name="jxpath" value="value" />
      </bean>
      <bean
        class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
        <property name="annoFieldName" value="cui" />
        <property name="columnName" value="cui" />
        <property name="jxpath" value="value" />
      </bean>
      <bean
        class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
        <property name="annoFieldName" value="tui" />
        <property name="columnName" value="tui" />
        <property name="jxpath" value="value" />
      </bean>
    </set>
  </property>
```

```
      </bean>
      <bean class="org.apache.ctakes.ytex.uima.mapper.AnnoMappingInfo">
        <property name="annoClassName"
          value="pico.typesystem.type.picoelements.Outcome" />
        <property name="tableName" value="anno_pico_outcome" />
        <property name="columnMappingInfos">
          <set>
            <bean
              class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
              <property name="annoFieldName" value="aspect" />
              <property name="columnName" value="aspect" />
              <property name="jxpath" value="value" />
            </bean>
            <bean
              class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
              <property name="annoFieldName" value="cui" />
              <property name="columnName" value="cui" />
              <property name="jxpath" value="value" />
            </bean>
            <bean
              class="org.apache.ctakes.ytex.uima.mapper.ColumnMappingInfo">
              <property name="annoFieldName" value="tui" />
              <property name="columnName" value="tui" />
              <property name="jxpath" value="value" />
            </bean>
          </set>
        </property>
      </bean>
    </set>
  </property>
</bean>
```

9.4 Code des annotateurs simples développés

9.4.1 UmlsEntryDerivAnnotationBuilder.java

```
/**
 * Cette classe contient des fonctions utilisées par les annotateurs qui
 * annotent les concepts PICO
 */
public class UmlsEntryDerivAnnotationBuilder {

  /**
   * Cette fonction sert à extraire d'une liste de concepts le concept identifié
   * comme étant le plus approprié (cette identification se fait par
   * l'annotateur cTAKES SenseDisambiguatorAnnotator, de la composante YTEX.)
   *
   * @param concepts
   *          une liste de concepts correspondant au texte annoté
   * @return le concept qui a été déterminé comme étant le plus approprié
   */
  public static OntologyConcept extractDisambConcept(FSArray concepts) {
    for (int i = 0; i < concepts.size(); i++) {
      OntologyConcept concept = (OntologyConcept) concepts.get(i);
      if (concept.getDisambiguated()) {
        return concept;
      }
    }
    return (OntologyConcept) concepts.get(0);
  }

  /**
   * Cette fonction permet de créer les annotations qui sont la transformation
   * de certains types sémantiques en un nouveau type Utilisé pour la création
   * des annotations Proble, Population et Intervention
   *
   * @param targetClass
   *          le type d'annotation à créer
   * @param index
   *          index d'annotations d'un certain type
   * @param aspect
   *          l'aspect du concept P ou I auquel correspond ce type sémantique
   * @param aJCas
   */
  public static void createAnnotations(Class<?> targetClass,
      AnnotationIndex<Annotation> index, String aspect, JCas aJCas) {
    FSIterator<Annotation> annotIter = index.iterator();
    PicoElement newAnnot = new PicoElement((JCas) aJCas, 0, 0);

    while (annotIter.hasNext()) {
      Annotation oldAnnot = annotIter.next();
      if (targetClass == Problem.class) {
        newAnnot = new Problem((JCas) aJCas, oldAnnot.getBegin(),
            oldAnnot.getEnd());
      } else if (targetClass == Population.class) {
        newAnnot = new Population((JCas) aJCas, oldAnnot.getBegin(),
            oldAnnot.getEnd());
      } else if (targetClass == Intervention.class) {
```

```
      newAnnot = new Intervention((JCas) aJCas, oldAnnot.getBegin(),
          oldAnnot.getEnd());
    }

    FSArray concepts = ((IdentifiedAnnotation) oldAnnot)
        .getOntologyConceptArr();
    OntologyConcept concept = extractDisambConcept(concepts);
    newAnnot.setCui(concept.getCode());
    newAnnot.setTui(((UmlsConcept) concept).getTui());
    newAnnot.setAspect(aspect);
    newAnnot.addToIndexes();
  }
}

/**
 * Cette fonction retourne toutes les annotations de type Intervention
 * correspondant à un concept dont l'identifiant unique est passé en paramètre
 *
 * @param aJCas
 *          JCas en cours
 * @param cui
 *          identifiant du concept visé
 * @return les annotations Intervention visées
 */

public static HashSet<Intervention> getInterventionsByCui(JCas aJCas,
    String cui) {
  AnnotationIndex<Annotation> index = aJCas
      .getAnnotationIndex(Intervention.type);
  FSIterator<Annotation> annotIter = index.iterator();
  HashSet<Intervention> listInterventionAnnot = new HashSet<Intervention>();

  while (annotIter.hasNext()) {
    Intervention oldAnnot = (Intervention) annotIter.next();
    if (oldAnnot.getCui().matches(cui)) {
      listInterventionAnnot.add(oldAnnot);
    }
  }

  return listInterventionAnnot;

}

/**
 * Permet de convertir une annotation Intervention passée en paramètre en
 * annotation Comparison
 *
 * @param aJCas
 *          JCas en cours
 * @param oldAnnot
 *          l'annotation Intervention qui est une comparaison
 */
public static void convertInterventiontoComparison(JCas aJCas,
    Intervention oldAnnot) {
  Comparison newAnnot = new Comparison(aJCas, oldAnnot.getBegin(),
```

```
        oldAnnot.getEnd());
    newAnnot.setAspect(oldAnnot.getAspect());
    newAnnot.setCui(oldAnnot.getCui());
    newAnnot.setTui(oldAnnot.getTui());
    newAnnot.addToIndexes();
    oldAnnot.removeFromIndexes();
  }

  /**
   * Permet de convertir toutes les annotation Intervention qui corresponde au
   * même concept que celui passé en paramètre en annotations Comparison
   *
   * @param aJCas
   *          JCas en cours
   * @param cui
   *          identifiant unique du concept visé
   */
  public static void convertInterventiontoComparison(JCas aJCas, String cui) {
    HashSet<Intervention> listInterventionAnnot = getInterventionsByCui(aJCas,
        cui);
    Iterator<Intervention> intervIt = listInterventionAnnot.iterator();
    while (intervIt.hasNext()) {
      Intervention oldAnnot = (Intervention) intervIt.next();
      convertInterventiontoComparison(aJCas, oldAnnot);
    }
  }

}
```

9.4.2 ProblemAnnotator.java

```
public class ProblemAnnotator extends JCasAnnotator_ImplBase {

  @Override
  public void process(JCas aJCas) throws AnalysisEngineProcessException {

    //Chaque annotation du type spécifié ici deviendra une annotation Problem
    UmlsEntryDerivAnnotationBuilder.createAnnotations(Problem.class,
      aJCas.getAnnotationIndex(DiseaseDisorderMention.type),"disease",aJCas);

    UmlsEntryDerivAnnotationBuilder.createAnnotations(Problem.class,
      aJCas.getAnnotationIndex(BehaviorMention.type),"behavior",aJCas);

    UmlsEntryDerivAnnotationBuilder.createAnnotations(Problem.class,
      aJCas.getAnnotationIndex(SignSymptomMention.type),"symptom",aJCas);

    UmlsEntryDerivAnnotationBuilder.createAnnotations(Problem.class,
      aJCas.getAnnotationIndex(AnatomicalSiteMention.type),"area",aJCas);
  }

}
```

9.4.3 PopulationAnnotator.java

```
public class PopulationAnnotator extends JCasAnnotator_ImplBase {

  @Override
  public void process(JCas aJCas) throws AnalysisEngineProcessException {

    //Chaque type spécifié ici deviendra une annotation Population

    UmlsEntryDerivAnnotationBuilder.createAnnotations(Population.class,
       aJCas.getAnnotationIndex(AgeGroupMention.type),"age",aJCas);

    UmlsEntryDerivAnnotationBuilder.createAnnotations(Population.class,
       aJCas.getAnnotationIndex(PopulationGroupMention.type),"group",aJCas);
  }

}
```

9.4.4 InterventionAnnoator.java

```
public class InterventionAnnotator extends JCasAnnotator_ImplBase {

  @Override
  public void process(JCas aJCas) throws AnalysisEngineProcessException {

  //Chaque type spécifié ici deviendra une annotation Intervention

    UmlsEntryDerivAnnotationBuilder.createAnnotations(Intervention.class,
       aJCas.getAnnotationIndex(MedicationMention.type),"drug",aJCas);

    UmlsEntryDerivAnnotationBuilder.createAnnotations(Intervention.class,
       aJCas.getAnnotationIndex(ProcedureMention.type),"procedure", aJCas);
  }

}
```

9.4.5 ComparisonAnnotator.java

```
/**
 * Annotateur qui détecte la présence deux interventions consécutives à
 * l'intérieur d'un intervalle et qui détermine une intervention alternative à
 * la première
 */
public class ComparisonAnnotator extends JCasAnnotator_ImplBase {
  // Intervalle de caractères maximal séparant les deux interventions
  private int mWindowSize;
  // Contient les mots clés indiquant une comparaison
  private HashSet<String> motsCles = new HashSet<String>();

  public void initialize(UimaContext aContext)
      throws ResourceInitializationException {
    super.initialize(aContext);
    // Aller chercher la valeur du paramètre indiquant l'intervalle maximal
    mWindowSize = ((Integer) aContext.getConfigParameterValue("WindowSize"))
        .intValue();
  }
```

```java
/**
 * Cette fonction détermine si un mot passé en paramètre est inscrit dans
 * motsCles, la liste de mots indiquant une comparaison.
 *
 * @param mot
 *          mot cherché dans la liste de mots clés indiquant une comparaison
 * @return true lorsque le mot est dans la liste, false sinon
 */
private boolean motsClesContains(String mot) {
  Iterator<String> motIter = motsCles.iterator();
  while (motIter.hasNext()) {
    if (motIter.next().equalsIgnoreCase(mot)) {
      return true;
    }
  }
  return false;
}

public void process(JCas aJCas) {
  // La liste des interventions
  FSIndex<Annotation> interventionIndex = aJCas
      .getAnnotationIndex(Intervention.type);
  String text = aJCas.getDocumentText();

  // position de la dernière comparaison pour éviter plusieurs annotations au
  // même endroit
  int lastComparison = -1;
  // Permet d'accéder à la première intervention de la paire
  Intervention previous = null;

  // Storer les concepts des comparaisons pour pouvoir transformer les
  // interventions correspondantes
  HashSet<String> compCuis = new HashSet<String>();

  // Ajouter ici les mots clés indiquant une comparaison
  motsCles.add("vs");
  motsCles.add("or");

  FSIterator<Annotation> interventionIter = interventionIndex.iterator();
  while (interventionIter.hasNext()) {
    // Première intervention de la paire
    Intervention intervention1 = null;
    // S'il s'agit de la première intervention du document
    if (previous == null) {
      intervention1 = (Intervention) interventionIter.next();
    } else {
      intervention1 = previous;
    }
    // S'il ne s'agit pas de la dernière intervention du document
    if (interventionIter.hasNext()) {
      Intervention intervention2 = (Intervention) interventionIter.next();

       /* Si l'intervalle entre les deux annotations est inférieur à
       * l'intervalle max passé en paramètre*/
```

```
        if (intervention2.getBegin() - intervention1.getEnd() < mWindowSize) {

           /* La deuxième intervention de la paire ne doit pas être située sur la
           * dernière comparaison*/
          if (intervention2.getBegin() > lastComparison) {

            // Les caractères entre les deux interventions
            String window = text.substring(intervention1.getEnd(),
                intervention1.getEnd() + mWindowSize);

            //Diviser cet espace en mots
            StringTokenizer tokenizer = new StringTokenizer(window,
                " \t\n\r.<.>/?\";:[{]}\\|=+()!");

            boolean motCle = false;

            //Vérifier si un de ces mots indique une comparaison
            while (tokenizer.hasMoreTokens()) {
              String token = tokenizer.nextToken();
              if (motsClesContains(token)) {
                motCle = true;
                break;
              }
            }

            //Si on est en présence d'une comparaison
            if (motCle) {
              //Créer la comparaison
              Comparison comparison = new Comparison(aJCas, intervention2.getBegin(),
                  intervention2.getEnd());
              //Ajouter ses attributs
              comparison.setAspect(intervention2.getAspect());
              comparison.setCui(intervention2.getCui());
              comparison.setTui(intervention2.getTui());
              comparison.addToIndexes();
              //Ajouter ce concept à liste de concepts à traiter
              compCuis.add(comparison.getCui());
              lastComparison = intervention2.getEnd();
            }
          }
        }
        previous = intervention2;
      }
    }

    // Transformer Interventions en comparaisons plus loin dans le texte
    Iterator<String> compIt = compCuis.iterator();
    while (compIt.hasNext()) {
      UmlsEntryDerivAnnotationBuilder.convertInterventiontoComparison(aJCas,
          compIt.next());
    }
  }

}
```

9.4.6 OutcomeAnnotator.java

```
/**
 * Annotateur qui détecte la présence de problèmes précédés d'un terme les
 * modulant et qui définit cet ensemble par une nouvelle annotation Outcome
 */
public class OutcomeAnnotator extends JCasAnnotator_ImplBase {
  // Intervalle de caractères devant contenir l'expression
  private int mWindowSize;
  // Contient les expressions régulières indiquant une issue clinique
  private HashSet<String> motsCles = new HashSet<String>();

  public void initialize(UimaContext aContext)
      throws ResourceInitializationException {
    super.initialize(aContext);
    // Aller chercher la valeur du paramètre indiquant l'intervalle
    mWindowSize = ((Integer) aContext.getConfigParameterValue("WindowSize"))
        .intValue();
  }

  /**
   * Cette fonction détermine si un mot passé en paramètre correspond à une
   * expression régulière comprise dans motsCles, la liste d'expressions
   * régulières décrivant la présence d'une issue clinique.
   *
   * @param mot
   *          mot cherché indiquant une issue clinique
   * @return true lorsque le mot est correspond à une expression dans la liste,
   *         false sinon
   */
  private boolean motsClesContains(String mot) {
    Iterator<String> motIter = motsCles.iterator();
    while (motIter.hasNext()) {
      Pattern p = Pattern.compile(motIter.next());
      Matcher m = p.matcher(mot);
      if (m.find()) {
        return true;
      }
    }
    return false;
  }

  public void process(JCas aJCas) {
    // La liste de problèmes
    FSIndex<Annotation> problemIndex = aJCas.getAnnotationIndex(Problem.type);
    String text = aJCas.getDocumentText();
    // Pour éviter que deux annotations couvre le même endroit
    int lastOutcome = -1;
    // Regex des termes caracterisant diminution du problème ou amélioration
    motsCles.add("reduc.*?");
    motsCles.add("dim.*?");
    motsCles.add("augm.*?");
    motsCles.add("ameli.*?");
    motsCles.add("high.*?");
    motsCles.add("low.*?");
    motsCles.add("eradic.*?");
```

```
    motsCles.add("reliev.*?");

    FSIterator<Annotation> problemIter = problemIndex.iterator();
    while (problemIter.hasNext()) {

      Problem problem = (Problem) problemIter.next();
      // Si on ne recouvre pas l'issue clinique précédente
      if (problem.getBegin() > lastOutcome) {
        //Stocke le début de l'annotation
        int trueBegin = 0;

        // Vérifier présence de mots - regarder x chars en arriere
        if (problem.getBegin() - mWindowSize > 0) {
          trueBegin = problem.getBegin() - mWindowSize;
        }
        //L'intervalle de texte cherché avant l'annotation Problem
        String window = text.substring(trueBegin, problem.getBegin());

        // Diviser en mots
        StringTokenizer tokenizer = new StringTokenizer(window,
            " \t\n\r.<.>/?\";:[{]}\\|=+()!");

        boolean motCle = false;

      //Vérifier si un de ces mots indique une issue clinique
        while (tokenizer.hasMoreTokens()) {
          String token = tokenizer.nextToken();
          if (motsClesContains(token)) {
            motCle = true;
            int newBegin = window.indexOf(token);
            //Permet d'ajuster le début de l'annotation finale
            if (newBegin != -1) {
              trueBegin += newBegin;
            }
            break;
          }
        }

        // Si issue clinique, créer la nouvelle annotation
        if (motCle) {
          Outcome outcome = new Outcome(aJCas, trueBegin, problem.getEnd());
          outcome.setAspect(problem.getAspect());
          outcome.setCui(problem.getCui());
          outcome.setTui(problem.getTui());
          outcome.addToIndexes();
          lastOutcome = problem.getEnd();
        }
      }
    }
  }

}
```

9.5 Correspondance entre un article et une question clinique

9.5.1 Requêtes SQL

```
/* Création de la procédure stockée getMatch qui retourne les concepts
présents à la fois dans la question et l'article choisis */

CREATE DEFINER=`root`@`localhost` PROCEDURE `getMatch`(IN `questionID`
INT(11), IN `articleID` INT(11), IN `typeID` INT(11))
    NO SQL
SELECT DISTINCT
      question.document_id AS Question_id,
      question.cui AS Question_annot_cui,
    getAspect(question.anno_base_id, typeID) AS Question_aspect,
    article.document_id AS Article_id,
      article.cui AS Article_annot_cui,
    getAspect(article.anno_base_id, typeID) AS Article_aspect,
    getPrefText(question.cui) AS cui_text


FROM v_annotation_pico question, v_annotation_pico article

WHERE       question.document_id = questionID
        AND article.document_id = articleID
        AND question.uima_type_id = typeID
        AND question.cui = article.cui

ORDER BY SUBSTR(question.cui FROM 1 FOR 1),CAST(SUBSTR(question.cui FROM
2) AS UNSIGNED)



/*Création de la vue v_annotation_pico*/

SELECT
`v_annotation`.`anno_base_id` AS `anno_base_id`,
`v_annotation`.`document_id` AS `document_id`,
`v_annotation`.`span_begin` AS `span_begin`,
`v_annotation`.`span_end` AS `span_end`,
`v_annotation`.`uima_type_id` AS `uima_type_id`,
`v_annotation`.`uima_type_name` AS `uima_type_name`,
`v_annotation`.`anno_text` AS `anno_text`,
`getCui`(`v_annotation`.`anno_base_id`,`v_annotation`.`uima_type_id`) AS `cui`,
`v_annotation`.`analysis_batch` AS `analysis_batch`
FROM `ytex`.`v_annotation`
```

```
/*Création de la fonction getCui*/

CREATE DEFINER=`root`@`localhost` FUNCTION `getCui`(`annoID` INT(11), `typeID`
INT(11)) RETURNS char(8) CHARSET utf8
    NO SQL
BEGIN

IF    typeID = 500
      THEN SET
      @cui=(
        SELECT cui FROM anno_pico_problem
        WHERE anno_base_id = annoID);

ELSEIF typeID = 501
      THEN SET
      @cui=(
        SELECT cui FROM anno_pico_population
        WHERE anno_base_id = annoID);

ELSEIF typeID = 502
      THEN SET
      @cui=(
        SELECT cui FROM anno_pico_intervention
        WHERE anno_base_id = annoID);

ELSEIF typeID = 503
      THEN SET
      @cui=(
        SELECT cui FROM anno_pico_comparison
        WHERE anno_base_id = annoID);

ELSEIF typeID = 504
      THEN SET
      @cui=(
        SELECT cui FROM anno_pico_outcome
        WHERE anno_base_id = annoID);

END IF;

RETURN @cui;

END
```