MINISTERUL EDUCAŢIEI ŞI CERCETĂRII ŞTIINŢIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Detectarea argumentelor din cadrul documentelor despre cancer utilizând procesarea limbajului natural şi ontologii

LUCRARE DE LICENŢĂ

Absolvent: **Oana Maria POPA**
Conducător ştiinţific: **Conf. dr. ing. Adrian GROZA**

2016

MINISTERUL EDUCAŢIEI ŞI CERCETĂRII ŞTIINŢIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Oana Maria POPA**

Detectarea argumentelor din cadrul documentelor despre cancer utilizând procesarea limbajului natural şi ontologii

1. **Enunţul temei:** *Implementarea unui sistem de detectare a argumentelor din cadrul documentelor ce conţin informaţii despre cancerul la sân. Sistemul va extrage structura fiecărui argument şi le va salva într-o ontologie, pe care utilizatorul o va putea interoga.*
2. **Conţinutul lucrării:** *Introducere - Contextul proiectului, Obiectivele Proiectului, Studiu Bibliografic, Analiză şi Fundamentare Teoretică, Proiectare de Detaliu şi Implementare, Testare şi Validare, Manual de Instalare şi Utilizare, Concluzii, Bibliografie, Anexe.*
3. **Locul documentării:** Intelligent System Group
4. **Consultanţi:** Conf. dr. ing. Adrian GROZA
5. **Data emiterii temei:** 1 Noiembrie 2015
6. **Date predării:** 30 Iunie 2016

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Declaraţie pe proprie răspundere privind autenticitatea lucrării de licenţă

Subsemnata Popa Oana Maria legitimată cu seria nr. CNP , autorul lucrării Detectarea argumentelor din cadrul documentelor despre cancer utilizând procesarea limbajului natural şi ontologii elaborată în vederea susţinerii examenului de finalizare a studiilor de licenţă la Facultatea de Automatică şi Calculatoare, Specializarea Calculatoare în limba Română din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea Iulie a anului universitar 2015-2016, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

În cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data

Nume, Prenume

_____

_____

Semnătura

Cuprins

Capitolul 1 Introducere - Contextul proiectului **1**
1.1 Sistem de detectare a argumentelor 2
1.2 Prezentare generală a ontologiilor 3
1.3 Structura tezei 3

Capitolul 2 Obiectivele Proiectului **5**
2.1 Specificarea problemei şi motivaţie 5
2.2 Obiectivele proiectului 6
2.3 Specificaţiile proiectului 6
2.3.1 Cerinţele funcţionale 6
2.3.2 Cerinţele nonfuncţionale 8

Capitolul 3 Studiu Bibliografic **9**
3.1 Logici de descriere 9
3.2 Detectarea argumentelor 10
3.3 Soluţii existente 12
3.3.1 ArguAna 12
3.3.2 Argument Workbench 12
3.4 Aplicaţii pentru procesarea limbajului natural 12

Capitolul 4 Analiză şi Fundamentare Teoretică **15**
4.1 Modelul argumentativ 15
4.2 Modelarea ontologiilor 19
4.2.1 Ontologie de argumentare 19
4.2.2 Ontologie de cancer la sân 21
4.3 Funcţionalitate 24
4.3.1 Prezentarea generală a fazelor sistemului 24
4.3.2 Funcţiile sistemului 25
4.4 Cazuri de utilizare 25
4.4.1 Identificarea argumentelor 26
4.4.2 Interogarea ontologiei 29
4.4.3 Vizualizarea adnotărilor 30

Capitolul 5 Proiectare de Detaliu şi Implementare **31**
5.1 Arhitectura sistemului 31
5.1.1 Prezentare generală a arhitecturii sistemului 31
5.1.2 Interacţiunile dintre componentele sistemului 32
5.2 Prezentare generală a sistemului 34
5.2.1 Diagrama de pachete 34
5.2.2 Diagrama de clase 35
5.3 Specificaţiile datelor de intrare şi ieşire 36
5.4 Componenta de procesare a textului 37
5.5 Componenta de identificare a argumentelor 39
5.5.1 Structura indicatorilor de concluzie 40
5.5.2 Structura indicatorilor de premisă 43
5.6 Componenta de cunoştinţe 46
5.7 Interfaţa grafică 49

Capitolul 6 Testare şi Validare **53**
6.1 Testarea sistemului de adnotare 53
6.2 Rezultatele obţinute 55
6.3 Testarea componentei de interogare 56

Capitolul 7 Manual de Instalare şi Utilizare **59**
7.1 Manual de instalare 59
7.2 Manual de utilizare 62
7.2.1 Rularea sistemului 62
7.2.2 Identificarea argumentelor 63
7.2.3 Interogarea ontologiei de argumente 65
7.2.4 Vizualizarea adnotărilor prin intermediul interfeţei GATE 65
7.2.5 Vizualizarea listelor de cuvinte .lst 67
7.2.6 Vizualizarea ontologiilor 67

Capitolul 8 Concluzii **71**
8.1 Contribuţii şi realizări 71
8.2 Posibilităţi de dezvoltare ulterioară 72

Bibliografie **73**

Anexa A Ontologii adnotate **75**

Anexa B Lucrări publicate **79**

Capitolul 1

Introducere - Contextul proiectului

Luând în considerare recentele rezultate contradictorii asupra cancerului publicate în diferite jurnale de ştiinţă şi natură, se poate observa aşa numitul "ghinion în ceea ce priveşte cancerul". Studiul efectuat în [1] susţine ideea că mutaţiile aleatorii în celulele sănătoase pot explica două treimi din cancer. Aceste rezultate sugerează că majoritatea cazurilor de cancer pot să fie prevenite.

O parte pozitivă a acestei "dezordini" a cancerului este că ajută pacienţii să ştie că nu e vina lor [2]. Cele mai interesante corelaţii descoperite de Tomasetti şi Vogelstein contrazic vechea piatră de temelie din articolul [3] lui Doll şi Peto argumentând că cele mai multe tipuri de cancer ar putea fi prevenite prin eliminarea diferitelor stiluri de viaţă. Pe altă parte însă, susţinătorii cancerului furnizează un set de contra-argumente împotriva lor [1], prin vocile lui Wodarz şi Zuaber [4].

Exemplul de mai sus este o bună instanţiere a problemelor care rezultă printr-un flux continuu de cercetare ştiinţifică, care conţine rezultate contradictorii susţinute de argumente şi contra-argumente. Deoarece expertiza medicală are loc la diferite niveluri, o parte din agenţii umani au dificultăţi în a face faţă cantităţii mari de studii, dar de asemenea şi în a înţelege motivele, bucăţile de probă pretinse atât de susţinători cât şi oponenţi în dezbaterile subiectului.

Pentru a înţelege mai bine argumentele suport pentru noile descoperiri realizate în domeniul medical avem nevoie de aplicaţii care să identifice argumente în articole ştiinţifice. Munca noastră are aici scopul de a umple acest gol în ceea ce priveşte domeniul tehnologic.

Destul de conştienţi de dificultatea acestei sarcini, ne îmbarcăm pe acest drum, bazându-ne pe intercalarea bine cunoscutului domeniu de cunoaştere a procesării limbajului natural. Pentru a formaliza domeniul medical existent ne bazăm pe ontologii. Pentru a structura modelul argumentării utilizăm expresivitatea şi capacităţile de raţionament a descrierii logice. Pentru a realiza identificarea argumentelor formalizăm diferite pattern-uri lingvistice într-o regulă bazată pe limbaj. Am testat soluţia noastră pe un corpus ştiinţific realizat din aticole care au ca şi punct de focus cancerul la sân.

În domeniul cancerului la sân, lunar apar din ce în ce mai multe articole, fiind o boală foarte răspândită de-a lungului femeilor de pe întregul glob. Recenta proliferare

a publicaţiilor on-line care prezintă ca subiect decoperirile medicale a creat nevoia critică pentru instrumente de acces la informaţii care ajută părţile interesate în domeniul medical.

Din cauza cantităţii mari de informaţii cu privire la un anumit subiect, extragerea datelor ("data mining") aduce un set de instrumente şi tehnici care pot fi aplicate asupra acestor date procesate pentru a descoperi modele ascunse. Acest lucru oferă profesioniştilor din domeniul sănătţii o sursă suplimentară de cunoştinţe pentru luarea deciziilor. Limitările actuale sau provocările în extragerea datelor pentru asistenţă medicală includ informaţii din surse eterogene, provocări sau valori lipsă, zgomot şi valori aberante.

Contextul proiectului este realizat în jurul domeniu numit argumentare având scopul de a ajuta profesioniştii din domeniul sănătăţii pentru sprijinirea procesului decizional. "Argumentation mining" implică detectarea automată a argumentelor existente în text, a relaţiilor dintre ele şi a structurii interne a fiecărui individ.

În acestă lucrare provocarea este reprezentată de recunoaşterea argumentelor individuale în texte medicale, care se ocupă în acelaşi timp, cu provocări diferite în a face acest lucru posibil.

1.1 Sistem de detectare a argumentelor

Argumentarea îşi are originile în fundaţionalism, o teorie de cunoştiţe în domeniul psihologiei. Ea a căutat să găsească motive pentru revendicările din logica şi legile concrete ale unui sistem universal de cunoaştere.

De-a lungul timpului eforturi mari au fost depuse în aria inteligenţei artificiale pentru a realiza şi a analiza "actul de a argumenta" cu ajutorul calculatorului. Argumentarea a fost folosită pentru a se obţine o dovadă teoretică, semantică pentru logica non-monotonă, începând cu influenţa muncii lui Dung (1995).

Sistemele de argumentare computaţională şi-au găsit aplicarea în special în domenii în care logica formală şi teoria deciziei clasice nu sunt în măsură să capteze bogăţia de raţionament, domenii cum ar fi dreptul şi medicina. In "Elemente de argumentaţie" , Philippe Besnard şi Anthony Hunter introduc tehnici pentru formalizarea argumentării deductive în inteligenţa artificială, subliniind formalizările în curs de dezvoltare, pentru argumentare practică.

În cadrul ştiinţei calculatoarelor, seria de workshop-uri ArgMAS (Argumentation in Multi-Agent Systems), seria de workshop-uri CMNA [5] şi mai nou conferinţele COMMA [6] sunt evenimente anuale care promovează domeniul argumentării. De asemenea, cartea intitulată "The journal Argument & Computation" (vezi [7]) este dedicată pentru explorarea intersecţiei dintre argumentare şi ştiinţa calculatoarelor.

Argumentarea este procesul în care argumentele sunt structurate şi evaluate pe baza interacţiunilor. Un argument constă dintr -un set de premise, oferite cu scopul susţinerii cererii. Argumentarea poate implica, de asemenea lanţuri de raţionament, în cazul în care revendicările sunt utilizate ca premise pentru revendici derivate ulterior. Exploatarea argumentativă[8, 9] este un domeniu nou de cercetare, care combină procesarea limbajului

natural (NPL) cu teoriile de argumentare şi răspunde la întrebări. "Argumentation mining" are drept scop detectarea în mod automat de argumente din documente de tip text, inclusiv alte funcţionalităţi, cum ar fi structura unui argument, relaţia dintre ele. Argumentarea este actul sau procesul de realizare a motivelor, de tragere a concluziilor şi de aplicare a lor în cazul în discuţie.

1.2 Prezentare generală a ontologiilor

Ontologiile sunt adesea utilizate în conjuncţie cu sistemele de extragere a argumentelor pentru a ajuta la extragerea şi detectarea argumentelor.

O definiţie concisă a ontologiei este dată de [10]: "O ontologie este o specificaţie explicită a unei conceptualizări". Termenul de ontologie este împrumutat din filozofie unde ontologia înseamnă "Existenţă". În domeniul calculatoarelor, o ontologie reprezintă o descrierea a conceptelor şi a relaţiilor care pot să existe între aceste concepte pentru un domeniu particular. Astfel, o ontologie poate să fie văzută ca un set de definiţii a conceptelor.

Pentru a creşte nivelul de împărtăşire a cunoştiinţelor un standard a fost dezvoltat pentru Web numit Ontology Web Language (OWL) [11]. Versiunea curentă a acestui standard este OWL 2. OWL 2 este un limbaj de reprezentare de cunoştiinţe cu înţeles formal, acesta reprezentând blocurile pe care Web Semantic se construieşte.

Aşa cum numele sugerează, OWL este utilizat pentru scrierea ontologiilor care încapsulează cunoştinţe despre un anumit domeniu de interes. O ontologie este o specificaţie a unui concept, de exemplu o descriere formală a acelui concept care poate să fie înţeleasă de maşini care folosesc aplicaţii speciale. Ontologiile sunt importante pentru sistemele care folosesc informaţia din diferite comunităţi; OWL reprezintă mediul prin intermediul cărui această informaţie poate să fie interschimbată.

1.3 Structura tezei

Această teză are 8 capitole unde fiecare capitol prezintă în detaliu câte o parte din procesul de cercetare şi dezvoltare realizat pentru acest proiect.

Capitolul 2 prezintă motivele pentru care a luat naştere acest proiect şi care sunt obiectivele specifice pe care trebuie să le îndeplinească.

Capitolul 3 deţine un sumar al articolelor studiate pentru acest proiect. Începe prin descrierea articolului pe care l-am citit despre Description Logic care este fundaţia pe care sunt construite ontologiile. După sunt prezentate anumite particularităţi a unui sistem de extragere a argumentelor. În ultimele două părţi ale acestui capitol am prezentat diferite implementări în ceea ce priveşte identificarea argumentelor în documentele text.

Capitolul 4 prezintă aspectele teoretice care au fost luate în considerare pentru realizarea sistemului de identificare a argumentelor. De asemenea, furnizează o descriere teoretică detaliată a algoritmului implementat.

Capitolul 5 prezintă arhitectura sistemului şi interacţiunea dintre componentele principale. Acest capitol de asemenea furnizează o detaliere a realizării componentelor utilizând GATE pipelines sau cum s-a realizat interacţiunea cu ontologii.

Capitolul 6 prezintă metoda de adnotare a corpusului, astfel încât sistemul să poată realiza evaluarea automată a performanţei. Acest capitol descrie şi tehnicile implementate pentru testarea aplicaţiei. Ultima parte a acestui capitol prezintă rezultatele obţinute pe corpusul adnotat.

Capitolul 7 prezintă un manual de instalare pentru sistem şi un ghid pentru îndrumarea realizării paşilor în realizarea celor mai importante taskuri.

Capitolul 8 prezintă rezultatele proiectului şi îmbunătăţirile care pot să fie aduse.

Capitolul 2

Obiectivele Proiectului

2.1 Specificarea problemei şi motivaţie

În domeniul medical, fluxul continuu de cercetări ştiinţifice conţine rezultate contradictorii suportate de argumente pro şi contra. Cum expertiza medicală se realizează la diferite niveluri, anumiţi agenţi umani au dificultăţi în a face faţă cu cantitatea mare de studii, dar de asemenea şi în a înţelege motivele, bucăţile de probă pretinse de susţinătorii şi oponenţii subiectului dezbătut.

Pentru a înţelege mai bine argumentele suport pentru noile descoperiri realizate în domeniul medical, avem nevoie de aplicaţii care să identifice argumente în articole ştiinţifice. Munca noastră are aici scopul de a umple acest gol în ceea ce priveşte domeniul tehnologic.

Destul de conştienţi de dificultatea acestei sarcini, ne îmbarcăm pe acest drum, bazându-ne pe intercalarea bine cunoscutului domeniu de cunoaştere a procesării limbajului natural. Pentru a formaliza domeniul medical existent ne bazăm pe ontologii. Pentru a structura modelul argumentării utilizăm expresivitatea şi capacităţile de raţionament a descrierii logice.

Pentru a realiza identificarea argumentelor formalizăm diferite pattern-uri lingvistice într-o regulă bazată pe limbaj. Am testat soluţia noastră pe un corpus ştiinţific realizat din aticole care au ca şi punct de focus cancerul la sân. Am realizat experimente care prezintă metrica F-measure între 0.84 şi 0.97 pentru identificarea concluziilor a unui argument şi între 0.84 şi 0.93 pentru identificarea premiselor unui argument.

Am ales domeniul cancerului la sân pentru a realiza identificarea argumentelor deoarece în ultima perioadă acest subiect a atins cote maxime în ceea ce priveşte volumul de informaţie digitală. Din păcate, această cantitate de informaţie a dus la depăşirea capacităţii umane de a menţine pasul cu ea. Rezultatul acestei probleme reprezintă un paradox, deoarece în zilele noastre avem multă informaţie disponibilă, dar nu avem aceeaşi creştere în ceea ce priveşte cunoştiinţele.

2.2 Obiectivele proiectului

Proiectul meu are două obiective primare. Primul obiectiv primar este de a dezvolta un sistem pentru identificarea argumentelor în cadrul documentelor text. Acesta prezintă posibilitatea de detectare a structurii argumentului, a concluziilor şi premiselor care le susţin. Sistemul trebuie să fie automatizat, astfel încât să suporte procesarea automată a documentelor text de diferite lungimi fără intervenţia utilizatorului uman.

Cel de-al doilea obiectiv primar este reprezentat de formarea unei ontologii în cadrul căreia argumentele detectate anterior vor fi salvate. Pentru îndeplinirea acestui obiectiv trebuie să se identifice ce concepte trebuie definite şi ce relaţii pot avea loc între acestea. Determinarea gamei conceptelor şi relaţiilor pe care ontologia trebuie să o acopere este realizată printr-un studiu asupra corpusului format din textele referitoare la domeniul cancerului la sân.

În concordanţa cu cele de mai sus, sistemul ar trebuie să procureze elemente vizuale pentru a arăta cum sunt adnotate documentele, de exemplu proiectul arată fragmentele de text identificate ca şi concluzii.

2.3 Specificaţiile proiectului

2.3.1 Cerinţele funcţionale

În această secţiune o să prezint cerinţele funcţionale pe care teza mea de licenţă ar trebui să le îndeplinească.

Încărcarea documentelor text

Utilizatorul va fi capabil să încarce documente text pentru a realiza corpusul de documente pe care sistemul este executat.

Încărcarea ontologiei

Utilizatorul va fi capabil să specifice ontologia care va fi utilizată de sistem pentru identificarea argumentelor.

Identificarea argumentelor

Sistemul identifică şi adnotează concluziile şi premise tuturor argumentelor care fac parte din articolele utilizate ca şi date de intrare.

Salvarea argumentelor

Sistemul va fi capabil să salveze în cadrul ontologiei de argumentare fiecare individ

(concluzie şi premisă) identificat în cadrul textului. Structura fiecărui argument va fi reconstruită prin intermediul relaţiilor existente în ontologie.

Vizualizarea adnotărilor

După adnotarea textelor, sistemul trebuie să asigure o interfaţă grafică intuitivă pentru vizualizarea adnotărilor generate.

Vizualizarea ontologiei

După formarea ontologiei de argumentare, aplicaţia va putea pune la dispoziţie o modalitate de vizualizare sub formă de graf a celor două ontologii (de argumentare şi de cancer).

Transformarea ontologiei .racer în .owl

Sistemul asigură modalitatea de transformare a unei ontologii cu extensia *.racer* într-una cu extensia *.owl*.

Vizualizarea listelor de cuvinte utilizate la indentificarea argumentelor

Aplicaţia prin intermediul interfeţei grafice va asigura modalitatea de vizualizare a listelor de cuvinte utilizate în indentificarea argumentelor, în cadrul textelor de intrare.

Vizualizarea identificatorilor şi a argumentelor

Aplicaţia prin intermediul interfeţei grafice va asigura modalitatea de vizualizare a identificatorilor şi a structurii argumentelor observate în cadrul textelor de intrare. De asemenea, prin precizarea unui număr se vor putea vizualiza concluziile şi premisele doar din documentul identificat în cadrul corpusului cu ajutorul numărului.

Realizarea listelor de cuvinte utilizate la identificarea argumentelor

Sistemul asigură realizarea listelor de cuvinte pe care identificarea argumentelor se bazează.

Identificarea ABox-ului şi TBox-ului

Sistemul pune la dispoziţie recunoaşterea ABox-ului şi a TBox-ului ontologiei încărcate în momentul respectiv.

2.3.2 Cerinţele nonfuncţionale

În această secţiune o să prezint cerinţele nonfuncţionale pe care sistemul trebuie de asemenea să le îndeplinească.

Extensibilitate

Se iau în considerare îmbunătăţirile viitoare, extinderea şi adăugarea de caracteristici sistemului.

Documentaţie

Un ghid de administrare şi un ghid de utilizare trebuie să fie dezvoltate în format .pdf.

Recall

Recall este o măsurătoare care arată câte argumente au fost identificate din total, având un impact major asupra aplicaţiei. O valoare cât mai mare asigură faptul că aplicaţia identifică o variatate cât mai mare de concluzii şi premise, dar în acelaşi timp poate să afecteze precizia sistemului prin introducerea rezultatelor fals pozitive. Sistemul trebuie să asigure o valoare mai mare de 0.6 a acestei măsurători.

Precizie

Precizia arată câte argumente sunt identificate corect dintre cele recunoscute de sistem, având de asemenea un impact major asupra aplicaţiei. O valoare cât mai mare asigură faptul că aplicaţia indentifică cât mai corect argumentele. Însă, o valoare mare a preciziei afectează şi recall-ul aplicaţiei, deoarece limitează varietatea de argumente identificate. Sistemul trebuie să asigure o valoare mai mare de 0.6 a acestei măsurători.

Constrângeri de operare

Motorul RacerPro este necesar pentru interogarea ontologiei de argumentare; librăria java jRacer fiind folosită pentru crearea unei client racer, necesar conexiunii.

Capitolul 3

Studiu Bibliografic

În acest capitol o să prezint articolele pe care le-am studiat şi cunoştiinţele pe care le-am acumulat pentru dezvoltarea tezei.

În secţiunea 3.1 o să înfăţişez o scurtă descriere a logicii de descriere (Description Logic - DL), care reprezintă fundaţia dezvoltării ontologiilor.

În secţiunea 3.2 o să rezum articolele citite care au ca subiect principal extragerea argumentelor din cadrul documentelor text.

În secţiunea 3.3 au să fie înfăţişate soluţiile existente care rezolvă problema prezentă în această teză.

În secţiunea 3.4 se prezintă aplicaţiile utilizate pentru procesarea textului.

3.1 Logici de descriere

Prin intermediul documentului [12], ni se oferă detalii referitoare la logica descriptivă utilizată pentru modelarea cunoştinţelor. În logica descriptivă (DL) conceptele sunt costruite folosind setul de constructori, formate din negaţie, conjuncţie, disjuncţie, restricţie de valori şi restricţii existenţiale (vezi Tabelul 3.1).

În acest caz C şi D reprezintă concepte, iar r este numele unui rol. Semantica este definită pe baza interpretării $I = (\Delta^I, \cdot^I)$, unde domeniul Δ^I al lui I conţine un set de indivizi nevid şi funcţia de interpretare $\cdot^I$ mapează fiecare nume de concept C la un set de instanţe $C^I \in \Delta^I$ şi fiecare rol r la o relaţie binară $r^I \in \Delta^I \times \Delta^I$. Ultima coloană din Tabelul 3.1 arată expresia funţiei $\cdot^I$ pentru conceptele care nu sunt atomice.

O ontologie constă din terminologii (sau *TBox*) şi asocieri (sau *ABox*). O terminologie *TBox* este un set finit de axiome de forma $C \equiv D$ (echivalenţă) sau $C \sqsubseteq D$ (incluziune). *ABox* este un set finit de asocieri de concepte (*a:C*) sau asocieri de roluri (*(a,b):r*), unde C reprezinta un concept, r un rol, iar a şi b două instanţe ale conceptului C. În cadrul aceluiaşi *ABox* se presupune utilizarea de nume unice.

Un concept C este satisfăcut, dacă există o interpretare I pentru care $C^I \neq \varnothing$. Un concept D subsumează conceptul C ($C \sqsubseteq D$), dacă $C^I \subseteq D^I$ oricare ar fi interpretarea

Tabelul 3.1: Semantica şi sintaxa logicii $\mathcal{ALC}$.

Constructor	*Sintaxă*	*Semantică*
negaţie	$\neg C$	$\Delta^I \setminus C^I$
conjuncţion	$C \sqcap D$	$C^I \cap D^I$
disjuncţion	$C \sqcup D$	$C^I \cup D^I$
restricţie existenţială	$\exists\, r.C$	$\{x \in \Delta^I \mid \exists\, y : (x, y) \in r^I \wedge y \in C^I\}$
restricţie de valoare	$\forall\, r.C$	$\{x \in \Delta^I \mid \forall\, y : (x, y) \in r^I \rightarrow y \in C^I\}$
atribuire de instanţe	$a : C$	$\{a\} \in C^I$
atribuire de rol	$(a, b) : r$	$(a, b) \in r^I$

I. Constrângeri aplicate pe concepte, cum ar fi disjuncţia, şi pe roluri, cum ar fi domeniul, intervalul de valori, roluri inverse şi proprietăţi tranzitive, pot fi specificate în logici descriptive mai complexe[1].

Pentru a utiliza ontologiile în cadrul sistemului, acestea au fost modelate utilizând aplicaţia *Racer*. În articolul [13] aplicaţia este descrisă, fiind considerată principala interfaţă pentru web-ul semantic. RacerPro (Renamed ABox and Concept Expression Reasoner Professional) este un server capabil să interacţioneze cu ontologiile încărcate. RacerPorter este interfaţa grafică care rulează ca şi un client pentru RacerPro.

În Figura 3.1 se ilustrează un fragment de cod care descrie cum se poate interacţiona cu server-ul Racer din cadrul unei aplicaţii Java utilizând API-ul bazat pe comunicarea TCP, scopul exemplului fiind de a ilustra uşurinţa de utilizare a acestuia.

Conexiunea la server-ul de Racer se realizează cu ajutorul unui obiect client de tipul RacerClient care prezintă IP-ul $IP_RACER = 127.0.0.1$ şi portul $PORT_RACER = 8088$.

3.2 Detectarea argumentelor

Pentru detectarea argumentelor în cadrul documentelor text, am studiat mai multe modalităţi de detecţie în cadrul documentelor text. După cum se va putea observa acest domeniu de identificare a fost studiat într-o varietate de contexte.

În [14] reguli pentru detectarea structurii unui argument au fost extrase utilizând limbajul SWRL cu ajutorul programului FluentEditor [15]. S-a propus, astfel o specificaţie în care sunt prezentate zece scheme de argumentare cauzale, de exemplu identificarea premiselor, concluziilor şi a numelui schemei de argumentare, în cadrul jurnalelor ştiinţifice având ca şi principal subiect genetica.

Specificaţiile şi câteva dintre exemplele din care au fost derivate au fost utilizate pentru a crea un proiect iniţial de orientări, pentru adnotarea unui corpus. Instrucţiunile au fost evaluate în cadrul unui studiu de început care arăta cum diverse scheme puteau să

[1]Noi oferim doar câteva terminologii de bază despre logica descriptivă în această lucrare. Pentru o explicaţie detaliată despre acest subiect se poate aprofunda [12]

```
public class ConnectionClass{
   public static void main(String[] args) {
      RacerClient racer = new RacerClient(IP_RACER, PORT_RACER);
        try {
      racer.openConnection();
      racer.fullReset();
        racer.racerReadFile("\"" + LOCATION_ARGUMENTS_ONTOLOGY + "\"");
      ...
       racer.closeConnection();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
   }
}
```

Figura 3.1: Exemplu de interacţiune a server-ului Racer, în cadrul unei aplicaţii Java

fie recunoscute de către alţi cercetători pe baza citirii orientărilor.

Feng and Hirst [16] au investigat schema de identificare a argumentelor utilizând corpusul Aucaria, care conţine argumente adnotate din ziare, articole, înregistrări parlamentare, reviste şi discuţii on-line. Luând concluziile şi premisele astfel date, Feng şi Hirst au abordat problema recunoaşterii numelui schemei de argumentare pentru cele cinci mai frecvente scheme ale lui Walton [17] în corpus: *Argument din exemplu*, *Argument din cauză la efect*, *Raţionament practic*, *Argument din consecinţa* şi *Argument din clasificare verbală*.

În articolul [18] subiectul principal atins este realizat pe un corpus de date textuale format din opinii a consumatorilor despre produse (review). În timp ce opiniile pot să fie sortate după rating există o limită în ceea ce priveşte subiectele particulare. Se poate observa, faptul că, sistemul prezentat este unul semi-automat utilizând aplicaţii pentru extragerea informaţiilor care ajută la identificarea argumentelor în cadrul corpusului. Adnotările sunt realizate manual pe un set mare de documente. Nu există detectare automată a adnotărilor, indicatorii discursului, de asemenea terminologia legată de utilizator domeniu şi sentimente fiind identificată manual.

În articolul [19] subiectul principal la fel ca şi în cazul de faţă este legat de domeniul medical, şi anume în acesta este descrisă metoda prin intermediul căreia se poate detecta indicele de masă corporală.

3.3 Soluţii existente

3.3.1 ArguAna

ArguAna este o aplicaţie de revizuire a corpusului pentru analiza argumentelor. Aceasta realizează compilarea unui corpus echilibrat format din documente text, culese din comentariile utilizatorilor. Se vor crea scheme de adnotare pentru revizuirea argumentelor sau procesări de adnotări manuale realizate de experţi şi utilizatori web. Exemplul pe care este testat este format din documente care prezintă comentarii legate de hoteluri.

Adnotările manuale din fiecare text sunt alcătuite din sentimente şi proprietăţi ale produselor. Rezultatul procesării identifică prin analiza textului un grafic în care sunt expuse sentimentele locale din cadrul textului. De asemenea, un graf în care sunt identificate fiecare parte de vorbire din cadrul propoziţiilor este prezent. Fiecare propoziţie este un argument care cuprinde un set de fapte şi opinii justificând astfel un anumit sentiment.

3.3.2 Argument Workbench

Argument Workbench este o aplicaţie iterativă, integrată şi modulară pentru extragerea, reconstruirea şi vizualizarea argumentelor. De asemenea în [20] se discută despre arhitectura şi dezvoltarea aplicaţiei Argument Workbench. Aceasta susţine reconstruirea argumentelor din surse textuale folosind informaţii pre-procesate, fiind o primă etapă din secvenţa de analiză ulterioară şi apoi construirea unui grafic pe baza argumentului.

Instrumentul recoltează şi preprocesează comentarii, evidenţiază identificatorii argumentelor, actele de vorbire şi terminologia epistemică, identificând în acelaşi timp terminologia domeniului. Inginerul de argument analizează ieşirea, apoi intrarea fiind extrasă în instrumentul de vizualizare DebateGraph.

3.4 Aplicaţii pentru procesarea limbajului natural

Conform [21], *GATE* este un mediu care pune la dispoziţie un set bogat de instrumente grafice interactive care pot fi utilizate pentru crearea, măsurarea şi întreţinerea de componente software destinate pentru procesarea limbajului natural.

În GATE logica este aranjată în module numite ”pipelines”. Ieşirea unui pipeline poate să reprezinte intrarea pentru alt pipeline, de asemenea mai multe pipeline-uri pot să fie grupate pentru a se crea unu sistem mai complex.

Framework-ul GATE conţine un pipeline de extragere a informaţiilor numit *ANNIE*, fiind un bun punct de început pentru realizarea altor sisteme personalizate. Acest pipeline este alcătuit din: *Tokenizer*, *POS Tagger*, *Gazetteer*, *Sentence Splitter* şi *JAPE Transducer*.

Tokenizer împarte textul în tokeni simpli, cum ar fi numere, semne de punctuaţie şi cuvinte de diferite tipuri. *ANNIE Sentence Splitter* segmentează textul în propoziţii,

Figura 3.2: Exemplu de analiză a argumentelor în fragmente ce reprezintă opinia consumatorilor, cu unealta ArguAna

în timp ce *Tagger-ul* produce etichete corespunzătoare fiecărei părţi de vorbire, a fiecărui cuvânt sau simbol.

Lista "lookup" conţinută de *Gazetteer* este utilizată în faza iniţială a regulilor de detectare a adnotărilor şi în procesarea resurselor. Framework-ul are abilitatea de a cascada reguli JAPE înlănţuite.

GATE ajută la crearea unori structuri complexe cum ar fi, software de prelucrare a limbajului natural utilizând structuri şi algoritmi cum ar fi, grafurile de adnotare, vizualizarea rezultatelor procesării şi suport pentru măsurarea preciziei, în comparaţie, cu rezultatele produse manual, prin adnotare manuală. Principala funcţionalitate oferită de GATE este adnotarea documentelor, toate celelalte funcţionalităţi fiind strâns legate de aceasta.

Capitolul 4

Analiză şi Fundamentare Teoretică

Înainte de a realiza implementarea propriu zisă este nevoie de analiza domeniului şi a posibilităţilor legate de instrumentele pe care le pot utiliza. În acest capitol o să prezint analiza domeniului medical împreună cu cel al identificării argumentelor şi problemele apărute. De asemenea, o să prezint instrumentele alese şi motivul pentru care le-am selectat pe acestea în defavoarea altora.

4.1 Modelul argumentativ

Filozofia şi logica propun o vastă cantitate de informaţie asupra teoriei de argumentare (e.g. Toulmin (1958),Walton et al. (2008), Freeman (2011))[1]. Majoritatea acestor teorii în general sunt de acord cu faptul că un argument este format din mai multe componente de argumentare care poate fi fie premise sau concluzii.

Cea mai simplă formă a unui argument $a\langle p_i, c\rangle$ include exact o concluzie c şi cel puţin o premisă p_i. Definiţia în DL este următoarea:

$$Argument \equiv \exists\ hasPremise.Premise \sqcap (= 1)hasClaim.Claim \tag{4.1}$$

Concluzia este componenta centrală a unui argument care poate fi fie falsă, fie adevărată. Astfel, afirmaţia este o declaraţie care trebuie să fie acceptată de cititor fără motive suplimentare, premisele aflându-se la baza plauzibilităţii acesteia.

Presupunem că concluziile şi premisele prezintă descriptori textuali, fiind semnalizaţi de indicatori lexicali specifici.

$$Claim \equiv \exists\, hasText.String \sqcap \exists\, hasIndicator.ClaimIndicator \tag{4.2}$$

$$Premise \equiv \exists\, hasText.String \sqcap \exists\, hasIndicator.PremiseIndicator \tag{4.3}$$

[1] O revizuire a teoriei argumentării este dincolo de domeniul de aplicare a acestei teze. Un studiu poate fi găsit în Bentahar şi colab. (2010)

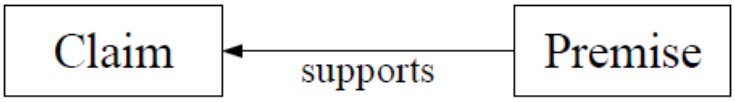

Figura 4.1: Ilustrarea unui argument simplu

Ne bazăm pe indicatori textuali clasificaţi în mai multe concepte:

$$Indicator \equiv PremiseIndicator \sqcup ClaimIndicator \sqcup MacroIndicator \tag{4.4}$$

Moştenirea între roluri a fost adoptată de subrolurile *hasPremiseIndicator* şi *hasClaimIndicator*:

$$\top \sqsubseteq \forall\, hasIndicator.Indicator \tag{4.5}$$
$$hasPremiseIndicator \sqsubseteq hasIndicator \tag{4.6}$$
$$hasClaimIndicator \sqsubseteq hasIndicator \tag{4.7}$$

Exemplul 1 (Exemplu de indicatori de concluzie) *Următorii indicatori lexicali semnalează, de obicei, o concluzie, acestea fiind instanţe ale conceptului ClaimIndicator (Ci): "consequently", "therefore", "thus", "so", "hence", "accordingly", "we can conclude that", "it follows that", "we may infer that", "this means that", "it leads us to believe that", "this bears out the point that", "which proves/implies that", "as a result", "often", "the purpose of", "results of such", "the fact that".*

Exemplul 2 (Exemplu de indicatori de premisă) *Următorii indicatori lexicali semnalează, de obicei, o premisă, acestea fiind instanţe ale conceptului PremiseIndicator (Pi): "since", "because", "for", "whereas", "in as much as", "for the reasons that", "in view of the fact", "as evidenced by", "given that", "seeing that", "as shown by", "assuming that", "in particular".*

MacroIndicator este format din mai multe cuvinte unul dintre ele indicând verbe legate de premisă sau concluzie. Ne bazăm pe următoarea moştenire:

$$VerbRelatedToClaim \sqcup VerbRelatedToPremise \sqsubseteq MacroIndicator \tag{4.8}$$

Exemplul 3 (Verbe legate de conluzie) *Următoarele verbe de obicei semnalează o concluzie, fiind o instanţă a conceptului ConclusionVerbs (Vc): "to report", "to believe", "to assesse", "to identify", "to highlight", to be essential", "to confirm", "to estimate", "to provide", "to express", "to experience", "to recall", "to accept", "to reflect", "to categorize", "to indicate", "to exemplify", "to define", "to show", "to qualify"*

Exemplul 4 (Verbe legate de premisă) *Următoarele verbe de obicei semnalează o premisă, fiind o instanţă a conceptului PremiseVerbs (Vp): "to note", "to subdivide", "to contain", "to result", "to observe", "to accord", "to regard", "to feel", "to show", "to receive", "to examine", "to report", "to transcribe", "to encompass".*

Se notează faptul că indicatorii de premisă pot să apară după ce concluzia a luat sfârşit, acest lucru fiind ilustrat în exemplul 5.

Exemplul 5 (Indicatori de premisă înainte de premisă) *Se consideră propoziţia:*

> *"[Spirituality was highlighted as a fundamental component of the healing process]$_{Claim}$. [[In particular]$_{PremiseIndicator}$, survivors noted that their faith in Godş direction over the doctors healed them.]$_{Premise}$*

Textul este adnotat cu o concluzie şi o premisă care prezintă indicator (PI), instanţiat cu textul "In particular". Argumentul are concluzia pe prima poziţie, premisa urmând-o. Indicatorul de premisă precedă premisa. ABox-ul corespunzător este următorul:

$$(a, c) : hasClaim \tag{4.9}$$
$$(a, p) : hasPremise \tag{4.10}$$
$$(p, "In\ particular") : hasIndicator \tag{4.11}$$
$$(c, p) : before \tag{4.12}$$
$$("In\ particular", p) : before \tag{4.13}$$

Bazându-se pe informaţia anterior indetificată, sistemul clasifică argumentul a ca fiind *ClaimPremiseArgument*.

Luându-se în considerare argumentul medical din exemplul 6:

Exemplul 6 (Exemplu argument) *[Key informants highlighted]$_{ClaimIndicator}$ spirituality as a very important component of many women's cancer experience. These communities, particularly African American, Asian and Latina, hold firm religious and spiritual beliefs and practices. [[In particular]$_{PremiseIndicator}$, many have an unshakable belief in the power of prayer, putting more importance on spirituality, their religious beliefs than on health care providers.]$_{Premise}$*

Structura acestuia este alcătuită din:

$$a : Argument \tag{4.14}$$
$$(a, c) : hasClaim \tag{4.15}$$
$$(a, p) : hasPremise \tag{4.16}$$
$$(c, \text{"Key informants highlighted"}) : hasIndicator \tag{4.17}$$
$$(p, \text{"In particular"}) : hasIndicator \tag{4.18}$$

Există argumente în care premisele precedă concluzia, dar de asemenea şi argumente în care premisa apare după concluzie.

$$PCArgument \equiv \exists\, hasPremise.(\exists\, before.Claim) \tag{4.19}$$
$$CPArgument \equiv \exists\, hasPremise.(\exists\, after.Claim) \tag{4.20}$$

Rolurile *before* and *after* sunt roluri inverse şi tranzitive, cu atât domeniu şi gamă reprezentată de propoziţiile:

$$before^{-} \equiv after \tag{4.21}$$
$$before^{+} \tag{4.22}$$
$$after^{+} \tag{4.23}$$
$$\top \sqsubseteq \forall\, before.Sentence \tag{4.24}$$
$$\exists\, before.\top \sqsubseteq Sentence \tag{4.25}$$

Instanţe de *PCArgument* and *CPArgument* sunt ilustrate în exemplele 7 şi 8. O să dăm fragmente de text unde structura unui argument consistă dintr-o concluzie şi mai multe premise, dar aranjate în ordine diferită.

Exemplul 7 (PCArgument) *Deoarece concluzia apare după premisă, argumentul identificat este clasificat ca şi PCArgument:*

> *"[For women with non proliferative findings, no family history, a weak family history of breast cancer]$_{Premise}$, [doctors reported no increased risk."]$_{Claim}$*

Exemplul 8 (CPArgument) *Consider textul:*

> *"[[Patients report]$_{MacroIndicator}$ on the risk of breast cancer]$_{Claim}$ [according to histologic findings, the age at diagnosis of benign breast disease, the strength of the family history."]$_{Premise}$*

Propoziţia prezintă prima dată partea Claim introdusă de macro identificatorul "Patients report", urmată de partea de Premise. Astfel, acest argument este clasificat ca CPArgument.

4.2 Modelarea ontologiilor

4.2.1 Ontologie de argumentare

Tbox-ul următor introduce subconceptul de *ClinicalArgument*, care este un tip particular de argument. De asemenea, *MedicalArgument* este un concept particular de *ClinicalArgument* (liniile 4.26 şi 4.27).

Un argument poate sprijini alt argument prin intermediul rolului *supports*, care are ca domeniu conceptul *Argument* (linia 4.28) şi aceeaşi gamă *Argument* (linia 4.29). Linia 4.30 specifică faptul că rolul *supports* este tranzitiv. Relaţia de atac dintre două argumente este modelată de rolul *attacks* care are domeniul şi gama un set de argumente (liniile 4.31 şi 4.32).

$$ClinicalArgument \sqsubseteq Argument \tag{4.26}$$
$$MedicalArgument \sqsubseteq ClinicalArgument \tag{4.27}$$
$$\exists\, supports.\top \sqsubseteq Argument \tag{4.28}$$
$$\top \sqsubseteq \forall\, supports.Argument \tag{4.29}$$
$$supports^{+} \tag{4.30}$$
$$\exists\, attacks.\top \sqsubseteq Argument \tag{4.31}$$
$$\top \sqsubseteq \forall\, attacks.Argument \tag{4.32}$$

Următoarele linii reprezintă modelarea prin intermediul limbajului folosit de aplicaţia Racer a conceptelor descrise mai sus :

```
(implies MedicalArgument argument)
(implies ClinicalArgument MedicalArgument)

(define-primitive-role attacks :domain Argument :range Argument)
(define-primitive-role supports :domain Argument
                                :range Argument :transitive t)
```

Tbox-ul următor pune în evidenţă relaţia *hasPremise* cât, şi rolul *hasClaim* care au ca şi gamă conceptul *Premise*, respectiv *Claim*.

$$\exists\, hasClaim.\top \sqsubseteq Claim \tag{4.33}$$
$$\exists\, hasPremise.\top \sqsubseteq Premise \tag{4.34}$$

Următoarele linii reprezintă modelarea prin intermediul limbajului folosit de aplicaţia Racer a conceptelor descrise mai sus :

```
(define-primitive-role hasClaim  :range Claim)
(define-primitive-role hasPremise  :range Premise)
```

Structura unui argument este formată din cel puţin o premisă şi o singura concluzie.

$$argument \equiv \exists\, hasPremise.Premise \sqcap \exists!hasClaim.Claim \tag{4.35}$$

```
(equivalent argument (and (at-least 1 hasPremise) (exactly 1 hasClaim)))
```

De asemenea, pot exista şi argumente cu cel puţin trei premise, intitulate *3PremisesArgument*.

$$3PremisesArgument \equiv Argument \sqcap \exists\, hasPremise.Premise \tag{4.36}$$

```
(equivalent 3PremisesArgument (and Argument (at-least 3 hasPremise Premise)))
```

Pentru identificarea mai uşoară a argumentelor în cadrul textelor se folosesc indicatori. Aceştia sunt clasificaţi în ontologie în două ramuri, *ClaimIndicator* şi *PremiseIndicator*, ambele fiind tipuri particulare de *ArgumentIndicator* (liniile 4.37 şi 4.38).

$$ClaimIndicator \sqsubseteq ArgumentIndicator \tag{4.37}$$
$$PremiseIndicator \sqsubseteq ArgumentIndicator \tag{4.38}$$

```
(implies ClaimIndicator ArgumentIndicator)
(implies PremiseIndicator ArgumentIndicator)
```

MacroIndicator este format din mai multe cuvinte, unul dintre ele indicând verbe legate de premisă sau concluzie. În cadrul descrierii logice, acest lucru este ilustrat prin instanţierea conceptului de *VerbRelatedToClaim* şi *VerbRelatedToPremise* ca tipuri particulare de *MacroIndicator* (liniile 4.39 şi 5.1).

$$VerbRelatedToClaim \sqsubseteq MacroIndicator \tag{4.39}$$
$$VerbRelatedToPremise \sqsubseteq MacroIndicator \tag{4.40}$$

```
(implies VerbRelatedToClaim MacroIndicator)
(implies VerbRelatedToPremise MacroIndicator)
```

$$a : MedicalArgument \tag{4.43}$$
$$(a, c) : hasClaim \tag{4.44}$$
$$(a, p) : hasPremise \tag{4.45}$$
$$(c, ci) : hasIndicator \tag{4.46}$$
$$(p, pi) : hasIndicator \tag{4.47}$$
$$(ci, vc) : containsVerb \tag{4.48}$$
$$(pi, vp) : containsVerb \tag{4.49}$$
$$(vc, "highlighted") : hasText \tag{4.50}$$
$$(vp, "resulting") : hasText \tag{4.51}$$
$$(ci, \text{"Key informants highlighted"}) : hasText \tag{4.52}$$
$$(pi, \text{"In particular resulting"}) : hasText \tag{4.53}$$

Figura 4.2: Exemplu de ABox cu argument clinic

Pentru fiecare argument şi concept elementele sunt instanţiate cu o descriere contextuală ataşată fiecăruia prin adoptarea atributului *hastText*. Atributul prezintă unul dintre domeniile *Claim*, *Premise*, *ArgumentIndicator*, *MacroIndicator* şi gama *String* (liniile 5.2 şi 5.3).

$$\top \sqsubseteq \forall\, hastText.String \tag{4.41}$$
$$\exists\, hasText.(Claim \sqcup Premise \sqcup ArgumentIndicator \sqcup MacroIndicator) \sqsubseteq \top \tag{4.42}$$

```
(define-concrete-domain-attribute hasText :domain (or Claim Premise
                                                   ArgumentIndicator MacroIndicator)
                                                   :type string)
```

În figura 4.2 este pun în evidenţă ABox-ul care îl conţine pe individul *a* de tip *MedicalArgument*, având concluzia *c* (linia 5.4) şi premisa *p* (linia 4.45). De asemenea, prezintă indicatorii *ci* şi *pi* (liniile 5.5 şi 4.47) instanţiaţi cu textele *"informants highlighted"* şi *"In particular resulting"* (liniile 4.52 şi 5.9). După cum se poate observa, indicatorii prezintă verbe care au legătură cu concluzia şi premisa (liniile 5.6 şi 5.7). Astfel indivizii *vc* şi *vp* sunt instanţiaţi cu textele *"highlighted"*, respectiv *"resulting"*.

4.2.2 Ontologie de cancer la sân

În lucrarea prezentă de teză focusul este realizat pe ontologii medicale, şi în special pe cele legate de cancer la sân.

$$Cancer \sqsubseteq Disease \tag{4.54}$$
$$BreastCancer \sqsubseteq Cancer \tag{4.55}$$
$$\top \sqsubseteq \forall\, manifestedSymptom.Symptom \tag{4.56}$$
$$\top \sqsubseteq \forall\, appliedTreatment.Treatment \tag{4.57}$$
$$\top \sqsubseteq \forall\, affectedDomain.Domain \tag{4.58}$$
$$\top \sqsubseteq \forall\, impliedPerson.Person \tag{4.59}$$
$$\top \sqsubseteq \forall\, haveCharacteristic.Characteristic \tag{4.60}$$
$$\exists\, haveQuantifier.\top \sqsubseteq People \tag{4.61}$$
$$\top \sqsubseteq \forall\, haveQuantifier.Quantifier \tag{4.62}$$

Figura 4.3: Exemplu de TBox în domeniul cancerului

Ontologia intitulată *Breast Cancer Grading Ontology* (BCGO) asignează o notă fiecărei tumori, începând de la cele trei criterii ale NGS, fiind parte din categoria procesului biologic.

Tbox-ul in Fig. 4.3 introduce concepte, ca şi *Cancer* care este un tip particular de *Disease*, iar *BreastCancer* fiind un tip particular de *Cancer* (liniile 4.54, 4.55).

O boală are simptome, prezentate de rolul *manifestedSymptom* care are ca şi gamă conceptul *Symptom* (linia 5.10). Unul sau mai multe tratamente pot să fie recomandate cu ajutorul rolului *appliedTreatment* care are ca şi gamă conceptul *Treatment* (linia 4.57). Cancerul la sân afectează puternic toate domeniile vieţii umane. Acest lucru este modelat de rolul *affectedDomain* având ca şi gamă conceptul *Domain* (linia 5.11).

Persoanele implicate, de exemplu doctori, pacienţi, sunt reprezentaţi prin rolul *impliedPerson*, care are ca şi gamă conceptul *Person* (linia 4.59). Cancerul la sân este alcătuit din caracteristici, acest lucru fiind modelat de relaţia *haveCharacteristic* cu gama *Characteristic* (linia 5.12). În unele cazuri persoanele implicate sunt numărate, pentru aceasta fiind folosită relaţia *haveQuantifier* cu domeniul *People* şi gama *Quantifier* (liniile 5.13, 4.62).

Următoarele linii reprezintă modelarea prin intermediul limbajului folosit de aplicaţia Racer a conceptelor descrise mai sus :

```
(implies Cancer Disease)
(implies BreastCancer Cancer)
(define-primitive-role manifestSymptom  :range Symptom)
(define-primitive-role appliedTreatment  :range Treatment)
(define-primitive-role affectDomains :range Domain)
(define-primitive-role impliePeople :range People)
(define-primitive-role implieFactors  :range Factors)
(define-primitive-role haveCharacteristic :range Characteristic)
```

$$a : BreastCancer \tag{4.63}$$
$$(a,\ \textit{"Skin irritation or dimpling"}) : manifestedSymptom \tag{4.64}$$
$$(a, \text{"}Chemotherapy\text{"}) : appliedTreatment \tag{4.65}$$
$$(a,\ \textit{"Family history"}) : affectedDomain \tag{4.66}$$
$$(a, \text{"}Doctors\text{"}) : impliedPerson \tag{4.67}$$
$$(a, \text{"}Woman\text{"}) : impliedPerson \tag{4.68}$$
$$(a, \textit{"Hormone receptivity"}) : haveCharacteristic \tag{4.69}$$
$$(a, \textit{"High levels of HER2"}) : haveCharacteristic \tag{4.70}$$

Figura 4.4: Abox cu o instanţa de cancer la sân. Aici a este o scurtătură pentru "Angiosarcoma".

```
(define-primitive-role haveQuantifier :range Quantifier :domain People)
```

ABox-ul din figura 4.4 conţine individul a de tipul *BreastCancer* instanţiat cu *"Angiosarcoma"*. Acest individ manifestă simptomele *"Skin irritation or dimpling"*, având persoanele implicate *"Doctors"* şi *"Woman"*. Tratamentul aplicat în acest caz este reprezentat prin instanţa individului *"Chemotherapy"*. De asemenea, această boală afectează domeniul *"Family history"* şi prezintă caracteristicile *"Hormone receptivity"* şi *"High levels of HER2"*.

Sistemul foloseşte ontologii despre cancer pentru a crea liste specifice de cuvinte. Terminologia este intrare pentru diferite fişiere text, cum ar fi *cancerRelatedWords.lst* (indicând termeni referitori la domeniul cancerului) sau *peopleInvolved.lst* (indicând termeni referitori la oamenii implicaţi sau afectaţi de această boală). Aceste liste sunt folosite de *Gazetteer* asociind termenii cu un *majorType* cum ar fi *"CancerRelatedWords"* şi *"PeopleInvolved"*. Regulile JAPE convertesc aceste adnotări pentru vizualizare şi interogare.

De exemplu, dacă se presupune că textul prezintă cuvântul *"breast cancer"* şi GATE are o listă în *Gazetteer* care conţine *"breast cancer"* în ea, atunci programul GATE (vezi [21]) găseşte cuvântul în listă pe care îl adnotează cu *majorType* ca fiind *"CancerRelatedWords"*, această adnotare va fi intitulată *CancerRelatedWords*. Gama de termeni care indică persoanele indicate sunt adnotate cu *"PeopleInvolved"*.

O metodă de a interoga ontologia de cancer se poate realiza cu comanda *(CONCEPT-INSTANCES individualName)*, unde *individualName* este înlocuit cu *TREATMENT*, *SYMPTOM*, *CHARACTERISTIC*, *PEOPLE*, *DOMAIN*. Altă interogare care poate să fie folosită este *(INDIVIDUAL-TOLD-ATTRIBUTE-VALUE individualName attributeName)*. Aceasta ajută utilizatorul pentru a obţine atributul unui *individualName*, unde *attributeName* este înlocuit cu *HASTEXT*.

Aplicaţia poate realiza adnotaţii pentru concepte mai complexe cu ajutorul adnotărilor de nivel jos. În acest fel, *Gazetteer* furnizează un concept de acoperire pentru termeni înrudiţi, care poate fi interogat sau utilizat prin procedeele de adnotare ulterioare.

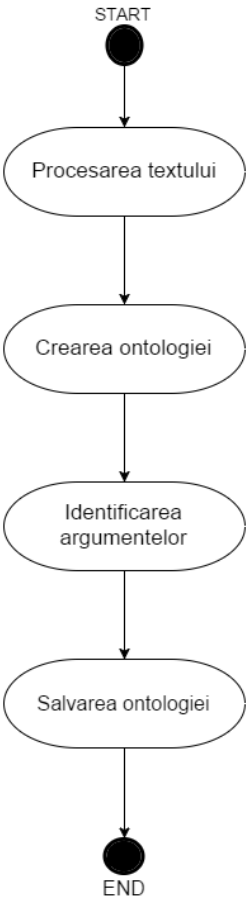

Figura 4.5: Ilustrarea fazelor generale ale sistemului

Avantajul utilizării ontologiilor pentru realizarea listelor specificate mai sus este acela de a a construi sisteme mult mai puternice şi interoperabile în domeniul medical. Ontologiile suportă nevoia proceselor din domeniul medical de: transmitere, reutilizare şi împărtăşire a datelor despre pacienţi.

Probabil cel mai semnificativ beneficiu dat de ontologii sistemelor din domeniul medical este, abilitatea de a suporta integrarea datelor şi a cunoştinţelor. Însă, partea negativă este reprezentată de impactul sceptic cu privire la menţinerea design-ului unei aplicaţii realizate cu ajutorul ontologiilor.

4.3 Funcţionalitate

4.3.1 Prezentarea generală a fazelor sistemului

În cadrul figurii 4.5 sunt identificate fazele importante pe care teza se bazează. În secţiunile ce au să urmeze acestea vor fi detaliate împreună cu acţiunile necesare realizării

acestora.

4.3.2 Funcţiile sistemului

Aplicaţia trebuie să îndeplinească următoarele cerinţe fundamentale:

- ataşarea documentelor la corpusul existent
- conectarea prin intermediul aplicaţie *Racer* la ontologia de cancer şi la cea de argumente
- crearea listelor de cuvinte (*.lst*) aferente ontologiei de cancer şi a celei de argumentare
- procesarea documentelor pentru a le împărţii pe tokeni şi propoziţii
- adnotarea documentelor pe baza algoritmului şi a listelor (*.lst*)
- extragerea informaţilor identificate de adnotări
- salvarea în ontologia de argumentare a informaţilor extrase
- vizualizarea adnotărilor
- interogarea ontologiei create
- convertirea ontologiei cu extensia *.racer* într-una cu extensia *.owl*
- vizualizarea ontologiei
- vizualizarea listelor de cuvinte (*.lst*)

4.4 Cazuri de utilizare

În acest capitol sunt prezentate cazurile de utilizare importante pe care utilizatorul uman le poate efectua în cadrul sistemului. După cum se poate observa, nu toate funcţionalităţiile sunt cazuri de utilizare, unele dintre ele fiind realizate implicit, fără ca utilizatorul să le solicite în mod special. În următoarele subsecţiuni au să fie ilustrate cazurile de utilizare, care stau la baza tezei de licenţă.

4.4.1 Identificarea argumentelor

În figura 4.6 se poate observa diagrama de flux a cazului de identificare a structurii argumentelor în cadrul documentelor text.

Titlu: Identificarea argumentelor

Rezumat: Acest caz de utilizare permite utilizatorului să identifice structura argumentelor din cadrul documentelor aflate în corpus

Actori: Utilizatorul (primar)

Precondiţii: Trebuie să existe documente în corpus şi ontologiile necesare

Postcondiţii: Crearea de adnotări asupra documentelor procesate, informaţiile detectate fiind salvate într-o ontologie cu posibilitate de vizualizare

Scenariu de succes:

1. Utilizatorul porneşte aplicaţia
2. Selectează butonul *Start RacerPorter* din meniul *Racer*
3. Apasă pe butonul numit *Load ontology* din cadrul meniului *Cancer ontology* şi selectează ontologia de cancer dorită
4. Apasă pe butonul *Create lists related to cancer* din meniul *Cancer ontology* pentru crearea listelor necesare procesării textelor
5. Apasă butonul *Close connection ontology* din meniul *Cancer ontology* pentru închiderea conexiunii cu această ontologie
6. Apasă pe butonul *Load ontology* din meniul *Arguments ontology* şi selectează ontologia de argumentare dorită
7. Apasă pe butonul *Create lists related to arguments* din meniul *Arguments ontology* pentru crearea listelor necesare procesării textelor
8. Se apasă pe butonul *Set corpus* din meniul *Arguments ontology* pentru încăcarea documentelor text în corpus
9. Pentru identificarea concluziilor şi a premiselor se va apăsa *Identify claim*, respectiv *Identify premise* din cadrul aceluiaşi meniu intitulat, *Arguments ontology*
10. Pentru salvare se alege *Save ontology* din meniul *Arguments ontology*

Secvenţe alternative:

2. Aplicaţia Racer nu porneşte din anumite cauze
   - 2.1. Sistemul informează utilizatorul faptul că aplicaţia RacerPorter nu este găsită
   - 2.2. Utilizatorul porneşte manual aplicaţia

3. Utilizatorul nu selectează ontologia de cancer dorită

   3.1. Utilizatorul apasă pe butonul *Close connection ontology* din meniul *Cancer ontology* pentru închiderea conexiunii cu această ontologie

   3.2. Utilizatorul apasă pe butonul numit *Load ontology* din cadrul meniului *Cancer ontology* şi selectează ontologia de cancer dorită

4. Aplicaţia nu poate să realizeze listele de cuvinte necesare procesării textului

   4.1. Utilizatorul este informat asupra erorii şi a cauzei acesteia

   4.2. Şcenariul se întoarce la pasul 3

5. Nu există conexiune care să fie închisă

   5.1. Utilizatorul este informat asupra erorii şi a cauzei acesteia

   5.2. Şcenariul se întoarce la pasul 3

6. Utilizatorul nu selectează ontologia de argumentare dorită

   6.1. Utilizatorul apasă pe butonul *Close connection ontology* din meniul *Arguments ontology* pentru închiderea conexiunii cu această ontologie

   6.2. Utilizatorul apasă pe butonul numit *Load ontology* din cadrul meniului *Arguments ontology* şi selectează ontologia de argumentare dorită

7. Aplicaţia nu poate să realizeze listele de cuvinte necesare procesării textului

   7.1. Utilizatorul este informat asupra erorii şi a cauzei acesteia

   7.2. Şcenariul se întoarce la pasul 6

8. Aplicaţia nu poate să seteze corpusul

   8.1. Utilizatorul este informat asupra erorii şi a cauzei acesteia

   8.2. Şcenariul se întoarce la pasul 7

9. Aplicaţia nu poate să identifice structura argumentului în cadrul textelor

   9.1. Utilizatorul este informat asupra erorii şi a cauzei acesteia

   9.2. Şcenariul se întoarce la pasul 8

10. Aplicaţia nu poate să salveze argumentele identificate în cadrul textelor

    10.1. Utilizatorul este informat asupra erorii şi a cauzei acesteia

    10.2. Se va urma şcenariul începând cu pasul 6

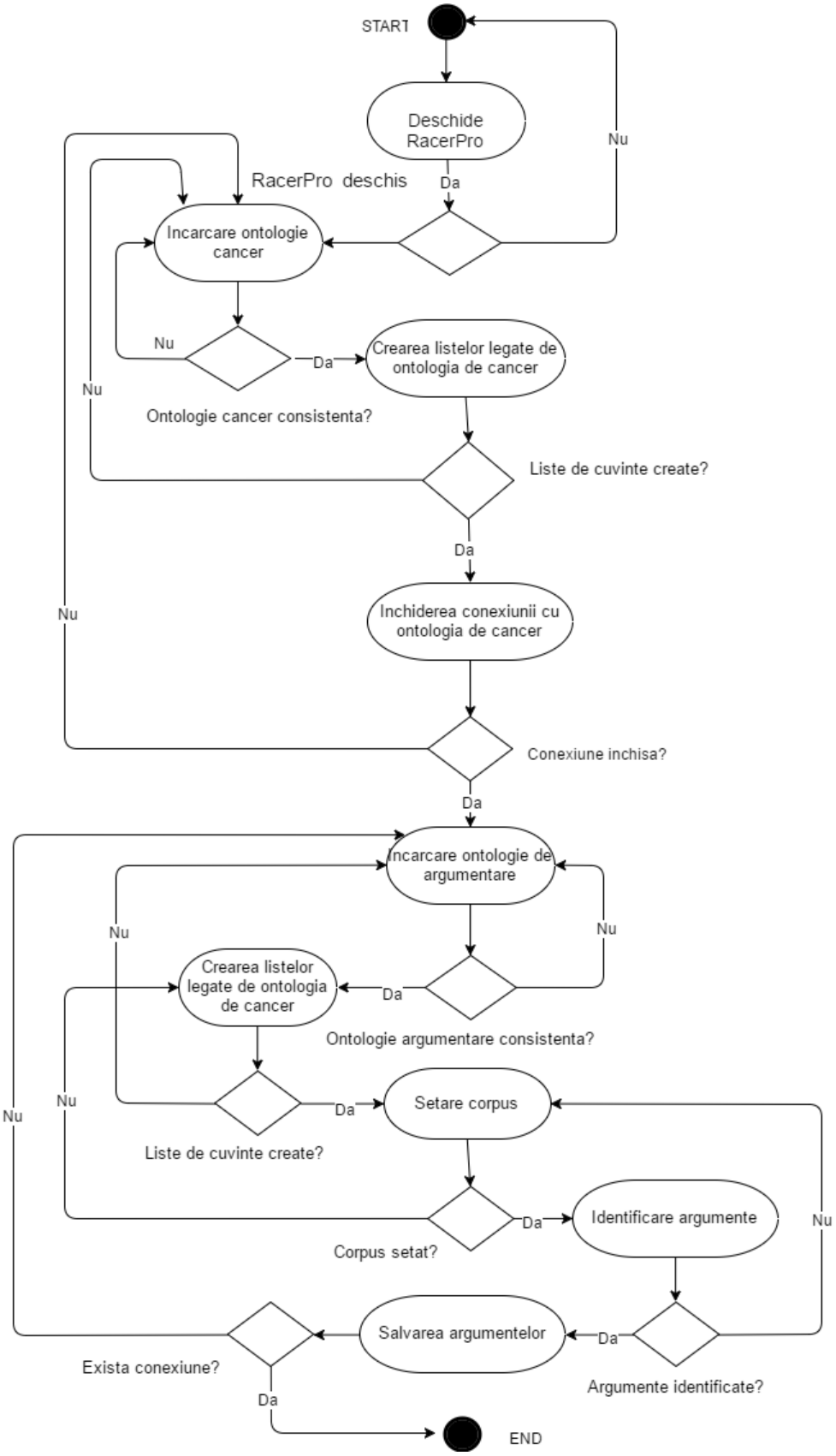

Figura 4.6: Diagrama de flux pentru cazul de salvare a argumentelor

4.4.2 Interogarea ontologiei

Titlu: Interogarea ontologiei

Rezumat: Acest caz de utilizare permite utilizatorului să interogheze ontologia de argumentare creată în urma procesării

Actori: Utilizatorul (primar)

Precondiţii: Trebuie să existe documente în corpus şi ontologiile necesare

Postcondiţii: Afişarea corectă a rezultatului interogării realizate de utilizator; ontologia trebuie să rămână neschimbată

Scenariu de succes:

1. Utilizatorul porneşte aplicaţia
2. Selectează butonul *Start RacerPorter* din meniul *Racer*
3. Apasă pe butonul numit *Load ontology* din cadrul meniului *Cancer ontology* şi selectează ontologia de cancer dorită
4. Apasă pe butonul *Create lists related to cancer* din meniul *Cancer ontology* pentru crearea listelor necesare procesării textelor
5. Apasă butonul *Close connection ontology* din meniul *Cancer ontology* pentru închiderea conexiunii cu această ontologie
6. Apasă pe butonul *Load ontology* din meniul *Arguments ontology* şi selectează ontologia de argumentare dorită
7. Apasă pe butonul *Create lists related to arguments* din meniul *Arguments ontology* pentru crearea listelor necesare procesării textelor
8. Se apasă pe butonul *Set corpus* din meniul *Arguments ontology* pentru încăcarea documentelor text în corpus
9. Pentru identificarea concluziilor şi a premiselor se va apăsa *Identify claim*, respectiv *Identify premise* din cadrul aceluiaşi meniu intitulat, *Arguments ontology*
10. Pentru salvare se alege *Save ontology* din meniul *Arguments ontology*
11. Selectează butonul aferent interogării pe care doreşte să o realizeze
12. Sistemul afişează răspunsul aferent interogării

Secvenţe alternative:

Secvenţele alternative sunt cele ilustrate mai sus pentru cazul de utilizare *Identificarea argumentelor*.

4.4.3 Vizualizarea adnotărilor

Titlu: Vizualizarea adnotărilor

Rezumat: Acest caz de utilizare permite utilizatorului să vizualizeze adnotările efectuate asupra documentelor din corpus

Actori: Utilizatorul (primar)

Precondiţii: Trebuie să existe documente în corpus, acestea necesitând procesare

Postcondiţii: Utilizatorul vede rezultatul adnotării pe documentul selectat

Scenariu de succes:

1. Utilizatorul porneşte aplicaţia
2. Acesta bifează căsuţa aferentă afişării interfeţei GATE
3. Selectează documentul dorit
4. Vizualizează rezultatul adnotării asupra documentului ales

Secvenţe alternative:

4. Sistemul întâmpină probleme legate de generarea de adnotări
   4.1. Utilizatorul este informat asupra erorii şi a cauzei acesteia
   4.2. Şcenariul se întoarce la pasul 2

Capitolul 5

Proiectare de Detaliu şi Implementare

În acest capitol o să prezint arhitectura generală a sistemului, fiecare componentă din care este formată fiind descrisă în parte. De asemenea, o să ilustrez algoritmul , cât şi pattern-urile lingvistice care stau la baza detecţiei automate de argumente în cadrul documentelor text. Diagrama de clase, de pachete, cât şi o prezentare a interfeţei grafice se regăseşte în acest capitol.

5.1 Arhitectura sistemului

5.1.1 Prezentare generală a arhitecturii sistemului

În această secţiune voi prezenta arhitectura sistemului şi modul în care interacţionează componentele între ele. Înainte de a prezenta şi descrie arhitectura sistemului, vreau să enumăr câţiva dintre factorii cei mai importanţi care au influenţat modul în care este dezvoltat sistemul.

Sistemul este bazat pe procesarea limbajului natural (NLP), această tehnică influenţând în mare măsură arhitectura sistemului. Procesarea limbajului natural foloseşte o abordare bazată pe pipeline-uri, secvenţe de procese care sunt aplicate pe documente. Pentru a obţine un rezultat, sistemul aplică o serie de transformări asupra limbajului natural, iar rezultatul din transformarea curentă serveşte ca dată de intrare pentru următoarea transformare. Astfel, la finalul acestui proces iterativ rezultatul va conţine toate modificările aplicate pas cu pas de aceste secvenţe de procesare.

De asemenea, sistemul se bazează şi pe ontologii, unde salvează informaţiile extrase din documentele text utilizate scrise în limbaj natural, sub formă de concepte, relaţii şi instanţe.

Arhitectura sistemului este prezentată în figura 5.1. Sistemul are patru componente majore: Interfaţa GATE Editor (*User Interface GATE Editor*), Componenta de procesare

de text (*Text Processing Component*), Componenta de identificare a argumentelor (*Argument Identification Component*) şi modulul de cunoştinţe. Componenta de stocare a cunoştinţelor este formată din TBox-ul şi ABox-ul aferent ontologiilor de argumentare şi de cancer. Sistemul lucrează îndeaproape cu corpusul format din documentele text.

Primul nivel este alcătuit din Editorul GATE [21] şi o interfaţă pentru interogările realizate pe ontologia de argumentare. Al doilea nivel este compus din componenta de procesare de text care realizează transformările necesare pentru extragerea argumentelor din textele de intrare. Modulele de procesare a argumentelor au scopul de a identifica propoziţiile argumentative în cadrul textului. Utilizând TBox-ul şi ABox-ul sistemul poate să salveze în cadrul ontologiei noile argumente. Ontologia de cancer este folosită pentru creearea listelor de cuvinte utilizate, în cadrul regulilor JAPE de identificare a argumentelor.

TBox-ul legat de argumente salvează definiţiile unui *Argument* realizat din *Claim* şi una sau mai multe *Premise*. ABox-ul corespunzător argumentelor conţine instanţele de caractere pe care aplicaţia le găseşte în interiorul documentelor text, de asemenea fiind utilizat pentru generarea listelor *ClaimIndicator*, *PremiseIndicator*, *VerbRelatedToClaim* şi *VerbRelatedToPremise*..

ABox-ul legat de cancer este utilizat pentru generarea listelor *CancerRelatedWords*, *"PeopleInvolved"*, *"DomainsAffected"*, *"Quantifiers"*, *"FactorsOfCancer"*.

5.1.2 Interacţiunile dintre componentele sistemului

În figura 5.1 utilizatorul prin intermediul interfeţei grafice introduce o interogare. Aceasta este procesată de către componenta numită *Text Processing Component*. Intrarea sistemului, documentele text sunt procesate de asemenea cu această componentă, care aplică o secvenţă de procese pentru a extragerea cuvintelor. Ea este realizată pe pipeline-ul ANNIE, care face parte din framework-ul GATE.

Pipeline-ul ANNIE este alcătuit din următoarele: *Tokenizer*, *POS Tagger*, *Gazetteer Sentence Splitter* şi *JAPE Transducer*.

Tokenizer împarte textul în tokeni simpli, cum ar fi numere, semne de punctuaţie şi cuvinte de diferite tipuri. *ANNIE Sentence Splitter* segmentează textul în propoziţii, în timp ce *Tagger-ul* produce etichete corespunzătoare fiecărei părţi de vorbire a fiecărui cuvânt sau simbol.

Lista "lookup" conţinută de *Gazetteer* este utilizată în faza iniţială a regulilor de detectare a adnotărilor şi în procesarea resurselor. Framework-ul are abilitatea de a cascada reguli JAPE înlănţuite.

Adnotările identificate prin intermediul componentei precedente sunt folosite ca şi intrare pentru următoarea parte din sistem, şi anume cea de detectare a argumentelor. Pentru îndeplinirea acestei sarcini, pipeline-ul are un set de reguli JAPE utilizate pentru identificarea diferitelor pattern-uri din text, care împreună vor ajuta la realizarea observării structurii argumentelor în cadrul textelor. Cu ajutorul ABox-ului celor două ontologii (de argumentare şi de cancer) se vor identifica şi forma listele de cuvinte necesare care vor ajuta

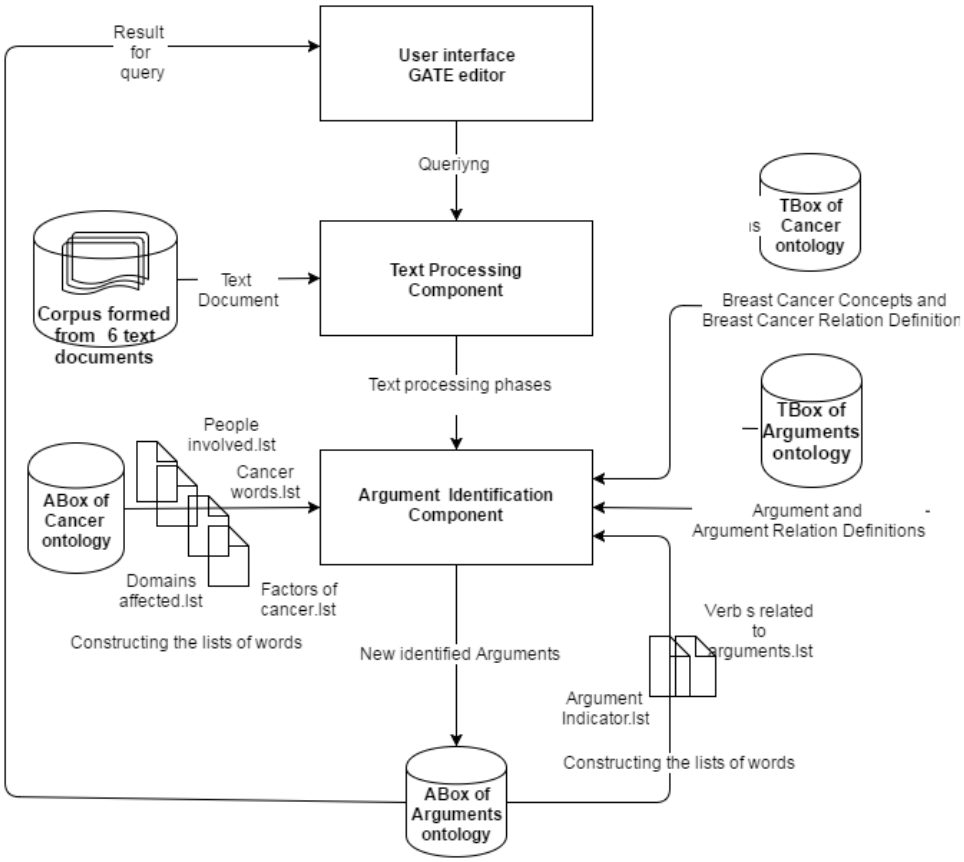

Figura 5.1: Arhitectura sistemului de detectare automată a argumentelor

la recunoaşterea argumentului. TBox-ul ontologiei de argumentare ilustrează relaţiile şi atributele dintre instanţele care au să fie salvate în cadrul acesteia.

Interogarea formulată la intrarea sistemului se aplică asupra ABox-ului de argumente identificate anterior. Rezultatul poate să fie observat prin intermediul interfeţei grafice.

5.2 Prezentare generală a sistemului

În această secţiune o să prezint arhitectura generală a sistemului descris în figura 5.1. Sistemul este dezvoltat utilizând limbajul Java, fiind strâns integrat cu framework-ul GATE. Datorită acestei integrări, aplicaţia prezintă două coduri sursă diferite. Primul este codul sursă Java, unde este descrisă logica de business, celălalt cod conţinând reguli gramaticale utilizate de framework-ul GATE pentru procesarea textului din cadrul corpusului. Pentru a prelua avantajul posibilităţii oferite de framework-ul GATE, codul Java a fost integrat cu cel realizat pentru detectarea automată a argumentelor în cadrul textelor.

5.2.1 Diagrama de pachete

Obiectivul subsistemului Java este de a automatiza procesul de detecţie a argumentelor şi de a furniza business-ul logic necesar regulilor JAPE. Astfel, subsistemul Java poate să fie văzut ca un mediu în care framework-ul GATE rulează pentru a îndeplinii scopul tezei.

Figura 5.2 ilustrează modul în care sunt grupate resursele sistemului în ceea ce priveşte partea de business. Aceasta este formată din următoarele pachete: pachetul de start reprezintă punctul de start al sistemului, acesta conţinând instanţa prin intermediul căreia procesul de detecţie automată a argumentelor începe. În pachetul *view* se regăseşte, clar implementată interfaţa grafică cu ajutorul cărei utilizatorul interacţionează. Pachetul *service* conţine două servicii, unul pentru gestionarea corpusului de documente şi cel de-al doilea pentru manipularea adnotaţiilor care sunt extrase din texte. Tot aici se regăsesc şi interfeţele care sunt implementate de cele două servicii.

În pachetul *racer* se regăseşte serviciul care face referire la gestionarea ontologiei de cancer şi de argumentare. Clasa implementează interfaţa aferentă acesteia. Pachetul *util* conţine constante, stocarea acestora într-un singur loc ajutând la diminuarea dificultăţiilor, în cazul apariţiei unei actualizări a codului. Pachetul *pipeline* conţine secvenţa de procesare a documentelor, aceasta moştenind clasa *PipelineWrapper*, bazându-se pe un set propriu de reguli gramaticale salvate în fişiere JAPE. Iar ultimul, însă nu cel din urmă, pachetul *controller*, instanţiază în cadrul clasei regăsite aici clasele referitoare la serviciile folosite şi interfaţa utilizator.

Subsistemul GATE este piesa centrală a întregului sistem. Obiectivul acestuia este de a adnota automat argumentele din cadrul documentelor text prin aplicarea tehnicilor de procesare a limbajului natural. Aceste sistem este compus din pipeline şi reguli gra-

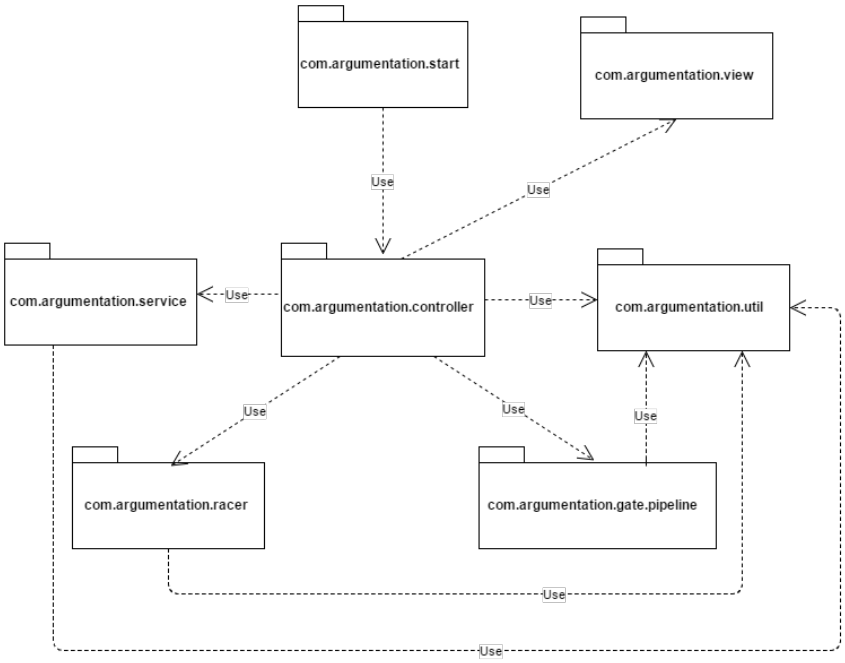

Figura 5.2: Diagrama de pachete a subsistemului Java

maticale JAPE, fiecare îndeplinind o anumită sarcină pentru realizarea obiectivului tezei. După cum se poate observa şi în figura 5.3 subsistemul este format din patru foldere. Pachetul *grammar* conţine regulile gramaticale JAPE definite pentru adnotarea sematică a textului. Pachetul *ontology* este destinat ontologiilor utilizate de aplicaţie, şi anume cea de argumentare şi de cancer la sân. Pachetul *copus* şi *datastore* stochează documentele care conţin textele extrase din articolele de cancer, descrise în limbaj natural, respectiv conţine corpusul care este salvat într-o unitate de stocare persistentă, fiind încărcată de fiecare dată când aplicaţia este folosită.

5.2.2 Diagrama de clase

Figura 5.4 ilustrează diagrama de clase care conţine principalele clase implementate ale sistemului. Clasele care apar aici sunt în concordanţă cu pachetele descrise în subsecţiunea anterioară.

Clasa de nivel superior *CommonStart* conţine metoda *main* executată de program pentru începerea rulării. Aceasta creează o instanţă a clasei *Controller*. Clasa *Controller* reprezintă principala componentă a arhitecturii sistemului, ea conţinând instanţe ale claselor: *MainWindow*, *AnnieAnalyzerPipeline* şi a interfeţelor *AnnotationService*, *OntologyRacerService*, *CorpusService*. Această clasă controlează întregul flux de date, conţinând metodele utilizate pentru procesarea evenimentelor realizate de butoanele aflate pe interfaţa grafică. Funcţiile conţin logică pentru procesarea ontologii, detectarea argumentelor, salvarea acestora şi realizarea diferitelor interogări asupra bazei de date.

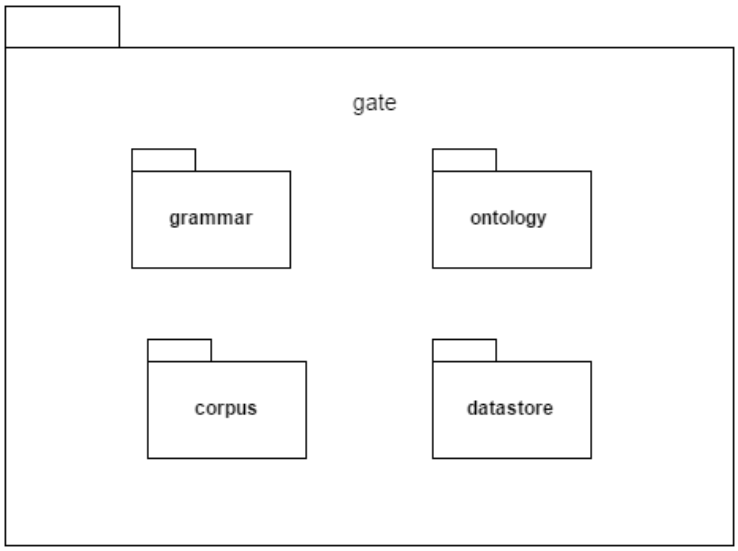

Figura 5.3: Diagrama de pachete a subsitemului GATE

Clasa *AnnieAnalyzerPipeline* extinde clasa *PipelineWrapper*, care înglobează pipeline-ul utilizat pentru procesarea documentelor text. La baza acestuia se regăsesc regulile JAPE şi listele de cuvinte *.lst* stocate în pachetul destinat resurselor, de care programul GATE are nevoie pentru identificarea şi extragerea argumentelor. Clasa *PipelineWrapper* conţine resursele necesare implementării pipeline, şi anume setează corpusul şi fişierul *.gapp*.

Pentru a realiza un nivel ridicat de modularitate o serie de servicii trebuie realizate, unde fiecare îndeplineşte un anumit task. Astfel, serviciul *CorpusServiceImpl* care implementează interfaţa *CorpusService* are sarcina de extragere a documentelor text din Datastore, formând corpusul cu acestea. De asemenea, serviciul *OntologyRacerServiceImpl* este o clasă mai complexă conţinând metode pentru crearea, încărcarea, extragerea indiviziilor, salvarea şi interogarea ontologiilor. Adnotările sunt extrase din cadrul textelor prin intermediul metodelor realizate în clasa *AnnotationServiceImpl*, acest serviciu fiind responsabil şi pentru formarea listelor de cuvinte *.lst* utilizate pentru detecţia automată a argumentelor. Aceste servicii implementează interfeţele aferente fiecăreia.

5.3 Specificaţiile datelor de intrare şi ieşire

În această secţiune o să prezint formatul datelor de intrare şi ieşire a sistemului şi ce contrângeri trebuie îndeplinite.

Prima intrare a sistemului constă dintr-o serie de documente text grupate într-un corpus. În cazul nostru, corpusul este format din texte extrase din articole cu referire la cancerul la sân. Sistemul realizează algoritmul implementat pe fiecare dintre aceste documente, extragând informaţiile necesare.

Deoarece sistemul este dezvoltat utilizând framework-ul GATE se utilizează funcţion-

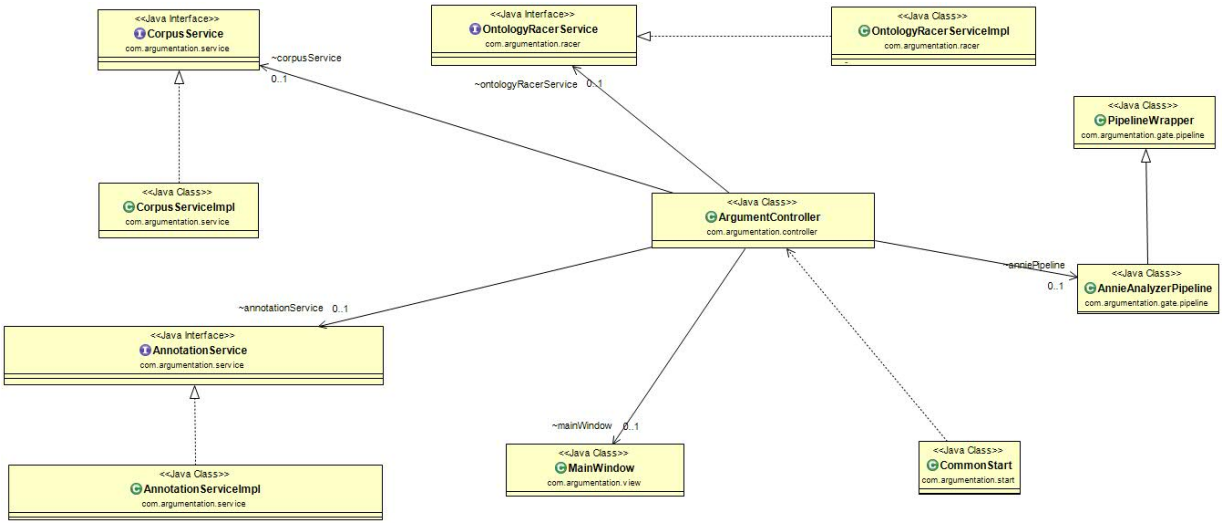

Figura 5.4: Diagrama de clase

alităţiile oferite de acesta în crearea corpusului. Mai exact, aplicaţia creează un mediu persistent şi adaugă documentele necesare bazei de date. La runtime, sistemul încărcă baza de date, realizează un nou corpus, adăugând în acelaşi timp documentele necesare din baza de date în corpusul nou creat.

Pentru documentele de intrare sistemul utilizează o convenţie asupra denumirii acestora, având rolul de eficientizare a procesului. Acestea sunt denumite utilizând trei numere, de exemplu: 000 urmate de extensia .txt. Astfel, pentru primul document avem notarea ”000.txt”, pentru al doilea ”001.txt” şi aşa mai departe.

A doua intrare a sistemului este reprezentată de ontologia de cancer şi de cea de argumentare. Acestea sunt realizate utilizând limbajul *racer*. TBox-ul acestora este folosit pentru ilustrarea diferitelor concepte şi relaţii care pot avea loc între instanţe. De asemenea, ABox-ul este utilizat pentru formarea listelor de cuvinte *.lst* utilizate în detectarea argumentelor într-un mod automat. Ambele fişiere prezintă extensia *.racer*, fiind denumite *breastCancer.racer* şi *ontologieArgumente.racer*.

Ieşirea sistemului constă în argumentele identificate şi salvate în ABox-ul referit de ontologia de argumentare. De asemnea, aplicaţia permite convertirea ontologiilor *.racer* în *.owl*, acestea fiind denumite *breastCancer.owl* şi *ontologieArgumente.owl*.

5.4 Componenta de procesare a textului

În această secţiune am să prezint în detaliu structura componentei de procesare a textului, aceasta fiind realizată din pipeline-ul ANNIE [22], după cum se poate observa în figura 5.5. Scopul acestei componente este de a împărţii documentul în tokeni şi de ai adnota pe fiecare cu informaţie sintactică şi morfologică.

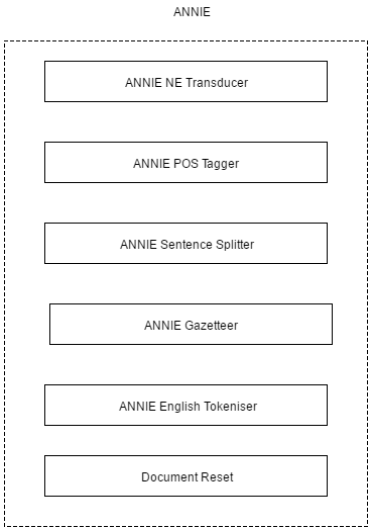

Figura 5.5: Resursele Pipeline-ului ANNIE

Prima componentă este *Document Reset* care şterge toate seturile de adnotări anterioare, readucând documentele la starea originală. A doua resursă este *ANNIE English Tokeniser* care are ca şi scop împărţirea textului în fragmente simple de text, numiţi tokeni. Acest tokeniser este case-sensitive, fiind capabil să facă distincţia între cuvintele formate din litere de tipar sau mici, de asemenea poate distinge diferite semne de punctuaţie şi numere. *Sentence Splitter-ul* prezintă rolul de a împărţi textul din cadrul documentului în propoziţii. *ANNIE Gazetteer* prezintă liste de cuvinte, aceste fiind căutate în cadrul documentelor aflate în corpus. *ANNIE POS Tagger* produce o adnotare pentru fiecare simbol şi cuvânt în ceea ce priveşte parte de vorbire din care este alcătuită. Şi ultima, dar nu cea din urmă componentă *ANNIE NE Transducer* prezintă regulile JAPE pentru adnotarea fragmentelor din cadrul textelor.

Componenta *ANNIE English Tokeniser* generează două tipuri de adnotări: o adnotare *SpaceToken* reprezentând un spaţiu alb şi adnotarea *Token* care poate să fie de mai multe feluri:

- word - care defineşte un set contiguu de caractere care pot să fie litere mari, mici inclusiv cratimă
- număr - defineşte o combinaţie consecutivă de cifre
- simbol - două categorii de simboluri sunt acceptate: simbolurile valutare şi cele regulate
- punctuaţia - trei tipuri de semne de punctuaţie sunt acceptate: punctuaţia de început (de exemplu "("), punctuaţia de final (de exemplu ")"), precum şi alte semne de punctuaţie (de exemplu ":")

5.5 Componenta de identificare a argumentelor

În această secţiune am să descriu în detaliu modul de identificare automat în cadrul unui text a argumentelor. Această componentă primeşte ca date de intrare cele două ontologii şi regulile JAPE necesare detecţiei.

Pentru identificarea concluziei şi a premisei folosim reguli JAPE (Java Annotation Patterns Engine) [21]. JAPE permite recunoaşterea expresiilor regulate în cadrul adnotăriilor din interiorul documentelor. O gramatică JAPE conţine un set de faze, fiecare fiind formată dintr-un set de pattern-uri/acţiuni. Fazele rulează secvenţial constituind o cascadare de stări finite în interiorul documentelor. Partea stângă a regulii (LHS) este alcătuită din descrierea unui pattern de adnotări. Partea dreaptă (RHS) reprezintă declaraţiile de manipulare a adnotaţiilor. Adnotările corespunzătoare în LHS pentru o regulă pot fi menţionate în RHS cu ajutorul unor etichete care sunt ataşate la elementele de pattern.

Algorithm 1 Algoritmul de detecţie automată al argumentelor

Input: $\mathcal{O}$ - breast cancer ontology; $\mathcal{A}(Tbox)$ - argumentation model (arg Tbox); $\mathcal{C}$ - corpus of medical documents

Output: $\mathcal{A}$, Abox containing mined arguments

```
1  foreach d ∈ C do
2      Sentences ← Tokenise(d)
3      foreach s ∈ Sentences do
4          if ∃ CC ∈ s then
5              if ClaimMacro ∈ s then
6                  if ofstClaimMacro ≤ ofstCC then
7                      Claim ← words[ofstClaimMacro, ofstCC]
                        if PremiseMacro ∈ s then
8                          Premise ← words[ofstCC, ofstPremiseMacro]
9                  else
10                     Claim ← words[ofstCC, ofstClaimMacro]
                        if PremiseMacro ∈ s then
11                         Premise ← words[ofstPremiseMacro, ofstCC]
12         else
13             if ClaimMacro ∈ s then
14                 Claim ← words[ofstClaimMacro, ofstPct] else
15                     if PremiseMacro ∈ s then
16                         Premise ← words[ofstPremiseMacro, ofstPct]
```

Abordarea de nivel ridicat pentru detecţia automată a argumentelor este formalizată

în cadrul algoritmului ilustrat mai sus. Prima dată sistemul analizează fiecare document d din corpusul $\mathcal{C}$ de documente medicale disponibile (linia 1). Fiecare document d este împărţit (liniile 2 şi 3) şi pentru fiecare propoziţie s sistemul verifică dacă o conjuncţie coordonatoare (CC) există în propoziţie (linia 4). Dacă asemenea CC este găsită, algoritmul caută după instanţe ale conceptului *ClaimMacro*, ca un posibil indicator a unei concluzii a unui argument (linia 5). Aplicaţia se uită după conjuncţie coordonatoare şi verifică dacă *PremiseMacro* este prezent, atunci *Premise* este identificată (liniile 8 and 9). Dacă offset-ul al unui *ClaimMacro* este mai mare decât offset-ul unui CC atunci sistemul caută *Claim* şi *Premise* în interiorul propoziţiilor (liniile 11, 12 şi 13).

Dacă în interiorul propoziţiilor nu se găseşte conjuncţie coordonatoare (linia 14) atunci o instanţă de *ClaimMacro* sau de *PremiseMacro* este căutată. Depinzând de prezenţa acesteia, sistemul determină dacă propoziţia poate să fie asociată pentru o adnotare, alegând între *Claim* sau *Premise* (liniile 15, 16, 17, 18, 19). Dacă propoziţia nu conţine conjuncţie coordonatoare sau *ClaimMacro* sau *PremiseMacro* atunci sistemul analizează următoarea propoziţie din setul *Sentences*.

5.5.1 Structura indicatorilor de concluzie

Sistemul utilizează macrouri, indicatori pentru a decide dacă o propoziţie sau o parte din aceasta este un potenţial candidat pentru a fi *Claim* sau *Premise*.

Următoarele template-uri sunt folosite pentru identificarea concluziilor:
i) *Ci*: indicator de concluzie;
ii) *CibPe*: indicator de concluzie plus persoane implicate;
iii) *CibPebVc*: indicator de concluzie urmat de persoane implicate şi verbe specifice concluziei;
iv) *CibVc*: indicatori de concluzie urmaţi de verbe specifice concluziei;
v) *ElOfCnbCw*: cuvânt sau cuvinte care exprimă elemente de cancer urmate de un cuvânt care se referă la cancer; după acest macro poate să fie prezent sau nu un verb care se referă la concluzie;
vi) *QbPebVc*: persoanele pot avea cuantificatori reprezentând numărul acestora.

Ne bazăm pe indicatorii textuali, clasificaţi astfel:

$Ci \sqcup CibPe \sqcup CibPebVc \sqcup CibVc \sqcup ElOfCnbCw \sqcup QbPebVc \sqsubseteq ClaimIndicator \sqsubseteq MacroIndicator$.

Macro-ul *ClaimIndicator before People (MI_CibPe)* conţine *ClaimIndicator* urmat de persoane implicate în domeniul medical al cancerului la sân:

MI : ClaimIndicator before People ≒ *MI_CibPe*

ci : *ClaimIndicator*
pe : *Person*
(*ci*, *pe*) : *before*
(*sentence*, *ci*) : *hasToken*
(*sentence*, *pe*) : *hasToken*

Ex_1 : ["we may [infer]$_{ci}$ that [woman]$_{pe}$"]$_{CibPe}$
Ex_2 : ["this [bears out]$_{ci}$ the point that [doctors]$_{pe}$"]$_{CibPe}$
Ex_3 : ["it [follows]$_{ci}$ that [patiences]$_{pe}$"]$_{CibPe}$

Propoziţia Ex_1 este formalizată ca o instanţă de MI_CibPe în Exemplul 9.

Exemplul 9 *Macro-ul CibPe conţine indivizii ci de tip ClaimIndicator şi pe de tipul Person. În sentence (liniile 5.3, 5.4), ci este localizat înaintea lui pe (linia 5.2). Textul ambelor instanţe este prezentat cu ajutorul rolului hasText (linia 4.45).*

$$ci : ClaimIndicator, pe : Person \tag{5.1}$$
$$(ci, Pe) : before \tag{5.2}$$
$$(sentence, ci) : hasToken \tag{5.3}$$
$$(sentence, pe) : hasToken \tag{5.4}$$
$$(ci, "infer") : hasText, (pe, "woman") : hasText \tag{5.5}$$

Macro-ul *ClaimIndicator before People before Verb related to Claim (MI_CibPebVc)* este o generalizare a macro-ului *ClaimIndicator before People (MI_CibPe)* plus verbe referitoare la concluzie (vezi Exemplul 3):

MI : ClaimIndicator before People before Verb ≒ *MI_CibPebVc*

ci : *ClaimIndicator*
pe : *Person*
vc : *VerbRelatedToClaim*
(*ci*, *pe*) : *before*
(*pe*, *vc*) : *before*
(*sentence*, *ci*) : *hasToken*
(*sentence*, *pe*) : *hasToken*
(*sentence*, *vc*) : *hasToken*

Ex_1 : ["we can [conclude]$_{ci}$ that [doctors]$_{pe}$ [identified]$_{vc}$"]$_{CibPebVc}$
Ex_2 : ["[so]$_{ci}$ the [key informants]$_{pe}$ [provides]$_{vc}$"]$_{CibPebVc}$
Ex_3 : ["it [follows]$_{ci}$ that [people]$_{pe}$ [estimated]$_{vc}$"]$_{CibPebVc}$

Macro-ul *ClaimIndicator before Verb related to Claim (MI_CibVc)* conţine expresii care sunt instanţe a conceptului *ClaimIndicator* urmate de verbe referitoare la concluzii (vezi Exemplul 3):

MI : ClaimIndicator before Verb $\doteqdot$ *MI_CibVc*

ci : *ClaimIndicator*
vc : *VerbRelatedToClaim*
(ci, vc) : *before*
$(sentence, ci)$: *hasToken*
$(sentence, vc)$: *hasToken*
Ex_1 : ["[therefore]$_{ci}$ [exemplifies]$_{vc}$"]$_{CibVc}$
Ex_2 : ["[so]$_{ci}$ [highlighted]$_{vc}$"]$_{CibVc}$
Ex_3 : ["[thus]$_{ci}$ [accepted]$_{vc}$"]$_{CibVc}$

Expresia Ex_1 este formalizată ca o instanţă de MI_CibVc după cum Exemplul 10 ilustrează.

Exemplul 10 *Macro-ul CibVc conţine indivizii ci de tipul ClaimIndicator şi vc de tipul VerbRelatedToClaim. În interiorul sentence (line 5.8), ci este localizat înaintea lui vc (linia 5.7). Textul ambelor instanţe este prezentat cu ajutorul rolului hasText (linia 5.9).*

$$ci : ClaimIndicator, vc : VerbRelatedToClaim \tag{5.6}$$
$$(ci, vc) : before \tag{5.7}$$
$$(sentence, ci) : hasToken, (sentence, vc) : hasToken \tag{5.8}$$
$$(ci, "therefore") : hasText, (vc, "exemplifies") : hasText \tag{5.9}$$

Macro-ul *Elements of Cancer before Cancer related words (MI_ElOfCnbCwVc)* conţine expresii care sunt compuse din unul sau mai multe cuvinte care sunt instanţe ale conceptului *ElementsOfCancer* plus un cuvânt din domeniul cancerului şi opţional având un verb cu referire la concluzie (vezi Exemplul 3):

MI : Elements of Cancer before Cancer words $\doteqdot$ *MI_ElOfCnbCw*

$elOfCn$: *ElementsOfCancer*
cw : *CancerWords*
$(elOfCn, cw)$: *before*
$(sentence, elOfCn)$: *hasToken*
$(sentence, cw)$: *hasToken*
Ex_1 : ["the [risk]$_{elOfCn}$ of [breast cancer]$_{cw}$"]$_{ElOfCnbCwVc}$
Ex_2 : ["these [factors]$_{elOfCn}$ of [cancer]$_{cw}$ [were equaled]$_{vc}$"]$_{ElOfCnbCwVc}$

Macro-ul *Qualifiers before People (MI_QbPebVc)* conţine cuantificatori înaintea instanţelor conceptului *PeopleInvolved*, ceea ce înseamnă că mai multe persoane sunt implicate:

MI : Qualifiers before People before Verb $\doteqdot$ *MI_QbPebVc*

q : *Qualifier*
pe : *Person*
vc : *VerbRelatedToClaim*
(q, pe) : *before*
(pe, vc) : *before*
$(sentence, q)$: *hasToken*
$(sentence, pe)$: *hasToken*
$(sentence, vc)$: *hasToken*

Ex_1 : ["[many]$_q$ [woman]$_{pe}$ [provides]$_{vc}$"]$_{QbPebVc}$
Ex_2 : ["[many]$_q$ [survivors]$_{pe}$ [accepted]$_{vc}$"]$_{QbPebVc}$

5.5.2 Structura indicatorilor de premisă

Următoarele template-uri sunt folosite pentru identificarea premiselor:
i) *Pi*: indicatori de premisă;
ii) *PibPe*: indicatori de premisă plus persoane implicate;
iii) *PibPebVp*: indicatori de premisă urmate de persoane implicate şi verbe specifice premisei;
iv) *PibVp*: persoane implicate urmate de verbe pentru premisă;
v) *ElOfCnbCw*: unul sau mai multe cuvinte exprimând elemente de cancer urmate de un cuvânt care se referă la cancer; după acest macro pot exista verbe care se referă la premisă;
vi) *DbVp*: cuvinte care exprimă domeniile afectate de cancerul la sân urmate de verbe specifice premisei;

Ne bazăm pe indicatorii textuali, clasificaţi astfel:

$Pi \sqcup PibPe \sqcup PibPebVp \sqcup PibVp \sqcup ElOfCnbCw \sqcup DbVp \sqsubseteq PremiseIndicator \sqsubseteq MacroIndicator$.

Macro-ul *PremiseIndicator before People (MI_PibPe)* conţine expresii care sunt instanţe ale conceptului care conţine *PremiseIndicator* urmate de persoanele implicate în domeniul medical al cancerului la sân:

MI : PremiseIndicator before People ≒ *MI_PibPe*

pi : *PremiseIndicator*
pe : *Person*
(pi, pe) : *before*
$(sentence, pi)$: *hasToken*
$(sentence, pe)$: *hasToken*
Ex_1 : ["[in view of the fact]$_{pi}$ that [woman]$_{pe}$"]$_{PibPe}$
Ex_2 : ["[as shown] $_{pi}$ by [doctors]$_{pe}$"]$_{PibPe}$
Ex_3 : ["[since]$_{pi}$ [patiences]$_{pe}$"]$_{PibPe}$

Macro-ul *PremiseIndicator before People before Verb related to premise (MI_PibPebVp)* este generalizare a macro-ului *PremiseIndicator before People (MI_PibPe)* plus verbe care fac referire la premise (vezi Exemplul 4):

MI : PremiseIndicator before People before Verb ≒ *MI_PibPebVp*

pi : *PremiseIndicator*
pe : *People*
vp : *VerbRelatedToPremise*
(pi, pe) : *before*, (pe, vp) : *before*
$(sentence, pi)$: *hasToken*
$(sentence, vp)$: *hasToken*
$(sentence, pe)$: *hasToken*
Ex_1 : ["as [evidenced]$_{pi}$ by [people]$_{pe}$ [received]$_{vp}$"]$_{PibPebVp}$
Ex_2 : ["[assuming]$_{pi}$ that [doctors]$_{pe}$ [observed]$_{vp}$"]$_{PibPebVp}$
Ex_3 : ["[because]$_{pi}$ the [key informants]$_{pe}$ [were noted]$_{vp}$"]$_{PibPebVp}$

Expresia Ex_1 este formalizarea instanţei macro indicatorului MI_PibPebVp, după cum Exemplul 11 ilustrează.

Exemplul 11 *Macro-ul PibPebVp conţine indivizii pi de tipul PremiseIndicator, pe de tipul People şi vp de tipul VerbRelatedToPremise. pi este localizat înaintea lui pe şi pe este localizat înaintea lui vp (linia 5.11). Textul instanţelor este prezentat cu ajutorul rolului hasText (linia 5.15).*

$$pi : PremiseIndicator, pe : People, vp : VerbRelatedToPremise \tag{5.10}$$
$$(pi, pe) : before, (pe, vp) : before \tag{5.11}$$
$$(sentence, pi) : hasToken, (sentence, vp) : hasToken \tag{5.12}$$
$$(sentence, pe) : hasToken \tag{5.13}$$
$$(pi, "evidenced") : hasText, (pe, "people") : hasText \tag{5.14}$$
$$(vp, "received") : hasText \tag{5.15}$$

Macro-ul *PremiseIndicator before Verb (MI_PibVp)* conţine expresii care sunt instanţe ale conceptului *PremiseIndicator* urmat de verbe specifice premisei (vezi Exemplul 4):

MI : PremiseIndicator before Verb $\doteqdot$ *MI_PibVp*

pi : *PremiseIndicator*
vp : *VerbRelatedToPremise*
(pi, vp) : *before*
$(sentence, pi)$: *hasToken*
$(sentence, vp)$: *hasToken*

Ex_1 : ”[since]$_{pi}$ [according]$_{vp}$”]$_{PibVp}$
Ex_2 : ”[given]$_{pi}$ that [noted]$_{vp}$”]$_{PibVp}$
Ex_3 : ”[seeing]$_{pi}$ that [served]$_{vp}$”]$_{PibVp}$

Macro-ul *Elements of Cancer before Cancer related words (MI_ElOfCnbCw)* conţine expresiile compuse din unul sau mai multe cuvinte instanţe ale conceptului *ElementsOfCancer* plus un cuvânt despre domeniul de cancer şi opţional urmat de un verb cu referire la premisă (vezi Exemplul 4):

MI : Elements of Cancer before Cancer words $\doteqdot$ *MI_ElOfCnbCw*

$elOfCn$: *ElementsOfCancer*
cw : *CancerWords*
$(elOfCn, cw)$: *before*
$(sentence, elOfCn)$: *hasToken*
$(sentence, cw)$: *hasToken*

Ex_1 : ”[the [risk]$_{elOfCn}$ of [breast cancer]$_{cw}$ [was noted]$_{vp}$”]$_{ElOfCnbCw}$

Macro-ul *Domains affected before Verb (MI_DbVp)* conţine expresii care sunt compuse din cuvinte care fac referire la domeniul afectat de cancerul la sân urmat de verbe specifice premisei, vezi Exemplul 4:

MI : Domains affected before Verb $\doteqdot$ *MI_DbVp*

d : *Domains*
vp : *VerbRelatedToPremise*
(d, vp) : *before*
$(sentence, d)$: *hasToken*
$(sentence, vp)$: *hasToken*

Ex_1 : ”[Family history]$_{d}$ [regarding]$_{vp}$”]$_{DbVp}$
Ex_2 : ”[physical changes]$_{d}$ [resulting]$_{vp}$”]$_{DbVp}$

Expresia Ex_1 formalizează instanţa MI_DbVp după cum exemplul 12 ilustrează.

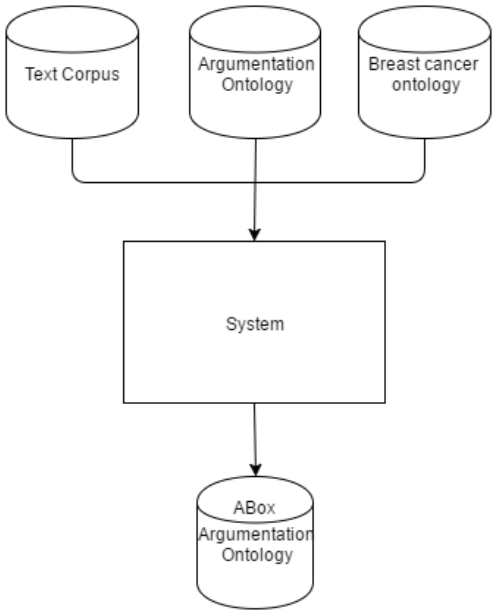

Figura 5.6: Componenta de reprezentare a cunoştinţelor

Exemplul 12 *Macro-ul DbVp macro conţine indivizii d de tipul Domains şi vp de tipul VerbRelatedToPremise. În interiorul sentence (linia 5.17), d este localizat înaintea lui vp (linia 5.16). Textul instanţelor este prezentat cu ajutorul rolului hasText (linia 5.18).*

$$d : Domains, vp : VerbRelatedToPremise, (d, vp) : before \tag{5.16}$$

$$(sentence, d) : hasToken, (sentence, vp) : hasToken \tag{5.17}$$

$$(d, "Familyhistory") : hasText, (vp, "regarding") : hasText \tag{5.18}$$

5.6 Componenta de cunoştinţe

În această secţiune o să prezint mai detaliat componenta formată din ontologii. În Figura 5.6 este reprezentată structura ontologiilor necesare pentru detectarea automată a argumentelor. Ambele ontologii sunt realizate cu ajutorul limbajului *Racer*, acestea fiind gestionate prin intermediul programului *RacerPro* şi a serviciului *OntologyService* implementat în cadrul sistemului.

Sistemul primeşte ca parametru de intrare corpusul format din documentele text şi cele două ontologii după cum se poate observa în Figura 5.6. Utilizând TBox-ul şi ABox-ul sistemul poate să salveze în cadrul ontologiei noile argumente. Ontologia de cancer este folosită pentru creearea listelor de cuvinte utilizate în cadrul regulilor JAPE, de identificare a argumentelor.

TBox-ul legat de argumente salvează definiţiile unui *Argument* realizat din *Claim*

şi una sau mai multe *Premise*, după cum se poate observa în următoarele câteva linii.

```
(equivalent argument (and (at-least 1 hasPremise) (exactly 1 hasClaim)))
```

ABox-ul corespunzător argumentelor conţine instanţele de caractere pe care aplicaţia le găseşte în interiorul documentelor text. De asemenea, ABox-ul este utilizat pentru generarea listelor *ClaimIndicator*, *PremiseIndicator*, *VerbRelatedToClaim* şi *VerbRelatedToPremise*, descrirea acestora fiind evidenţiată în exemplul următor:

```
(instance INDC1 CLAIMINDICATOR)
(instance INDP1 PREMISEINDICATOR)
(instance VC1 VERBRELATEDTOCLAIM)
(instance VP1 VERBRELATEDTOPREMISE)

(attribute-filler INDC1 "consequently" hasText)
(attribute-filler INDP1 "consequently" hasText)
(attribute-filler VC1 "reported" hasText)
(attribute-filler VP1 "were noted" hasText)
```

ABox-ul legat de cancer este utilizat pentru generarea listelor *CancerRelatedWords*, *"PeopleInvolved"*, *"DomainsAffected"*, *"Quantifiers"*, *"FactorsOfCancer"*, descrirea acestora fiind evidenţiată în exemplul următor:

```
(instance p1 People)
(instance d1 Domain)
(instance q1 Quantifier)
(instance t1 Treatment)
(instance s1 Symptom)
(instance c1 Characteristic)

(attribute-filler p1 "key informant" hasText)
(attribute-filler d1 "Health-related quality of life (HRQOL)" hasText)
(attribute-filler q1 "Many" hasText)
(attribute-filler t1 "tamoxifen" hasText)
attribute-filler s1 "genomic" hasText)
(attribute-filler c1 "breast cancer" hasText)
```

O listă de cuvinte *.lst* care ajută la identificarea argumentelor este ilustrată în Exemplul 13.

Exemplul 13 *O parte din lista indicatorilor de concluzie este formată din următoarele cuvinte:*
consequently

therefore
thus
so
hence
accordingly
we can conclude that
it follows that
we may infer that

O instanţă de argument salvată în cadrul ontologiei de argumentare este reprezentată astfel:

```
(instance A1 ARGUMENT)
(instance C1 CLAIM)
(instance P1 PREMISE)

(related A1 C1 hasClaim)
(related A1 P1 hasPremise)

(attribute-filler C1 "We report on the risk of breast cancer" hasText)
(attribute-filler P1 "according to histologic findings, the age at diagnosis
        of benign breast disease, the strength of the family history" hasText)
```

Pentru vizualizarea ontologiilor acestea sunt trasformate din fişiere cu extensia *.racer*, în fişiere cu extensia *.owl*. Astfel, atât ontologia *breastCancer.owl* cât şi *ontologieArgumente.owl* sunt alcătuite în mare parte din proprietăţi de obiecte şi clase, ca şi cele reprezentate mai jos:

```
<owl:ObjectProperty rdf:about="AFFECTDOMAINS">
   <rdfs:range>
      <owl:Class rdf:about="DOMAIN"/>
   </rdfs:range>
</owl:ObjectProperty>

<owl:Class rdf:about="PREMISEINDICATOR">
   <rdfs:subClassOf>
      <owl:Class rdf:about="ARGUMENTINDICATOR"/>
   </rdfs:subClassOf>
</owl:Class>
```

O instanţă în cadrul acesteia este scrisă în limbajul OWL astfel:

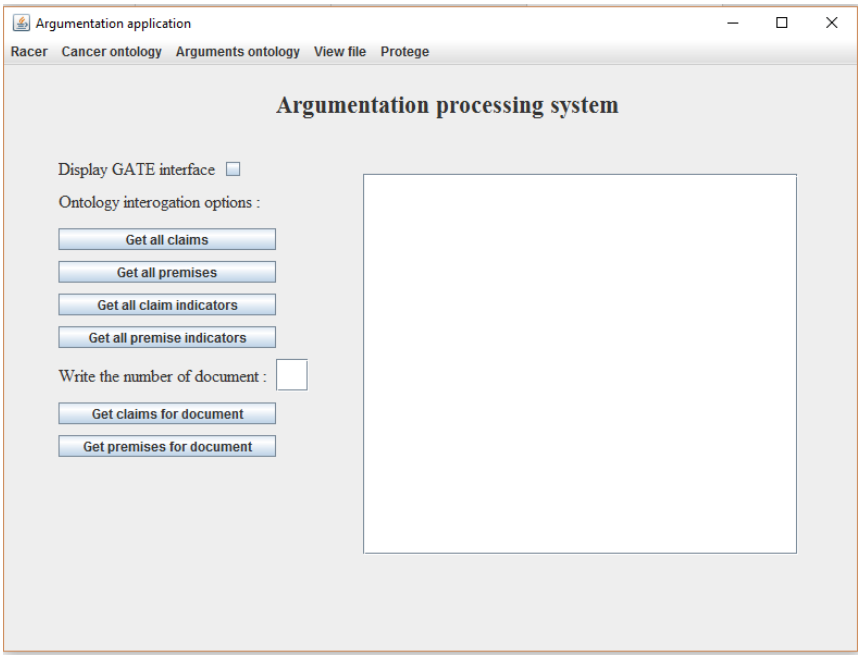

Figura 5.7: Interfaţa sistemului de detecţie automată a argumentelor

```
<SYMPTOM rdf:about="S2">
</SYMPTOM>

<CLINICALARGUMENT rdf:about="A">
     <HASCLAIM rdf:resource="C1"/>
     <HASPREMISE rdf:resource="P2"/>
     <HASPREMISE rdf:resource="P1"/>
</CLINICALARGUMENT>
```

5.7 Interfaţa grafică

Interfaţa grafică a sistemului este prezentată în Figura 5.7. După cum se poate observa există două interfeţe externe, una oferită de aplicaţia GATE pentru vizualizarea adnotărilor, iar cealaltă oferită de programul Protege pentru vizualizarea ontologiilor sub forma unui graf.

Interfaţa utilizator din Figura 5.7 este cea implementată de mine utilizând Java Swing, nefiind una complexă, având ca şi scop principal identificarea argumentelor şi realizarea de interogări asupra rezultatelor obţinute. Interfaţa este formată dintr-o singură fereastră, în care meniul se ocupă cu procesarea textului, iar butoanele care sunt vizibile realizând interogări asupra ontologiei.

Pentru ca interfaţa framework-ului GATE să fie vizibilă trebuie bifat checkbox-ul

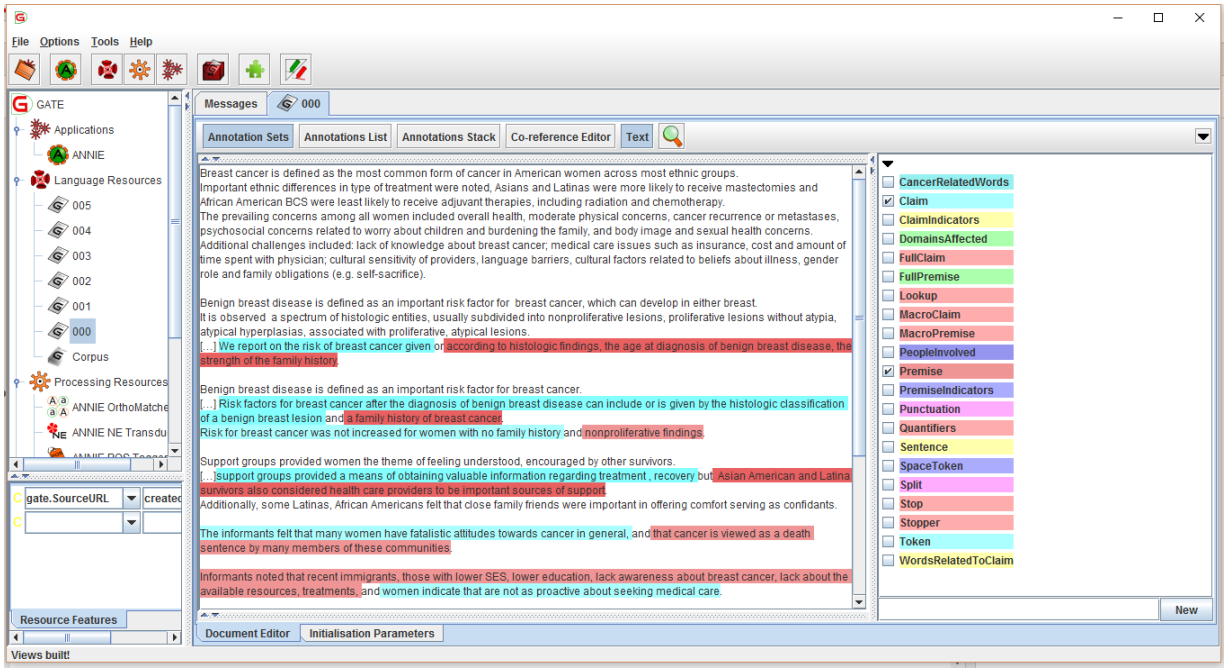

Figura 5.8: Vizualizarea adnotărilor în aplicaţia GATE

aferent textului *"Display GATE Interface"*. Aceasta este folosită de către utilizator în scopul de a vizualiza adnotările. Din fereastra de resurse se selectează documentul dorit şi astfel acesta va fi afişat în centrul interfeţei, ca şi în exemplul din Figura 5.8.

Pentru ca interfaţa aplicaţiei Protege să fie vizibilă trebuie selectat din meniul *Protege*, submeniul *Start Protege*. Odată deschisă aplicaţia, trebuie selectată ontologia pentru vizualizare şi apoi print-un simplu click pe clasele dorite acestea apar în spaţiul destinat afişării. Instanţele şi fiecare clasă din ambele ontologii sunt prezentate în figurile A.1, A.2 din Anexa A. În figura 7.1 este evidenţiată interfaţa Protege de observare a unei ontologii sub forma unui graf.

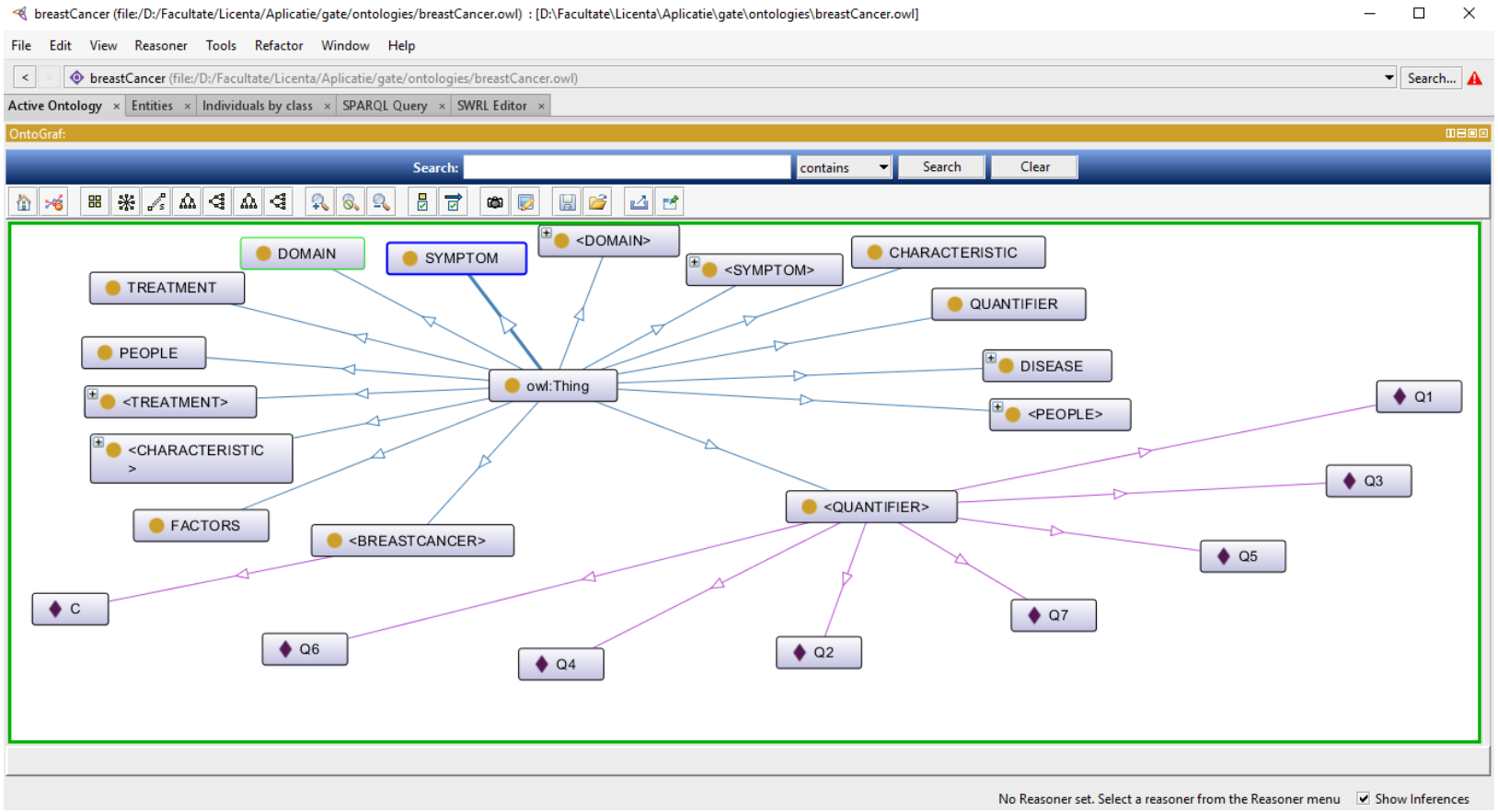

Figura 5.9: Vizualizarea ontologiei de cancer în aplicaţia Protege

Capitolul 6

Testare şi Validare

În acest capitol o să prezint cum am realizat testarea sistemului de detectare automată a argumentelor în cadrul documentelor text despre cancer la sân, precum şi rezultatele obţinute. Pentru evaluarea sistemului am realizat un corpus format din şase documente pe care le-am adnotat manual.

6.1 Testarea sistemului de adnotare

Pentru a testa sistemul dezvoltat am utilizat *Annotation Diff Tool* deţinut de framework-ul GATE. Acest tool compară două seturi de adnotări din acelaşi document. Rezultatul comparării este exprimat prin intermediul a trei parametrii diferiţi: *Precision*, *Recall* şi *F-measure*.

Precision măsoară câte adnotări au fost identificate corect din setul de adnotări identificate.

Recall măsoară câte adnotări au fost identificare din numărul total de adnotări.

F-Measure se calculează cu ajutorul celorlalte două măsurători, rezultatul fiind o medie ponderată a acestora.

Pentru a determina performanţa sistemului trebuie să comparăm setul de adnotări *Default* cu cel realizat prin adnotare manuală, numit *AdnotareManuala*. Metoda prin care am realizat acest lucru este utilizând interfaţa grafică oferită de instrumentul *Annotation Diff Tool*, care poate să compare adnotările unui singur document deodată. Am setat *Key doc* şi *Resp doc* la documentul pe care dorim să îl analizăm. De asemenea, am setat *Key set* şi *Resp. set* la seturile de adnotări pe care dorim să le comparăm şi anume *AdnotareManuala*, respectiv *Default set*. Tipul este setat pentru adnotarea *Claim*, iar *Features* este setat pe *none*. Se apasă pe butonul *Compare* pentru ca sistemul să calculeze rezultatele.

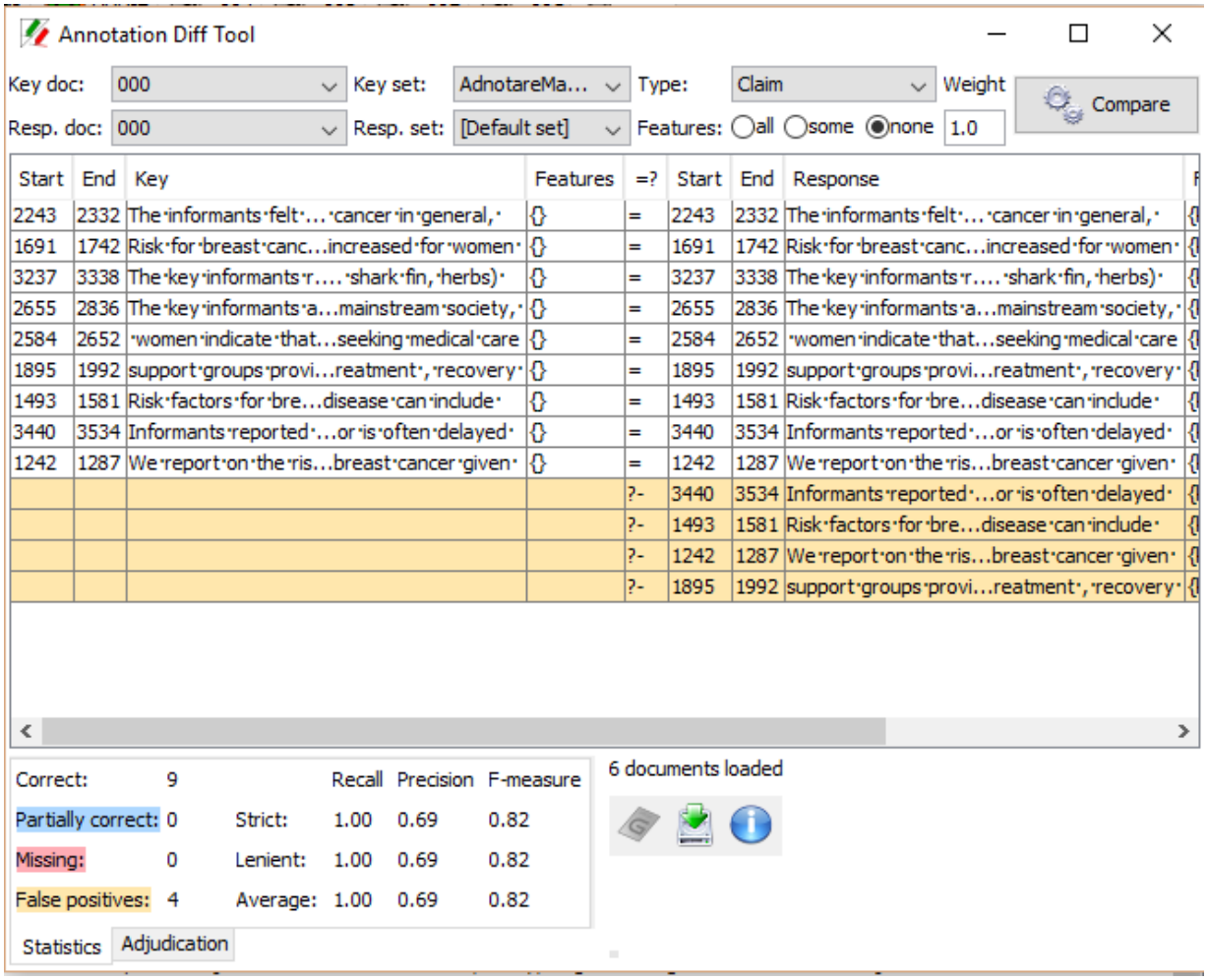

Figura 6.1: Testarea sistemului utilizând Annotation Diff Tool

Tabelul 6.1: Identificarea concluziei (stânga) şi a premiselor (dreapta) unui argument în cadrul aceleaşi propoziţii.

#	*Recall*	*Precision*	*F*	#	*Recall*	*Precision*	*F*
1	1.00	0.62	0.77	1	1.00	0.67	0.80
2	0.96	0.64	0.77	2	1.00	0.72	0.84
3	0.88	0.96	0.92	3	0.92	0.92	0.92
4	1.00	1.00	1.00	4	0.67	1.00	0.80
5	1.00	0.69	0.82	5	1.00	0.69	0.82
6	1.00	0.67	0.80	6	0.88	0.58	0.70

Tabelul 6.2: Identificarea concluziei (stânga) şi a premiselor (dreapta) unui argument aflate în propoziţii separate.

#	*Recall*	*Precision*	*F*	#	*Recall*	*Precision*	*F*
1	1.00	0.92	0.96	1	0.85	0.94	0.89
2	0.92	0.92	0.92	2	0.83	0.88	0.86
3	0.93	0.88	0.90	3	0.93	0.97	0.95
4	0.95	0.95	0.95	4	0.91	1.00	0.95
5	1.00	1.00	1.00	5	0.6	1.00	0.75
6	0.95	0.95	0.95	6	1.00	1.00	1.00

6.2 Rezultatele obţinute

Pentru a evalua performanţa sistemului am adnotat manual şase documente ce conţin fragmente din articole care au ca principal subiect cancerul la sân. În total am detectat 135 de argumente. Apoi am comparat aceste adnotări cu cele detectate de sistem, rezultatele atât pentru concluzie, cât şi pentru premisă fiind vizibile în tabelele 6.1, 6.2.

Rezultate au fost obţinute pe un calculator care prezintă următoarea configuraţie: Intel i7-3610QM CPU cu o frecvenţă de 2.30GHz şi cu o memorie RAM instalată de 6.00 GB.

Aceste documente conţin în total 203 de propoziţii extrase din trei articole care au ca şi principal subiect cancerul la sân. Dintr-un total de 135 de adnotări, 127 au fost detectate automat, corect de către algoritmul descris în cadrul tezei.

Primul document conţine 16 argumente ce conţin conjuncţie coordonatoare în interior şi 11 fără. Dintre acestea toate cele 16 au fost identificate corect, respectiv 11 concluzii şi 9 premise dintre cele identificate ca şi o propoziţie. Al doilea document conţine în total 31 de argumente, 29 dintre aceste fiind detectate automat. În cel de-al treilea document s-au detectat 30 dintre cele 32 de argumente. În al patrulea text s-au detectat toate cele 17 argumente. În al cincilea document s-au identificat nouă dintre cele nouă concluzii şi premise separate de conjuncţie coordonatoare şi cinci, respectiv trei concluzii şi premise

dintre cele cinci argumente. În ultimul document s-au detectat toate cele 14 argumente.

Identificarea incorectă este prezentă atunci când concluzia şi premisa se regăseşte în propoziţii separate. Identificarea incorectă este ilustrată în exemplele 14, 15.

Exemplul 14 (Exemplu de identificare incorectă) *Propoziţia "When the family history is known, risk profiles can be refined." este adnotată ca şi Claim începând de la cuvântul "the", nu de la primul cuvânt "When".*

Exemplul 15 (Exemplu de identificare incorectă) *Propoziţiile "Additionally, many women reflected some dissatisfaction with the quality of the relationship with their doctors." şi "Almost all the women reported doctor recommendations of healthier diet, exercise to aid recovery." sunt adnotate ca şi Claim începând de la cuvântul "women", nu de la primul cuvânt al propoziţiei.*

Rezultale obţinute de sistem sunt influenţate de performanţa identificării corecte a diferitelor părţi de vorbire. Argumentele sunt identificate de sistem cu ajutorul listelor de cuvinte şi a părţilor de vorbire obţinute cu plugin-ul *MiniPar Parser* integrat în framework-ul GATE. Dacă adnotaţiile nu sunt identificate corect acestea vor limita performanţa sistemului discutat în cadrul tezei.

6.3 Testarea componentei de interogare

Pentru testarea funcţionalităţii corecte a interogărilor din cadrul aplicaţiei am rulat sistemul şi am urmat paşii necesari pentru obţinerea argumentelor şi salvarea acestora în cadrul ontologiei de argumentare. Pentru realizarea de interogări pe ABox-ul ontologiei se apasă butoanele afişate în cadrul ferestrei principale a interfeţei grafice.

Fiecare buton va returna un set de date regăsit în cadrul ontologiei despre domeniul argumentării. Dacă acţionăm pe unul dintre ele putem observa rezultatul interogării în fereastra albă destinată afişării de rezultate.

Testarea presupune interogări multiple pe mai multe documente text pentru a putea verifica corectitudinea în mai multe cazuri, nu doar într-un caz particular.

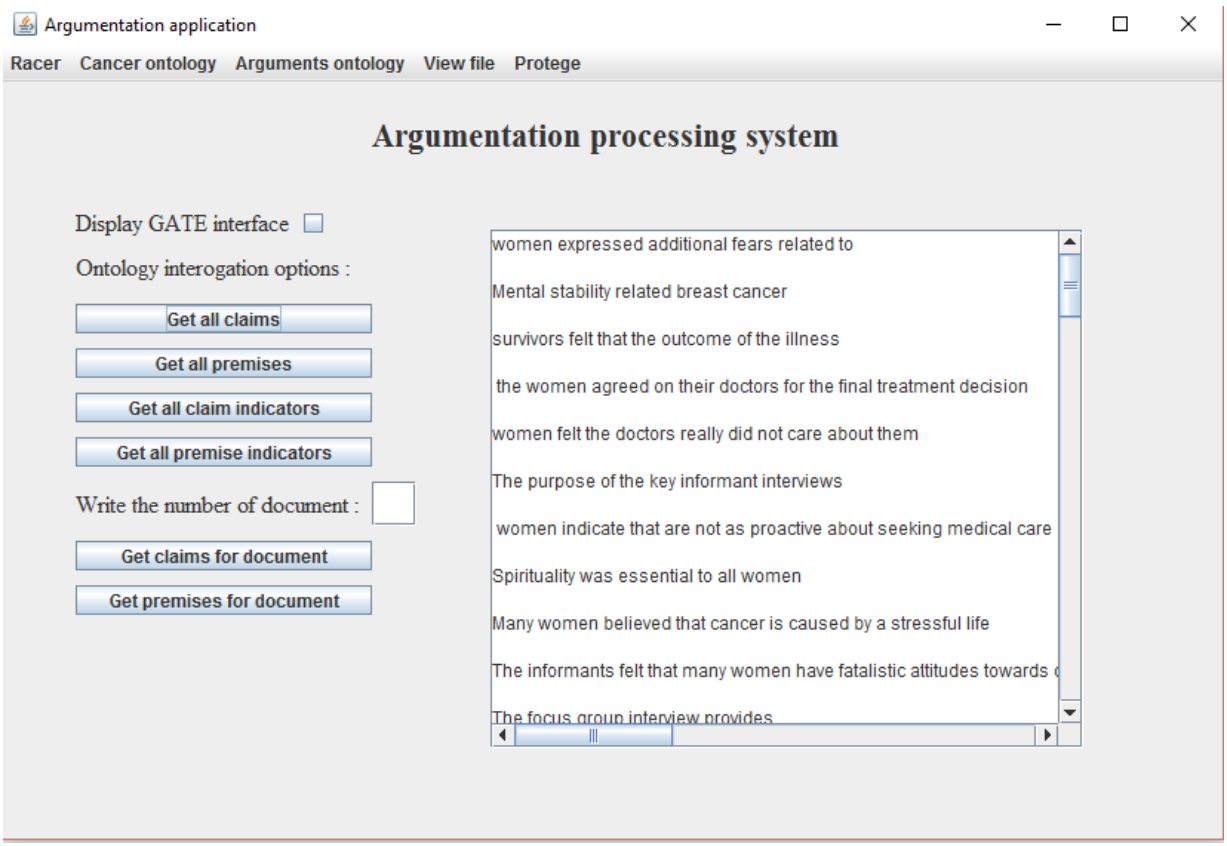

Figura 6.2: Interfaţa sistemului de detectare automată a argumentelor

Capitolul 7

Manual de Instalare şi Utilizare

În secţiunea următoare sunt detaliate resursele hardware şi software necesare pentru instalarea şi rularea aplicaţiei, precum şi o descriere pas cu pas a procedeului de instalare. Tot în acest capitol este descris modul în care se utilizează aplicaţia din punctul de vedere al utilizatorului, fără menţionarea aspectelor tehnice interne.

7.1 Manual de instalare

Pentru a utiliza acest instrument, utilizatorul trebuie să aibă instalat pe calculator următoarele programe:

- Eclipse IDE
- RacerPro Reasoner, bază de cunoştinţe pe 32 biţi
- Protege 5.0.0 pentru Windows
- GATE Developer 8.0 pentru Windows

Pentru instalarea sistemului trebuie efectuaţi următorii paşi:

1. Se verifică dacă calculatorul pe care se va rula aplicaţia are instalat JDK 7 sau o versiune mai nouă, iar dacă aceasta nu există se va instala.
2. Se verifică dacă pe maşină este instalat framework-ul GATE. Dacă nu există atunci acesta este descărcat de la adresa "https://gate.ac.uk/".
3. Se adaugă variabila de mediu GATE_HOME, setându-se locaţia spre folderul unde este instalat programul GATE.
4. Se crează un folder nou unde se copiază codul sursă al sistemului.

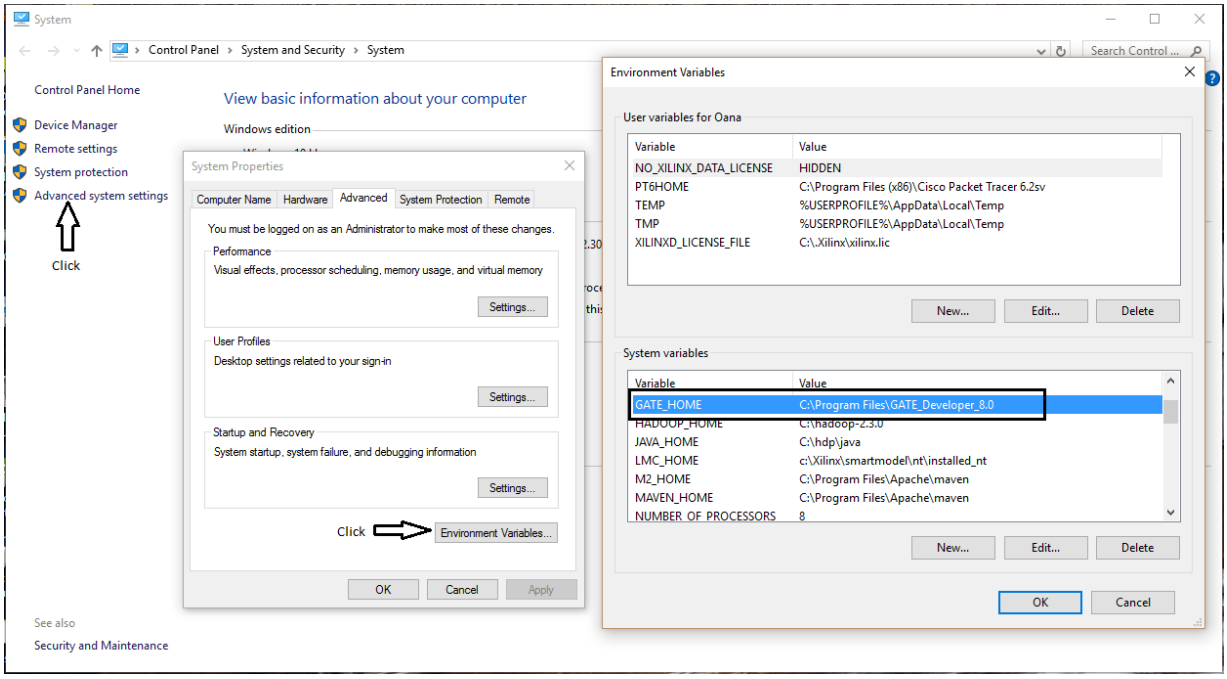

Figura 7.1: Adăugarea variabilei de mediu GATE_HOME

5. Se verifică dacă pe maşină este instalată aplicaţia RacerPro. Dacă nu există, atunci aceasta este descărcată de la adresa ”http://www.ifis.uni-luebeck.de/ moeller/racer/download.html”.

6. Se verifică dacă pe maşină este instalată aplicaţia Protege. Dacă nu există, atunci aceasta este descărcată de la adresa ”http://protege.stanford.edu/products.php#desktop-protege”.

7. Se descarcă un Eclipse IDE, pentru implementare s-a folosit Eclipse Mars.

8. Se importă proiectul în Eclipse utilizând opţiunea *File -> Import*.

9. În fereastra de *Package-Explorer* se apasă click dreapta pe numele proiectului selectându-se apoi opţiunea *Build Path -> Configure Build Path*.

10. Dacă se observă în fereastra din figura 7.3 intrări eronate, datorită căii conţinute invalide acestea sunt şterse cu ajutorul opţiunii *Remove*.

11. În fereastra din figura 7.3 se selectează opţiunea *Add External JARSs...* şi se adaugă librăriile aflate în folderul *lib* al proiectului.

12. Se selectează opţiunea *Project -> Build Project* sau se apasă tastele *CTRL + B* pentru a se verifica dacă există erori de cale.

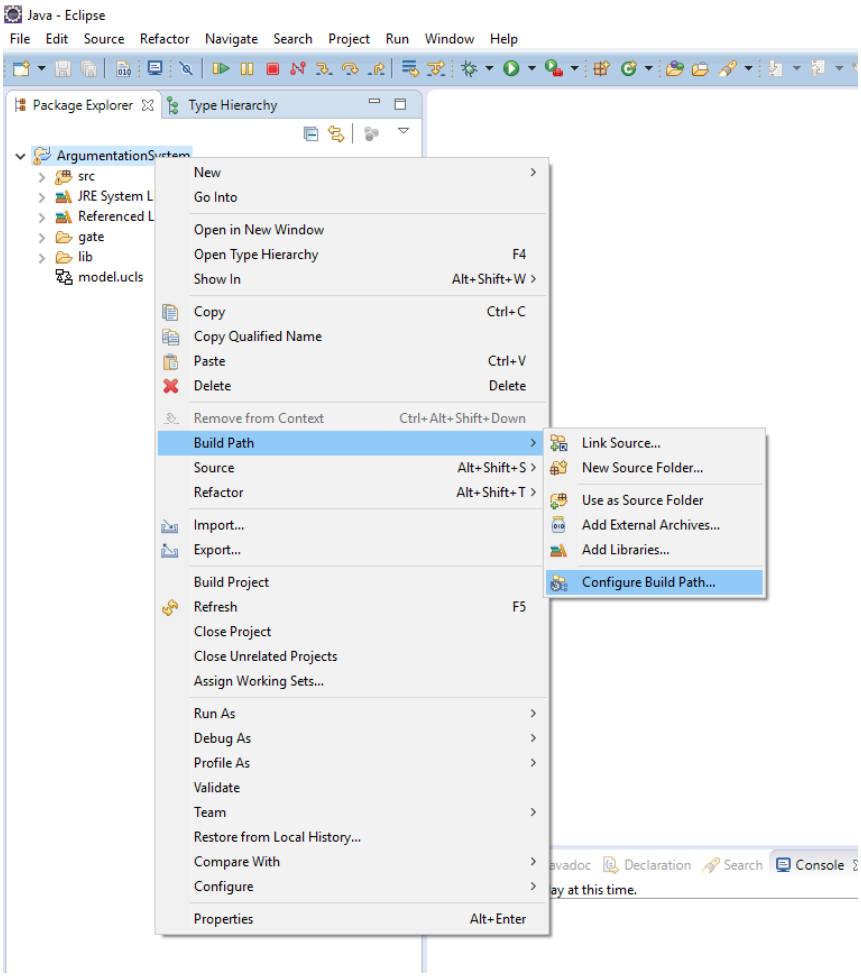

Figura 7.2: Actualizarea build path-ului

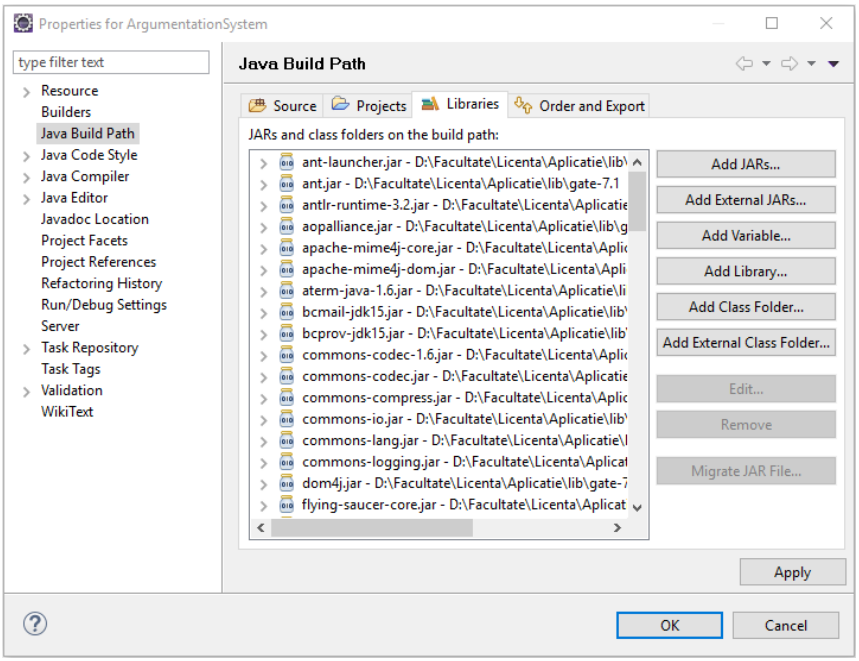

Figura 7.3: Adăugarea sau ştergerea librăriilor

7.2 Manual de utilizare

Sistemul pe care l-am implementat prezintă trei interfeţe grafice, una dezvoltată de mine, iar celelate sunt oferite de aplicaţiile GATE şi Protege. Atunci când aplicaţia este rulată apare interfaţa dezvoltată în Java SWING, iar dacă sunt setate opţiunile pentru afişarea celorlate două acestea au să fie afişate.

7.2.1 Rularea sistemului

Pentru rularea aplicaţiei pentru prima dată utilizatorul trebuie să realizeze următorii paşi:

1. Se deschide proiectul cu ajutorul programului Eclipse IDE.

2. Se vor modifica căile din cadrul codului spre aplicaţiile GATE şi Protege spre locaţia unde se regăsesc acestea în cadrul sistemului curent.

3. Se copiază conţinutul fişierelor intitulate *gazetteer* şi *NE* aflate în locaţia *Aplicatie -> gate -> grammar* la destinaţia unde este instalat programul GATE, în cazul de faţă *C: -> Program Files -> GATE_Developer_8.0 -> plugins -> ANNIE -> resources -> gazetteer*, respectiv *C: -> Program Files -> GATE_Developer_8.0 -> plugins -> ANNIE -> resources -> NE*.

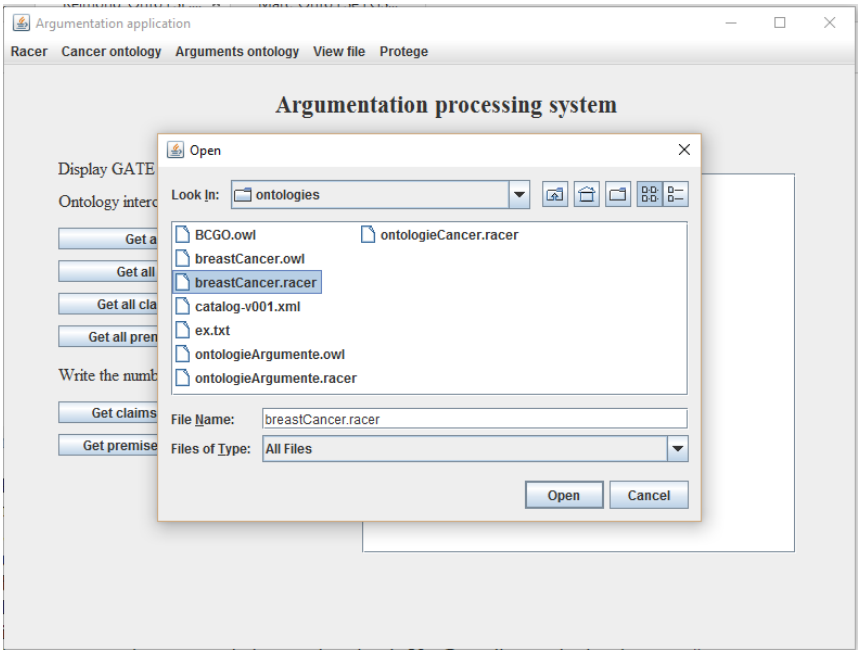

Figura 7.4: Adăugarea ontologiei de cancer

4. Se va rula aplicaţia utilizând tastele *CRTL + F5* sau selectându-se proiectul din fereastra *Package-Explorer* se alege opţiunea *Run* din meniul *Run*.

5. În urma rulării, interfaţa grafică va apărea pe ecran.

7.2.2 Identificarea argumentelor

Pentru identificarea argumentelor din cadrul documentelor text se vor realiza următorii paşi:

1. Se selectează din meniul *Racer* opţiunea *Start RacerPorter*, astfel rezultând deschiderea aplicaţiei RacerPro pentru încărcarea ontologiilor.

2. Se selectează din meniul *Cancer ontology* opţiunea *Load ontology*. Astfel, pe ecran apare un *File Chooser* cu ajutorul căruia se navighează spre locaţia ontologiei de cancer, aceasta fiind *Aplicatie -> gate -> ontologies -> breastCancer.racer*, după cum se poate observa şi în Figura 7.4.

3. Se apasă pe butonul *Open*, în urma căruia va fi vizibil un mesaj ca în Figura 7.5.

4. Se selectează din meniul *Cancer ontology* opţiunea *Create lists related to cancer*. În urma acestei operaţii un mesaj de succes va fi vizibil pe ecran, se apasă butonul *OK*.

5. Se selectează din meniul *Cancer ontology* opţiunea *Close connection ontology*. În urma acestei operaţii un mesaj de succes va fi vizibil pe ecran, se apasă butonul *OK*.

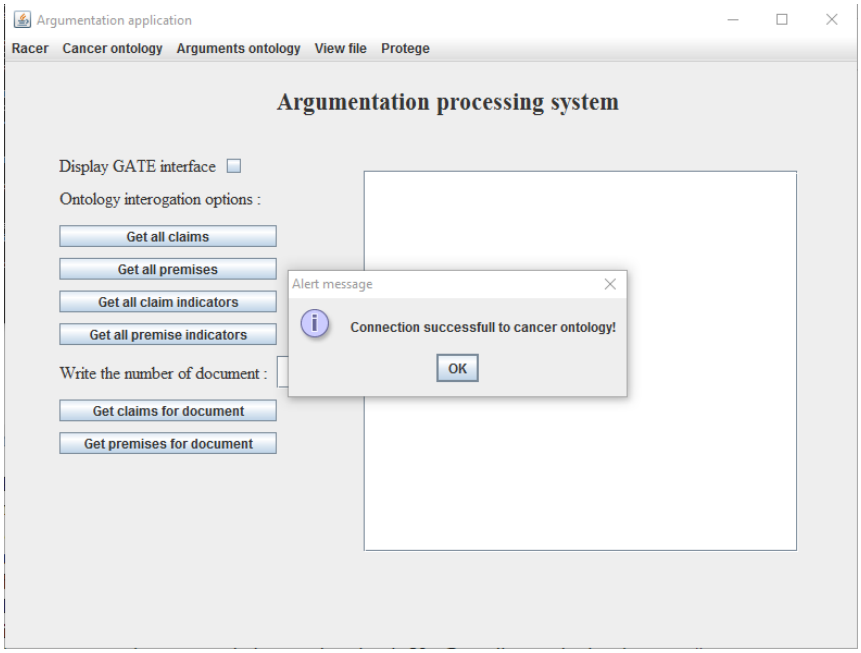

Figura 7.5: Încărcarea cu succes a ontologiei de cancer

6. Se selecteză din meniul *Arguments ontology* opţiunea *Load ontology*. Astfel, pe ecran apare un *File Chooser* cu ajutorul cărui se navighează spre locaţia ontologiei de argumentare, aceasta fiind *Aplicatie -> gate -> ontologies -> ontologieArgumente.racer*

7. Se apasă pe butonul *Open*, în urma căruia va fi vizibil un mesaj ca în Figura 7.6.

8. Se selecteză din meniul *Arguments ontology* opţiunea *Create lists related to arguments*. În urma acestei operaţii un mesaj de succes va fi vizibil pe ecran, se apasă butonul *OK*.

9. Se selecteză din meniul *Arguments ontology* opţiunea *Set corpus*. În urma acestei operaţii un mesaj de succes va fi vizibil pe ecran, se apasă butonul *OK*.

10. Se selecteză din meniul *Arguments ontology* opţiunea *Identify claim*. În urma acestei operaţii un mesaj de succes va fi vizibil pe ecran, se apasă butonul *OK*.

11. Se selecteză din meniul *Arguments ontology* opţiunea *Indentify premise*. În urma acestei operaţii un mesaj de succes va fi vizibil pe ecran, se apasă butonul *OK*.

12. Se selecteză din meniul *Arguments ontology* opţiunea *Save ontology*. În urma acestei operaţii un mesaj de succes va fi vizibil pe ecran, se apasă butonul *OK*.

13. Se selecteză din meniul *Arguments ontology* opţiunea *Close connection ontology*. În urma acestei operaţii un mesaj de succes va fi vizibil pe ecran, se apasă butonul *OK*.

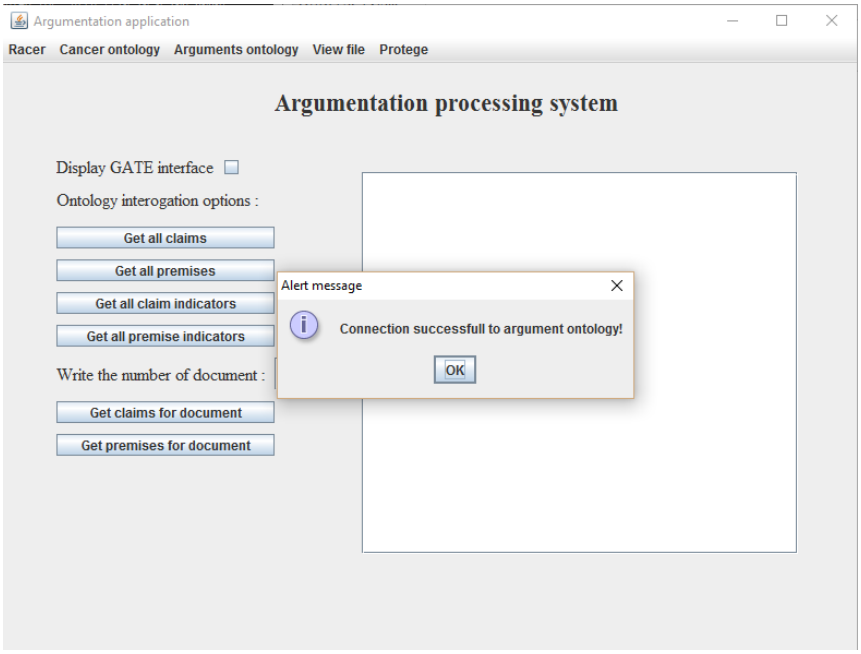

Figura 7.6: Încărcarea cu succes a ontologiei de argumentare

7.2.3 Interogarea ontologiei de argumente

În urma detecţiei de la secţiunea anterioară a argumentelor şi a salvării acestora, se pot realiza interogări asupra ontologiei obţinute urmând următorii paşi:

1. Se selectează din cadrul interfeţei principale un buton aferent interogării dorite. De exemplu, pentru afişarea tuturor concluziilor se apasă butonul *Get all claims*, rezultatul fiind afişat în spaţiul liber, aflat în dreapta butoanelor, ca în Figura 7.7.

2. Pentru detectarea concluziilor sau premiselor din cadrul unui anumit document se tastează în căsuţa din dreptul textului *Write the number of the document :* un număr între 0 şi 5. Dacă orice alt caracter, simbol sau număr este introdus, un mesaj de eroare va fi vizibil ca în Figura 7.8.

7.2.4 Vizualizarea adnotărilor prin intermediul interfeţei GATE

Pentru a observa adnotăriile din documentele text prin intermediul interfeţei frameworkului GATE se urmează paşii:

1. Se selectează checkbox-ul din dreptul textului *Display GATE interface* aflat pe fereastra principală a sistemului rulat.

2. După apariţia interfeţei aplicaţiei GATE trebuie selectat un document din meniul aflat în partea stângă numit *Language Resources*.

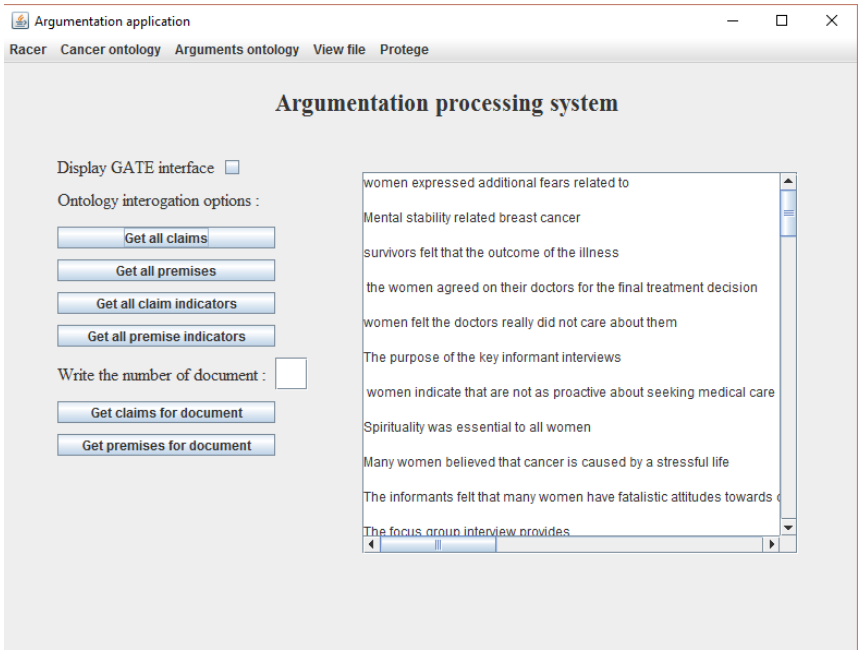

Figura 7.7: Afişarea tuturor concluziilor din cadrul argumentelor detectate

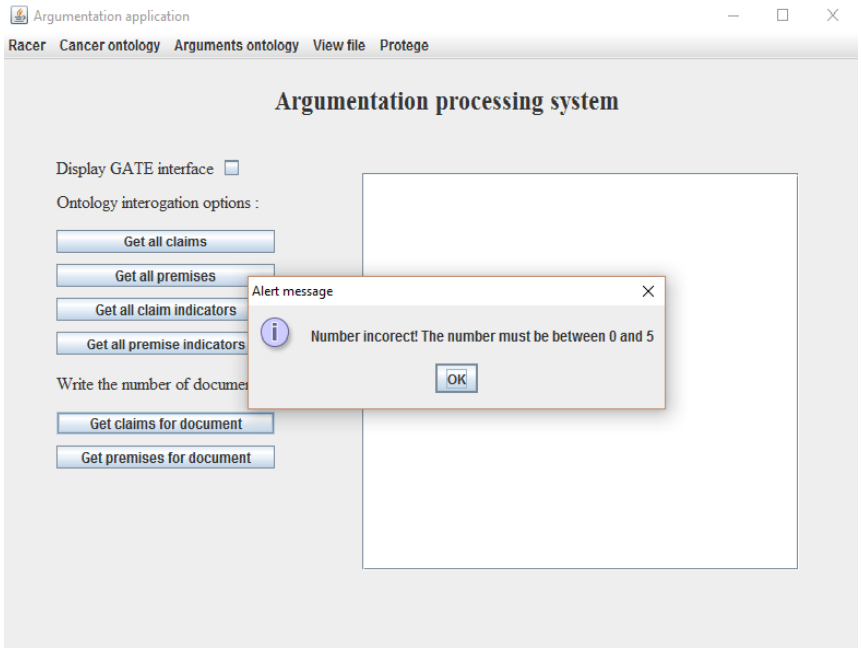

Figura 7.8: Afişarea mesajului de eroare la introducerea unui caracter greşit

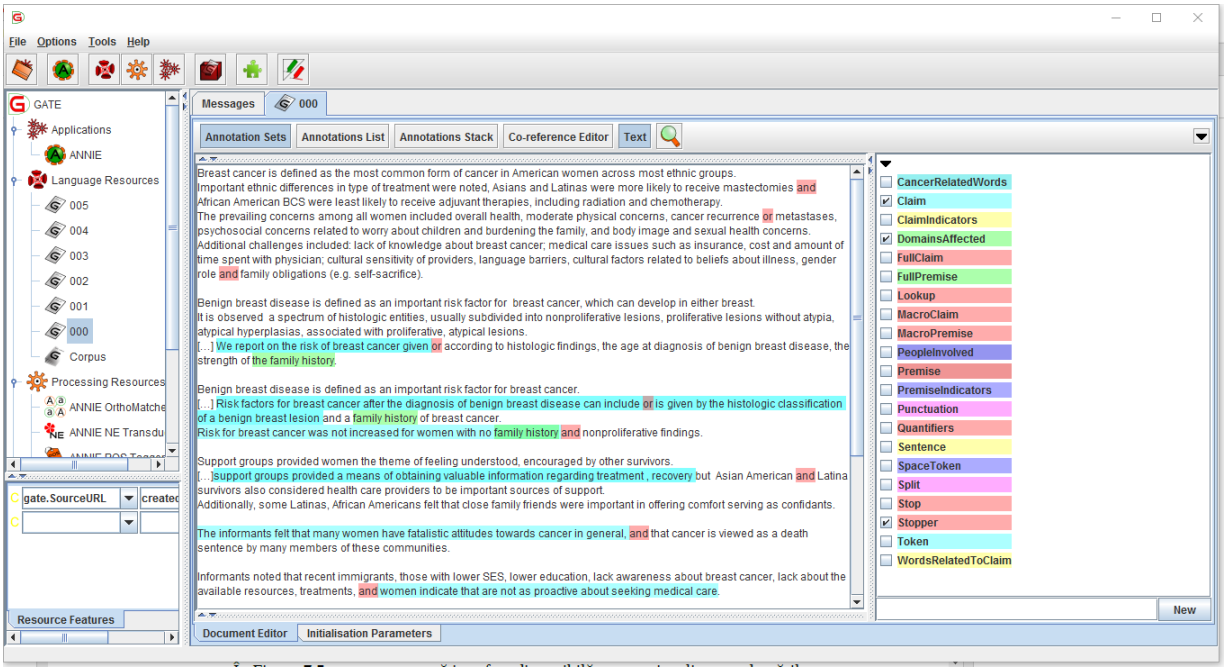

Figura 7.9: Vizualizarea adnotărilor

3. Se apasă pe butonul *Annotation Sets* şi se va selecta adnotarea dorită spre vizualizare, după cum se observă şi în Figura 7.9.

7.2.5 Vizualizarea listelor de cuvinte .lst

Pentru a vizualiza listele de cuvinte care ajută la identificarea cuvintelor se urmează paşii:

1. Se selectează din meniul *View file* opţiunea *Open file*. Astfel, pe ecran o să apară un *File Chooser* cu ajutorul căruia se navighează spre locaţia listelor, aceasta fiind în cazul de faţă *C: -> Program Files -> GATE_Developer_8.0 -> plugins -> ANNIE -> resourcer -> gazetteer*.

2. Se selectează o listă dintre cele prezente şi se apasă pe butonul *Open*.

3. În urma acestor paşi pe ecran se va afişa lista, ca în Figura 7.10.

4. Pentru ştergerea căsuţei aferente afişării se alege din meniul *View file* opţiunea *Clear text field*.

7.2.6 Vizualizarea ontologiilor

Ontologiile pot să fie vizualizate prin intermediul aplicaţiei Protege sub forma unui graf; însă pentru ca acest lucru să fie posibil trebuie transformată extensia fişierului ce

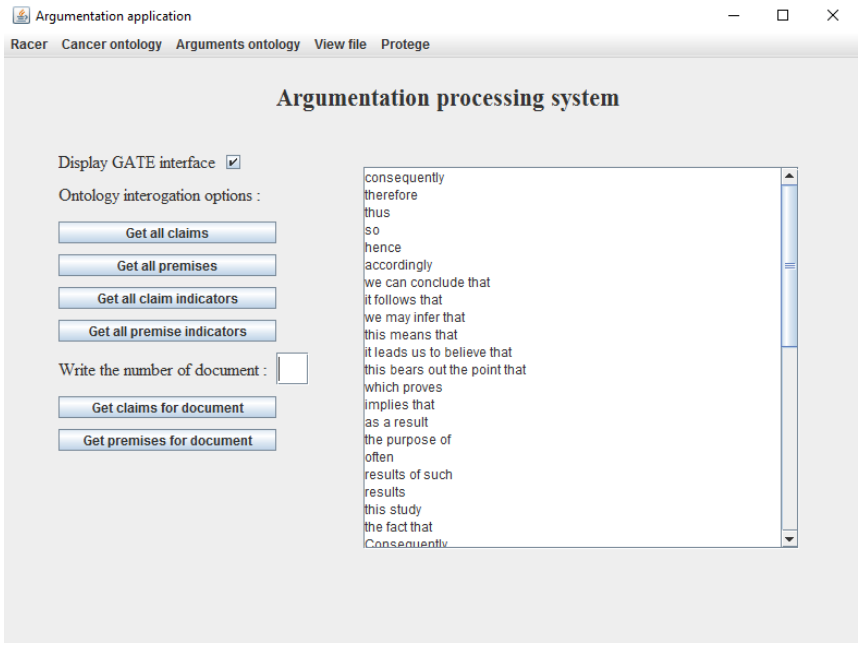

Figura 7.10: Vizualizarea listelor de cuvinte

conţine ontologia din *.racer* în *.owl*.

1. Se selectează din meniul *Protege* opţiunea *Convert cancer ontology to owl*. În urma acestei operaţii un mesaj de succes va fi vizibil pe ecran, se apasă butonul *OK*.

2. Se selectează din meniul *Protege* opţiunea *Convert arguments ontology to owl*. În urma acestei operaţii un mesaj de succes va fi vizibil pe ecran, se apasă butonul *OK*.

3. Cele două operaţii au dus la realizarea celor două ontologii necesare vizualizării.

4. Se selectează din meniul *Protege* opţiunea *Start Protege*, ducând la deschiderea aplicaţiei Protege pentru încărcarea ontologiilor.

5. Din meniul *File* se selectează *Open* şi se încarcă ontologia de argumentare în cazul de faţă de la destinaţia *Aplicatie -> gate -> ontologies -> ontologieArgumente.owl*

6. Din meniul *Window* se selectează *Views -> Ontology views -> OntoGraf import View*.

7. Prin selectarea entităţii *owl:Thing* aceasta va fi vizibilă implicit în fereastra *OntoGraf:* aferentă vizualizării.

8. Dacă se doreşte expandarea grafului se realizează dublu click asupra entităţii, un exemplu de graf fiind similar cu cel din Figura 7.11.

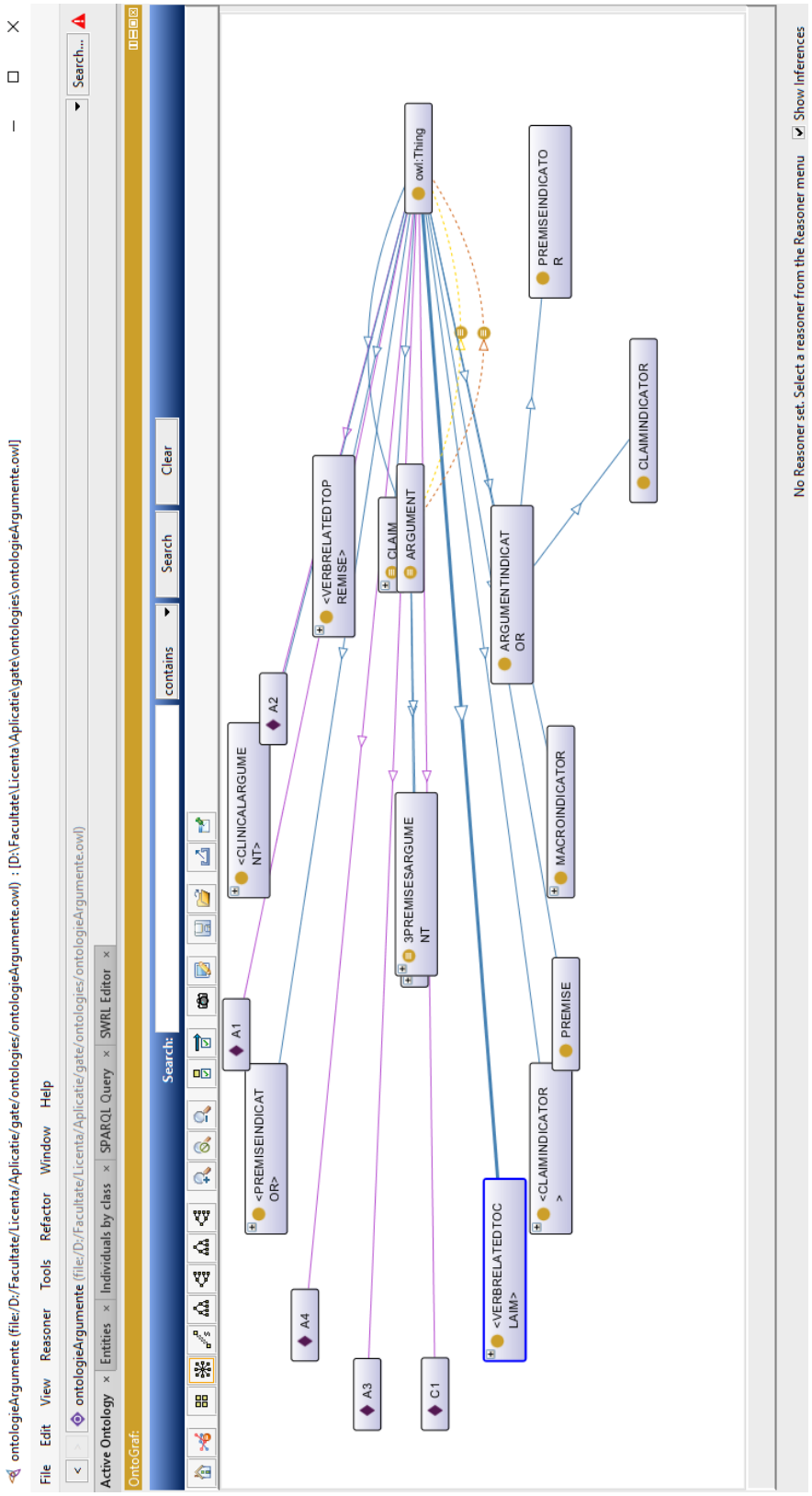

Figura 7.11: Vizualizarea ontologiei de argumentare cu ajutorul instrumentului Protege

Capitolul 8

Concluzii

8.1 Contribuţii şi realizări

Prin intermediul acestui proiect am avut posibilitatea să dezvolt un sistem de detecţie şi extragere a argumentelor din cadrul documentelor text care au ca şi subiect principal cancerul la sân. Mai precis, aplicaţia extrage prin procesarea limbajului natural structura argumentelor, şi anume concluzia şi premisa sau premisele din care este format. După această etapă de detecţie, indivizii sunt salvaţi în cadrul unei ontologii de argumentare.

Astfel contribuţia adusă de mine constă în două obiective propuse şi realizate. Primul se referă la formalizarea prin intermediul descrierii logice a unui model de argumentare. Am construit de la zero logica care modelează lumea cancerului la sân şi cea a argumentării. Bazându-ne atât pe analiza realizată asupra textelor care formează corpusul cât şi pe structura propriu zisă a unui argument am descris familia de concepte şi relaţii care există între acestea.

Ontologia de cancer este realizată după o analiză îndelungată a cuvintelor care se regăsesc în textele pe care are loc procesarea. În urma unui studiu îndelungat, am ajuns la concluzia că realizarea unei ontologii de la zero şi nu utilizarea uneia existente reprezintă o soluţie mai bună. Performanţa este îmbunătăţită, aceasta conţinând majoritatea cuvintelor regăsite în textele procesate care se referă la cancer.

De asemenea, detectarea automată a argumentelor, clasificarea acestora, dar şi serviciile de raţionament a descrierii logice pot să fie utilizate pentru modelarea, obţinerea răspunsurilor la diferite interogări realizate de utilizator asupra instanţelor recent salvate în cadrul ontologiei.

Cea de-a doua contribuţie şi cea mai importantă este detectarea automată a argumentelor. Sistemul este capabil să detecteze structura unui argument prin intermediul unor pattern-uri lingvistice. De asemenea, prezenţa conjuncţiei coordonatoare reprezintă încă o structură decisivă în ceea ce priveşte configuraţia argumentului. În timpul detecţiei, sistemul utilizează concepte şi roluri din ontologia cancerul la sân.

Am testat soluţia noastră pe un corpus ştiinţific realizat din aticole care au ca şi

punct de focus cancerul la sân. Am realizat experimente care prezintă metrica F-measure între 0.84 şi 0.97 pentru identificarea concluziilor a unui argument şi între 0.84 şi 0.93 pentru identificarea premiselor unui argument. Evaluarea realizată în timpul dezvoltării a descoperit faptul că influenţa scrierii unei propoziţii poate să îmbunătăţească, sau nu performanţa în ceea ce priveşte detecţia.

8.2 Posibilităţi de dezvoltare ulterioară

Fiind un sistem care are la bază procesarea limbajului natural şi extragerea de informaţii care urmează să fie salvate în ontologii, domenii ce sunt în continuă dezvoltare, în acest sistem pot să fie adăugate noi funcţionalităţi sau îmbunătăţite cele existente. Unele dintre optimizări sunt: introducerea altor plugin-uri, cum ar fi WordNet [23] şi SimpleNLG [24] care vor ajuta la realizarea listelor de cuvinte cât mai precise. Acestea ajută la determinarea tuturor formelor reflexive a unui verb pornind de la rădăcina lui, dar şi la găsirea sinonimelor cuvintelor detectate în cadrul textelor.

O altă îmbunătăţire este reprezentată de interfaţa grafică care este şi ea un factor de luat în calcul în cazul unei dezvoltări ulterioare. Aceasta ar putea oferi mai mult control utilizatorului în procesarea documentelor. Utilizatorul v-a putea să selecteze care document doreşte să îl proceseze, v-a putea introduce anumite interogări de mână scrise într-un limbaj special creat.

O altă posibilitate de dezvoltare este îmbunătăţirea ontologiilor, atât de argumentare cât şi de cancer la sân. TBox-ul lor ar putea fi realizat astfel încât să prezinte mai multe concepte şi relaţii care să sublinieze cât mai mult domeniile la care se fac referire. Exportarea grafică sub formă de imagine este de altfel o bună propunere spre înţelegerea bazei de date.

Bibliografie

[1] C. Tomasetti and B. Vogelstein, "Variation in cancer risk among tissues can be explained by the number of stem cell divisions,' *Science*, vol. 347, no. 6217, pp. 78–81, 2015.

[2] J. Couzin-Frankel, "The bad luck of cancer,' *Science*, vol. 347, no. 6217, pp. 12–12, 2015.

[3] R. Doll and R. Peto, "The causes of cancer: quantitative estimates of avoidable risks of cancer in the united states today,' *Journal of the National Cancer Institute*, vol. 66, no. 6, pp. 1192–1308, 1981.

[4] D. Wodarz and A. G. Zauber, "Cancer: Risk factors and random chances,' *Nature*, vol. 517, no. 7536, pp. 563–564, 2015.

[5] R. H. Johnson. and J. A. Blair, "The current state of informal logic.' in *Informal Logic 9*, 1987, pp. 147–51.

[6] ——, *Informal logic and critical thinking.*, R. G. In F. van Eemeren and F. S. H. (Eds.), Eds., 1996.

[7] ——, "The current state of informal logic.' in *Informal logic: An overview.*, 2000, pp. 93–99.

[8] R. Mochales and M.-F. Moens, "Argumentation mining,' *Artificial Intelligence and Law*, vol. 19, no. 1, pp. 1–22, 2011.

[9] A. Peldszus and M. Stede, "From argument diagrams to argumentation mining in texts: A survey,' *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, vol. 7, no. 1, pp. 1–31, 2013.

[10] T. R. Gruber, *A translation approach to portable ontology specifications.* Knowl. Acquis., June 1993, vol. 5, no. 2, pp. 199-200.

[11] P. Hitzler, M. Krőtzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, "Owl 2 web ontology language primer. w3c recommendation 27 october 2009.' 2009. [Online]. Available: http://www.w3.org/TR/owl2-primer/

[12] F. Baader, *The description logic handbook: theory, implementation, and applications*. Cambridge university press, 2003.

[13] V. Haarslev, K. Hidde, R. Möller, and M. Wessel, "The RacerPro knowledge representation and reasoning system,' *Semantic Web Journal*, vol. 3, no. 3, pp. 267–277, 2012.

[14] N. L. Green, "Identifying argumentation schemes in genetics research articles,' *NAACL HLT 2015*, p. 12, 2015.

[15] A. Wróblewska, P. Kaplanski, P. Zarzycki, and I. Lugowska, "Semantic rules representation in controlled natural language in FluentEditor,' in *Human System Interaction (HSI), 2013 The 6th International Conference on*. IEEE, 2013, pp. 90–96.

[16] V. W. Feng and G. Hirst, "Classifying arguments by scheme,' in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 987–996.

[17] F. M. Douglas Walton, Chris Reed, *Argumentation Schemes*. Cambrige University Press, 2008.

[18] J. Schneider and A. Wyner, "Identifying consumers arguments in text.' in *SWAIE*, 2012, pp. 31–42.

[19] D. Bell, D. Fried, L. Huangfu, M. Surdeanu, and S. G. Kobourov, "Towards using social media to identify individuals at risk for preventable chronic illness.' *CoRR*, vol. abs/1603.03784, 2016.

[20] A. Wyner, W. Peters, and D. Price, "Argument discovery and extraction with the argument workbench,' *NAACL HLT 2015*, p. 78, 2015.

[21] H. Cunningham, "Gate, a general architecture for text engineering,' *Computers and the Humanities*, vol. 36, no. 2, pp. 223–254, 2002.

[22] K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham, "Evolving gate to meet new challenges in language engineering,' *Natural Language Engineering*, vol. 10, no. 3-4, pp. 349–373, 2004. [Online]. Available: Evolving GATE to Meet New Challenges in Language Engineering

[23] C. Fellbaum, *WordNet*. Wiley Online Library, 1998.

[24] A. Gatt and E. Reiter, "SimpleNLG: A realisation engine for practical applications,' in *Proceedings of the 12th European Workshop on Natural Language Generation*. Association for Computational Linguistics, 2009, pp. 90–93.

Anexa A

Ontologii adnotate

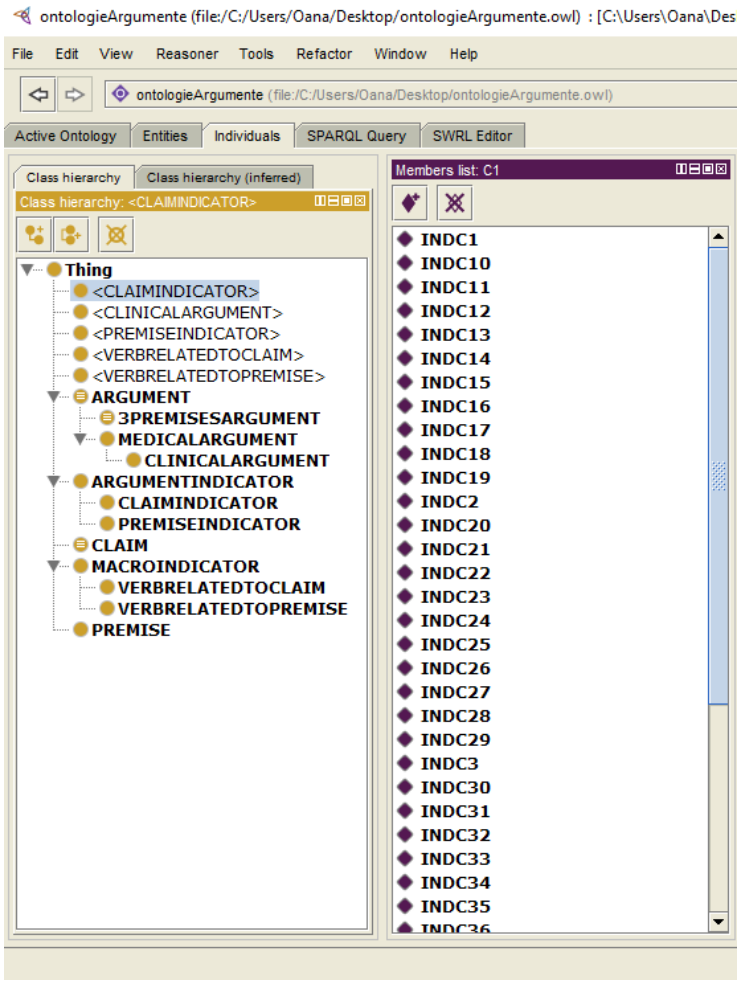

Figura A.1: Prezentarea ontologiei de argumentare cu ajutorul instrumentului Protege

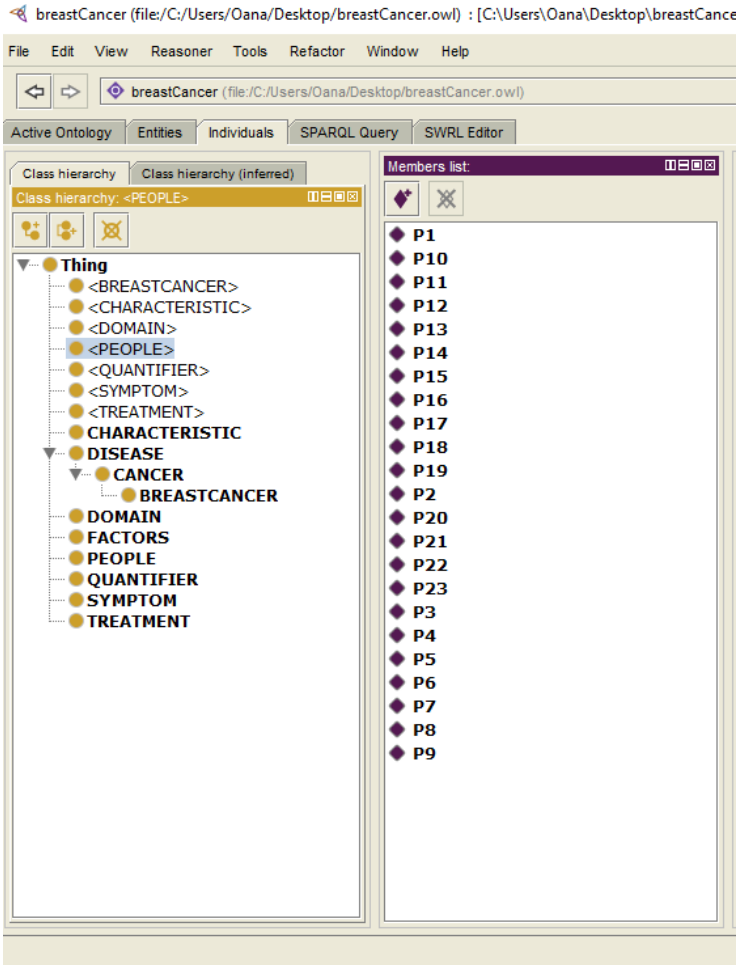

Figura A.2: Prezentarea ontologiei de cancer cu ajutorul instrumentului Protege

Anexa B

Lucrări publicate

Mining Arguments from Cancer Documents Using Natural Language Processing and Ontologies

Adrian Groza*, Popa Oana Maria*,
*Intelligent Systems Group, Department of Computer Science,
Technical University of Cluj-Napoca, Romania

***Abstract*—In the medical domain, the continuous stream of scientific research contains contradictory results supported by arguments and counter-arguments. As medical expertise occurs at different levels, part of the human agents have difficulties to face the huge amount of studies, but also to understand the reasons and pieces of evidences claimed by the proponents and the opponents of the debated topic. To better understand the supporting arguments for new findings related to current state in the medical domain we need tools able to identify arguments in scientific papers. Our work here aims to fill the above technological gap.**

Quite aware of the difficulty of this task, we embark to this road by relying on the well-known interleaving of domain knowledge with natural language processing. To formalise the existing medical knowledge, we rely on ontologies. To structure the argumentation model we use also the expressivity and reasoning capabilities of Description Logics. To perform argumentation mining we formalise various linguistic patterns in a rule-based language. We tested our solution against a corpus of scientific papers related to breast cancer. The run experiments show a F-measure between 0.84 and 0.97 for identifying conclusions of an argument and between 0.84 and 0.93 for identifying premises of an argument.

***Index Terms*—breast cancer, medical arguments, argumentation mining, natural language processing, argumentation model**

I. Introduction

Consider the recent contradictory results on cancer published in the distinguished journals Science and Nature. On the one hand we have the advocaters of the so called "bad-luck of cancer". The study in [1] supports the idea that random mutations in healthy cells may explain two-thirds of cancers. These results suggest that most cancer cases can not be prevented. One positive side of this randomness of cancer is that it helps cancer patients to know that is not their fault [2]. The intriguing correlations discovered by Tomasetti and Vogelstein contradict the older landmark paper [3] of Doll and Peto arguing that most cancers could be prevented by removing various lifestyles. On the other hand, the advocaters of risk factors of cancer provide set of counter-arguments against [1] through the voices of Wodarz and Zauber [4].

The above example is a good instantiation of the problems arising by continuous stream of scientific research that contains contradictory results supported by arguments and counter-arguments. As medical expertise occurs at different levels, part of the human agents have difficulties to face the huge amount of studies, but also to understand the reasons and pieces of evidences claimed by the proponents and the opponents of debate topic. To better understand the supporting arguments for new findings related to current state in the medical domain we need tools able to identify arguments in scientific papers. Our work here aims to fill the above technological gap.

Quite aware of the difficulty of this task, we embark to this road by relying on the well-known interleaving of domain knowledge with natural language processing. To formalise the existing medical knowledge, we rely on ontologies. To structure the argumentation model we use also the expressivity and reasoning capabilities of Description Logics. To perform argumentation mining we formalise various linguistic patterns in a rule-based language. We tested our solution against a corpus of scientific papers related to breast cancer.

In the breast cancer domain, where monthly appear more and more articles, this being a disease very spread across women in many countries. The recent proliferation of the on-line publication of medical research articles has created a critical need for information access tools that help stakeholders in the medical domain.

Because of the amount of information about a particular subject, data mining brings a set of tools and techniques that can be applied to this processed data to discover hidden patterns. This provides healthcare professionals an additional source of knowledge for making decisions. Current limitations or challenges in data mining for healthcare include information from heterogeneous sources present challenges or missing values, noise and outliers.

We proposed argumentation as the underlying technological instrumentation having the purpose of helping healthcare professionals for supporting decision making. This research focus on understanding by generating cognitive maps or argumentation graphs. Argumentation is the process where arguments are structured and evaluated based on the their interactions with each other. An argument consists of a set of premises, offered with the purpose of supporting the claim. Argumentation may also involve chains of reasoning, where claims are used as premises for deriving further claims. Argumentation mining [5], [6] is a new research area that combines Natural Language Processing (NPL) with the argumentation theories and question answering. Argumentation mining aims to automatically detect arguments from text documents, including other functionalities like the structure of an argument and the relationship between them.

The remaining of this paper is structured as follows: Section

II discusses related work and section. Section III introduces the technical instrumentation used throughout the paper. Section IV details the architecture of the system. Section V details the running experiments. VI concludes the paper.

II. Related work and discussion

There are several tools used for identifying arguments inside texts using natural language processing. Rules have been extracted from scientific papers using SWRL in Controlled English (SR-CE) in FluentEditor [7]. [8] has proposed a specification of ten causal argumentation schemes used to detect arguments for scientific claims in genetics research journal articles. The specifications and some of the examples from which they were derived were used to create an initial draft of guidelines for annotation of a corpus. Feng and Hirst [9] have investigated argumentation scheme recognition using the Araucaria corpus, which contains annotated arguments from newspaper articles, parliamentary records, magazines, and online discussion boards (Reed et al. 2010). Taking premises and conclusion as given, Feng and Hirst addressed the problem of recognizing the name of the argumentation scheme for the five most frequently occurring schemes of Walton [10] in the corpus: Argument from example, Argument from cause to effect, Practical reasoning, Argument from Consequences, and Argument from Verbal Classification.

Other applications [11] have used annotations made by hand. There is no automatic detection of annotations, discourse indicators as well as user, domain, and sentiment terminology being identify manually. The difference between our system and this tool is based on this identification. Our application uses JAPE rules implemented in GATE for the identification of the claim and premise. Other researchers [12] discuss the architecture and development of an Argument Workbench, which is a interactive, integrated, modular tool set to extract, reconstruct, and visualise arguments. The Argument Workbench supports an argument engineer to reconstruct arguments from textual sources, using information processed at one stage as input to a subsequent stage of analysis, and then building an argument graph. The tool harvest and preprocess comments; highlight argument indicators, speech act and epistemic terminology; model topics; and identify domain terminology. The argument engineer analysis the output and then the input is extracted into the DebateGraph visualisation tool.

III. Argumentation model

This section formalises in Description Logic (DL) the argumentation model. First, we introduce the basic terminology of DLs. Second, we detail the argumentation model used for the argumentation mining task.

A. Description logics

In Description Logics (DLs) concepts are built using the set of constructors formed by negation, conjunction, disjunction, value restriction, and existential restriction [13] (Table I). Here, C and D represent concepts and r is a role. The semantics is defined based on an interpretation $I = (\Delta^I, \cdot^I)$, where the domain Δ^I of I contains a non-empty set of individuals, and the interpretation function $\cdot^I$ maps each concept C to a set of individuals $C^I \in \Delta^I$ and each role r to a binary relation $r^I \in \Delta^I \times \Delta^I$. The last column of Table I shows the extension of $\cdot^I$ for non-atomic concepts.

TABLE I
Syntax and semantics of $\mathcal{ALC}$.

Constructor	*Syntax*	*Semantics*
negation	$\neg C$	$\Delta^I \setminus C^I$
conjunction	$C \sqcap D$	$C^I \cap D^I$
disjunction	$C \sqcup D$	$C^I \cup D^I$
existential restriction	$\exists r.C$	$\{x \in \Delta^I \mid \exists y : (x,y) \in r^I \wedge y \in C^I\}$
value restriction	$\forall r.C$	$\{x \in \Delta^I \mid \forall y : (x,y) \in r^I \rightarrow y \in C^I\}$
individual assertion	$a : C$	$\{a\} \in C^I$
role assertion	$(a,b) : r$	$(a,b) \in r^I$

$$\mathit{ClinicalArgument} \sqsubseteq \mathit{Argument} \quad (1)$$
$$\exists\, \mathit{supports}.\top \sqsubseteq \mathit{Argument} \quad (2)$$
$$\top \sqsubseteq \forall\, \mathit{supports}.\mathit{Argument} \quad (3)$$
$$\mathit{supports}^{+} \quad (4)$$
$$\exists\, \mathit{attacks}.\top \sqsubseteq \mathit{Argument} \quad (5)$$
$$\top \sqsubseteq \forall\, \mathit{attacks}.\mathit{Argument} \quad (6)$$

Fig. 1. TBox example in the argumentation domain.

A knowledge base (KB) is formed by a terminological box *TBox* and an assertional box *ABox*. The $TBox$ contains terminological axioms of the forms $C \equiv D$ or $C \sqsubseteq D$. The $ABox$ represents a finite set of concept assertions *a:C* or role assertions *(a,b):r*, where C is a concept, r a role, and a and b are two individuals. A concept C is satisfied if there exists an interpretation I such that $C^I \neq \varnothing$. The concept D subsumes the concept C, represented by $C \sqsubseteq D$ if $C^I \subseteq D^I$ for all interpretations I. Constraints on concepts (i.e. *disjoint*) or on roles (*domain*, *range*, *inverse* role, or *transitive* properties) can be specified in more expressive description logics[1].

The Tbox in Fig. 1 introduces the subconcept *ClinicalArgument* which is a particular type of argument. An argument can support another argument via the role *supports* that has as domain the concept *Argument* (line 2) and the same range *Argument* in line 3. Line 4 specifies that the role *supports* is transitive. The attack relationship between two arguments is modeled by the role *attacks* which has as domain and range the set of arguments (lines 5 and 6). The following Abox contains the individual a of type *ClinicalArgument* which has the premise p and the claim c. a : *ClinicalArgument*, (a,p) : *hasPremise*, (a,c) : *hasClaim*.

B. Breast cancer ontology

We are interested in breast cancer ontologies. The *Breast Cancer Grading Ontology* (BCGO) assigns a grade to a tumor

[1]We provide only some basic terminologies of description logics in this paper to make it self-contained. For a detailed explanation about families of description logics, the reader is referred to [13].

$$Cancer \sqsubseteq Disease \quad (7)$$
$$BreastCancer \sqsubseteq Cancer \quad (8)$$
$$\top \sqsubseteq \forall\, manifestedSymptom.Symptom \quad (9)$$
$$\top \sqsubseteq \forall\, appliedTreatment.Treatment \quad (10)$$
$$\top \sqsubseteq \forall\, affectedDomain.Domain \quad (11)$$
$$\top \sqsubseteq \forall\, impliedPerson.Person \quad (12)$$
$$\top \sqsubseteq \forall\, haveCharacteristic.Characteristic \quad (13)$$
$$\exists\, haveQuantifier.\top \sqsubseteq People \quad (14)$$
$$\top \sqsubseteq \forall\, haveQuantifier.Quantifier \quad (15)$$

Fig. 2. TBox example in the cancer domain.

$$a : BreastCancer \quad (16)$$
$$(a,\ \text{"Skin irritation or dimpling"}) : manifestedSymptom \quad (17)$$
$$(a, \text{"Chemotherapy"}) : appliedTreatment \quad (18)$$
$$(a,\ \text{"Family history"}) : affectedDomain \quad (19)$$
$$(a, \text{"Doctors"}) : impliedPerson \quad (20)$$
$$(a, \text{"Woman"}) : impliedPerson \quad (21)$$
$$(a, \text{"Hormone receptivity"}) : haveCharacteristic \quad (22)$$
$$(a, \text{"High levels of HER2"}) : haveCharacteristic \quad (23)$$

Fig. 3. Abox with an instance of breast cancer disease. Here a is a shortcut of "Angiosarcoma".

starting from the three criteria of the NGS, being part of the Biological Process category.

The Tbox in Fig. 2 introduces concepts like *Cancer* which is a particular type of *Disease* and *BreastCancer* is a particular type of *Cancer* (axioms 7, 8). A disease has symptoms, presented by the role *manifestedSymptom* that has as range the concept *Symptom* (axiom 9). One or more treatments can be recommended via the role *appliedTreatment* that has as range the concept *Treatment* (axiom 10). Breast cancer heavily affects all fields of the human life. This is modeled by the role *affectedDomain Domain* (axiom 11).

People are implied here, like doctors and patients, this being presented by the *impliedPerson* role, which has as range the concept *Person* (axiom 12). Breast cancer has characteristics, this being modeled by the role *haveCharacteristic* with range in the concept *Characteristic* (axiom 13). In some cases the people involved of affected by this disease are numbered and for this is used the role *haveQuantifier* with the domain *People* and the range *Quantifier* (axioms 14, 15).

The Abox in Fig. 3 contains the individual *a* of type *BreastCancer* instantiated with *"Angiosarcoma"*. This individual manifest the symptom *"Skin irritation or dimpling"* and implied the persons *"Doctors"* and *"Woman"*. The treatment applied for this instance is the individual *"Chemotherapy"*. Also, this disease affected the individual domain *"Family history"* and have as characteristics *"Hormone receptivity"* and *"High levels of HER2"*.

The system uses cancer ontology to build more specific lists of words. The terminology is input to text files such as *cancerRelatedWords.lst* for terms relating to the cancer domain and *peopleInvolved.lst* for terms that may indicate people involved or affected by this disease. The lists are used by a gazetteer that associates the terms with a majorType such as *"CancerRelatedWords"* or *"PeopleInvolved"*. JAPE rules convert these to annotations that can be visualised and queried.

For example, suppose a text has a token term *"breast cancer"* and GATE has a gazetteer list with *"breast cancer"* on it; GATE (see [14]) finds the string on the list, then annotates the token with majorType as *"CancerRelatedWords"*; we convert this into an annotation that can be visualised or searched such as *CancerRelatedWords*. A range of terms that may indicate people involved are all annotated with *"PeopleInvolved"*.

The tool can also create annotations for complex concepts out of lower level annotations. In this way, the gazetteer provides a cover concept for related terms that can be queried or used by subsequent annotation processes. The advantage of using ontologies for making the lists specified above is that they can help build more powerful and more interoperable information systems in healthcare.

C. Argumentation model

An argument $a\langle p_i, c\rangle$ contains an exactly one conclusion c and a set of supporting premises p_i. The definition in DL follows:

$$Argument \equiv \exists\ hasPremise.Premise \sqcap (=1) hasClaim.Claim \quad (24)$$

We assume that claims and premises have textual descriptors and are signaled by specific lexical indicators.

$$Claim \equiv \exists\, hasText.String \sqcap \exists\, hasIndicator.ClaimIndicator \quad (25)$$
$$Premise \equiv \exists\, hasText.String \sqcap \exists\, hasIndicator.PremiseIndicator \quad (26)$$

We rely on textual indicators classified in several concepts:

$$Indicator \equiv PremiseIndicator \sqcup ClaimIndicator \sqcup MacroIndicator \quad (27)$$

Inheritance between roles has been enacted for subroles *hasPremiseIndicator* and *hasClaimIndicator*:

$$\top \sqsubseteq \forall\, hasIndicator.Indicator \quad (28)$$
$$hasPremiseIndicator \sqsubseteq hasIndicator \quad (29)$$
$$hasClaimIndicator \sqsubseteq hasIndicator \quad (30)$$

Example 1 (Sample of claim indicators)**.** *The following lexical indicators usually signal a claim and they are instances of the concept ClaimIndicator (Ci): "consequently", "therefore", "thus", "so", "hence", "accordingly", "we can conclude that", "it follows that", "we may infer that", "this means that", "it leads us to believe that", "this bears out the point that", "which proves/implies that", "as a result"*

Example 2 (Sample of premise indicators)**.** *The following expressions usually signal a premise and they are instances of the concept PremiseIndicator (Pi): "since", "because", "for", "whereas", "in as much as", "for the reasons that", "in view of the fact", "as evidenced by", "given that", "seeing that", "as shown by", "assuming that", "in particular".*

The *MacroIndicator* is formed by several words, one of then indicating verbs related to claim or premise:

$$VerbRelatedToClaim \sqcup VerbRelatedToPremise \sqsubseteq MacroIndicator \tag{31}$$

Example 3 (Verbs related to claim)**.** *The following verbs usually signal a claim and they are instances of the concept ConclusionVerbs (Vc): "to report", "to believe", "to assesse", "to identify", "to highlight", to be essential", "to confirm", "to estimate", "to provide", "to express", "to experience", "to recall", "to accept", "to reflect", "to categorize", "to indicate", "to exemplify", "to define", "to show", "to qualify"*

Example 4 (Verbs related to premise)**.** *The following verbs usually signal a premise they are instances of the concept PremiseVerbs (Vp): "to note", "to subdivide", "to contain", "to result", "to observe", "to accord", "to regard", "to feel", "to show", "to receive", "to examine", "to report", "to transcribe", "to encompass".*

Note that the premise indicators might appear after the conclusion was stated, as example 5 illustrates.

Example 5 (Premise indicator before the premise)**.** *Consider the phrase:*

> *"[Spirituality was highlighted as a fundamental component of the healing process]$_{Claim}$. [[In particular]$_{PremiseIndicator}$, survivors noted that their faith in Gods direction over the doctors healed them.]$_{Premise}$*

The text is annotated with the claim and premise which has a premise indicator (PI), namely "In particular". The argument has its claim on the first position and one premise following the claim. The premise indicator precedes its premise. The corresponding Abox follows:

$$(a,c) : hasClaim, (a,p) : hasPremise \tag{32}$$
$$(p, "In\ particular") : hasPremiseIndicator \tag{33}$$
$$(c,p) : before, ("In\ particular", p) : before \tag{34}$$

Based on the above identified information, the system classifies argument a as ClaimPremiseArgument.

Consider the medical argument in example 6 :

Example 6 (Argument example)**.**

> *[Key informants highlighted]$_{ClaimIndicator}$ spirituality as a very important component of many women's cancer experience. These communities, particularly African American, Asian and Latina, hold firm religious and spiritual beliefs and practices. [[In particular]$_{PremiseIndicator}$, many have an unshakable belief in the power of prayer, putting more importance on spirituality, their religious beliefs than on health care providers.]$_{Premise}$*

The argument structure is formalised by:

$$a : Argument, (a,c) : hasClaim, (a,p) hasPremise \tag{35}$$
$$(c, "Key\ informants\ highlighted") : hasIndicator \tag{36}$$
$$(p, "In\ particular") : hasIndicator \tag{37}$$

There are arguments in which premise precedes the conclusion, but also arguments in which the premise appears after the claim.

$$PCArgument \equiv \exists\, hasPremise.(\exists\, before.Claim) \tag{38}$$
$$CPArgument \equiv \exists\, hasPremise.(\exists\, after.Claim) \tag{39}$$

The roles *before* and *after* are inverse and transitive roles, with both the domain and range represented by sentences:

$$before^{-} \equiv after \tag{40}$$
$$before^{+} \tag{41}$$
$$after^{+} \tag{42}$$
$$\top \sqsubseteq \forall\, before.Sentence \tag{43}$$
$$\exists\, before.\top \sqsubseteq Sentence \tag{44}$$

Instances of *PCArgument* and *CPArgument* are illustrated in examples 7 and 8.

Example 7 (PCArgument)**.** *As premise appears before the claim, the identified argument is classified as a PCArgument:*

> *"[For women with non proliferative findings, no family history, a weak family history of breast cancer]$_{Premise}$, [doctors reported no increased risk."]$_{Claim}$*

Example 8 (CPArgument)**.** *Consider the text:*

> *"[[Patients report]$_{MacroIndicator}$ on the risk of breast cancer]$_{Claim}$ [according to histologic findings, the age at diagnosis of benign breast disease, the strength of the family history."]$_{Premise}$*

This sentences presents first the Claim part introduced by the macro identifier "Patients report", followed by the Premise part. Hence, this argument is classified as a CPArgument.

IV. System architecture

The developed argumentation mining system in Fig. 4 has four components: the Gate editor, text processing component, argument identification modules and the knowledge module.

The first layer consists of the GATE Editor [14] and a query interface for the updated ontology. The second layer is composed by the text processing component performs the Natural Language Processing transformations required for extracting arguments. The argument processing modules aims to identify argumentative sentences in the text. Using the TBox and the ABox the system can save in the ontology the structure of the new arguments. The cancer ontology is used for creating the lists of words used inside the JAPE rules for identifying the argument structure. The TBox related to arguments stores the definitions of an *Argument* formed by *Claim* and one or more *Premise*. The ABox related to arguments contains the character

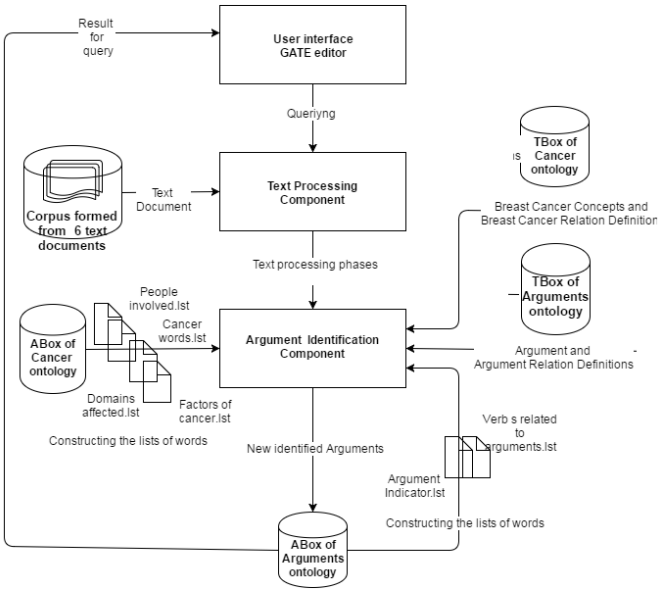

Fig. 4. Architecture of the system

instances the application found in the text document. This TBox is used to generate the lists of *ClaimIndicator*, *PremiseIndicator*, *VerbRelatedToClaim* and *VerbRelatedToPremise*.

We apply the tool to the detection of arguments from different articles within the breast cancer domain.

GATE [14] engineering tool was enacted for natural language processing and the ANNIE pipeline was exploited: tokenizer, POS tagger, gazetteer and JAPE transducer. The *Tokeniser* splits the text in simple tokens, such as numbers, punctuation and words of various kinds. The ANNIE Sentence *Splitter* segments text into sentences, while the *Tagger* produces labels corresponding for each part of speech of each word or symbol. The lookup list contained by the *Gazetteer* is used in the initial phase of the detection rules of annotations and processing resources. The framework has the ability to cascade Gate JAPE chained rules.

Racer [15] was used to perform reasoning in DL and query the system. Using Racer, the system saves the newly arguments detected in the breast cancer documents into an Abox. The resulted ontology is used for query answering.

A. Jape rules

For identifying the claim and the premises, we use JAPE (Java Annotation Patterns Engine) rules [14]. A JAPE grammar consists of a set of phases, each of which consists of a set of pattern/action rules. The left-hand-side (LHS) of the rules consists of an annotation pattern description. The right-hand-side (RHS) consists of annotation manipulation statements. Annotations matched on the LHS of a rule may be referred to on the RHS by means of labels that are attached to pattern elements.

The top level approach for detecting arguments is formalised in Algorithm 1. First, the system analysis every documents d from the corpus $\mathcal{C}$ of available medical documents (line 1).

Algorithm 1: Argumentation mining with patterns.

Input: $\mathcal{O}$ - breast cancer ontology; $\mathcal{A}(Tbox)$ - argumentation model (arg Tbox);
$\mathcal{C}$ - corpus of medical documents
Output: $\mathcal{A}$, Abox containing mined arguments

```
1  foreach d ∈ C do
2      Sentences ← Tokenise(d)
3      foreach s ∈ Sentences do
4          if ∃ CC ∈ s then
5              if ClaimMacro ∈ s then
6                  if ofstClaimMacro ≤ ofstCC then
7                      Claim ←
                       words[ofstClaimMacro, ofstCC]
8                      if PremiseMacro ∈ s then
9                          Premise ←
                           words[ofstCC, ofstPremiseMacro]
10                 else
11                     Claim ←
                       words[ofstCC, ofstClaimMacro]
12                     if PremiseMacro ∈ s then
13                         Premise ←
                           words[ofstPremiseMacro, ofstCC]
14         else
15             if ClaimMacro ∈ s then
16                 Claim ← words[ofstClaimMacro, ofstPct]
                   else
17                     if PremiseMacro ∈ s then
18                         Premise ←
                           words[ofstPremiseMacro, ofstPct]
```

Each document d is tokenised (line 3) and for each sentence s the system verifies if a coordinating conjunction (*CC*) exists in the sentence (line 4). If such *CC* is found, the algorithm searches an instance of the concept *ClaimMacro*, as a possible indicator for an argument claim (line 5). The tool looks after the conjunction and verifies if the *PremiseMacro* is present, then the *Premise* is identified (lines 8 and 9). If the offset of the *ClaimMacro* is higher than the CC offset than the system looks for the *Claim* and *Premise* inside the sentences (lines 11, 12 and 13).

If within sentences does not contain a coordinating conjunction (line 14) then an instance of *ClaimMacro* or *PremiseMacro* is searched. Depending on its presence the tool determines whether that sentence can be associated to an annotation between *Claim* or *Premise* (lines 15, 16, 17, 18, 19). If the sentence does not contain a coordinating conjunction or *ClaimMacro* or *PremiseMacro* than the system analyses the next sentence in the set of *Sentences*.

The system uses macros, indicators to decide if a sentence or a part of this sentence is a potential candidate to *Claim* or

Premise.

The following templates for claim were used: i) *Ci*: claim indicator; ii) *CibPe*: claim indicator plus people involved; iii) *CibPebVc*: claim indicators followed by people involved and specific verbs to conclusion; iv) *CibVc*: people involved and than verbs for claim; v) *ElOfCnbCw*: word or words expressing elements of cancer followed by a word that refers to cancer; after this macro can be present or not a verb that refers to claim; v) *QbPebVc*: people can have quantifiers before them representing the number of them. We rely on the textual indicators, classified as follows : $Ci \sqcup CibPe \sqcup CibPebVc \sqcup CibVc \sqcup ElOfCnbCw \sqcup CbQ \sqsubseteq ClaimIndicator$; $ClaimIndicator \sqsubseteq MacroIndicator$.

The macro *ClaimIndicator before People (MI_CibPe)* contains *ClaimIndicator* succeeded by people involved in the medical domain of breast cancer:

MI : ClaimIndicator before People ≒ *MI_CibPe*

$ci : ClaimIndicator$
$pe : Person$
$(ci, pe) : before$
$(sentence, ci) : hasToken$
$(sentence, pe) : hasToken$

Ex_1 : ["we may [infer]$_{ci}$ that [woman]$_{pe}$"]$_{CibPe}$
Ex_2 : ["this [bears out]$_{ci}$ the point that [doctors]$_{pe}$"]$_{CibPe}$
Ex_3 : ["it [follows]$_{ci}$ that [patiences]$_{pe}$"]$_{CibPe}$

The sentence Ex_1 is formalised as an instance of MI_CibPe in Example 9.

Example 9. *CibPe macro contains individuals ci of type ClaimIndicator and pe of type Person. Within sentence (lines 47, 48), ci is located before pe (line 46). The text of both individuals is presented via the role hasText (line 49).*

$$ci : ClaimIndicator, pe : Person \quad (45)$$
$$(ci, Pe) : before \quad (46)$$
$$(sentence, ci) : hasToken \quad (47)$$
$$(sentence, pe) : hasToken \quad (48)$$
$$(ci, "infer") : hasText, (pe, "woman") : hasText \quad (49)$$

The macro *ClaimIndicator before People before Verb related to Claim (MI_CibPebVc)* is a generalisation of the macro *ClaimIndicator before People (MI_CibPe)* plus verbs related to conclusion (recall Example 3):

MI : ClaimIndicator before People before Verb ≒ *MI_CibPebVc*

$ci : ClaimIndicator$
$pe : Person$
$vc : VerbRelatedToClaim$
$(ci, pe) : before$
$(pe, vc) : before$
$(sentence, ci) : hasToken$
$(sentence, pe) : hasToken$
$(sentence, vc) : hasToken$

Ex_1 : ["we can [conclude]$_{ci}$ that [doctors]$_{pe}$ [identified]$_{vc}$"]$_{CibPebVc}$
Ex_2 : ["[so]$_{ci}$ the [key informants]$_{pe}$ [provides]$_{vc}$"]$_{CibPebVc}$
Ex_3 : ["it [follows]$_{ci}$ that [people]$_{pe}$ [estimated]$_{vc}$"]$_{CibPebVc}$

The macro *ClaimIndicator before Verb related to Claim (MI_CibVc)* contains expressions that are instances of the concept *ClaimIndicator* followed by verbs related to conclusion (recall Example 3):

MI : ClaimIndicator before Verb ≒ *MI_CibVc*

$ci : ClaimIndicator$
$vc : VerbRelatedToClaim$
$(ci, vc) : before$
$(sentence, ci) : hasToken$
$(sentence, vc) : hasToken$

Ex_1 : ["[therefore]$_{ci}$ [exemplifies]$_{vc}$"]$_{CibVc}$
Ex_2 : ["[so]$_{ci}$ [highlighted]$_{vc}$"]$_{CibVc}$
Ex_3 : ["[thus]$_{ci}$ [accepted]$_{vc}$"]$_{CibVc}$

The expression Ex_1 is formalised as an instance of MI_CibVc as Example 10 illustrates.

Example 10. *CibVc macro contains the individuals ci of type ClaimIndicator and vc of type VerbRelatedToClaim. Inside the sentence (line 52), ci is located before vc (line 51). The text of both individuals is presented via the role hasText (line 53).*

$$ci : ClaimIndicator, vc : VerbRelatedToClaim \quad (50)$$
$$(ci, vc) : before \quad (51)$$
$$(sentence, ci) : hasToken, (sentence, vc) : hasToken \quad (52)$$
$$(ci, "therefore") : hasText, (vc, "exemplifies") : hasText \quad (53)$$

The macro *Elements of Cancer before Cancer related words (MI_ElOfCnbCwVc)* contains expressions that are composed by one or more words that are instances of the concept *ElementsOfCancer* plus a word from the cancer domain and optional can be succeeded by a verb related to claim (recall Example 3):

MI : Elements of Cancer before Cancer words ≒ *MI_ElOfCnbCw*

$elOfCn : ElementsOfCancer$
$cw : CancerWords$
$(elOfCn, cw) : before$
$(sentence, elOfCn) : hasToken$
$(sentence, cw) : hasToken$

Ex_1 : ["the [risk]$_{elOfCn}$ of [breast cancer]$_{cw}$"]$_{ElOfCnbCwVc}$
Ex_2 : ["these [factors]$_{elOfCn}$ of [cancer]$_{cw}$ [were equaled]$_{vc}$"]$_{ElOfCnbCwVc}$

The macro *Qualifiers before People (MI_QbPebVc)* contains qualifiers before the instances of the concept *PeopleInvolved*,

meaning that more people are involved :

MI : Qualifiers before People before Verb $\doteqdot$ *MI_QbPebVc*

q : *Qualifier*
pe : *Person*
vc : *VerbRelatedToClaim*
(q, pe) : *before*
(pe, vc) : *before*
$(sentence, q)$: *hasToken*
$(sentence, pe)$: *hasToken*
$(sentence, vc)$: *hasToken*

Ex_1 : ["[many]$_q$ [woman]$_{pe}$ [provides]$_{vc}$"]$_{QbPebVc}$
Ex_2 : ["[many]$_q$ [survivors]$_{pe}$ [accepted]$_{vc}$"]$_{QbPebVc}$

The following templates for premises were searched: i) *PibPe*: premise indicators; ii) *PibPebVp*: premise indicators plus people involved; iii) *PibPebVp*: premise indicators followed by people involved and specific verbs to premise; iv) *PibVp*: people involved and than verbs for premise v) *ElOfCnbCw*: word or words expressing elements of cancer followed by a word that refers to cancer; after this macro can be present a verb that refers to premise or not; vi) *DbVp*: words that express domains affected of breast cancer followed by verbs specific to premise; We rely on the textual indicators: $Pi \cup PibPe \sqcup PibPebVp \sqcup PibVp \sqcup ElOfCnbCW \sqcup DbVp \sqsubseteq PremiseIndicator$; $PremiseIndicator \sqsubseteq MacroIndicator$.

The macro *PremiseIndicator before People (MI_PibPe)* contains expressions that are instances of the concept that contains *PremiseIndicator* followed by people involved in the medical domain of breast cancer :

MI : PremiseIndicator before People $\doteqdot$ *MI_PibPe*

pi : *PremiseIndicator*
pe : *Person*
(pi, pe) : *before*
$(sentence, pi)$: *hasToken*
$(sentence, pe)$: *hasToken*

Ex_1 : ["[in view of the fact]$_{pi}$ that [woman]$_{pe}$"]$_{PibPe}$
Ex_2 : ["[as shown] $_{pi}$ by [doctors]$_{pe}$"]$_{PibPe}$
Ex_3 : ["[since]$_{pi}$ [patiences]$_{pe}$"]$_{PibPe}$

The macro *PremiseIndicator before People before Verb related to premise (MI_PibPebVp)* is a generalisation of the macro *PremiseIndicator before People (MI_PibPe)* plus verbs related to premise (recall Example 4):

MI : PremiseIndicator before People before Verb $\doteqdot$ *MI_PibPebVp*

pi : *PremiseIndicator*
pe : *People*
vp : *VerbRelatedToPremise*
(pi, pe) : *before*, (pe, vp) : *before*
$(sentence, pi)$: *hasToken*
$(sentence, vp)$: *hasToken*
$(sentence, pe)$: *hasToken*

Ex_1 : ["as [evidenced]$_{pi}$ by [people]$_{pe}$ [received]$_{vp}$"]$_{PibPebVp}$
Ex_2 : ["[assuming]$_{pi}$ that [doctors]$_{pe}$ [observed]$_{vp}$"]$_{PibPebVp}$
Ex_3 : ["[because]$_{pi}$ the [key informants]$_{pe}$ [were noted]$_{vp}$"]$_{PibPebVp}$

The expression Ex_1 is formalised as an instance of the MI_PibPebVp macro indicator, as Example 11 illustrates.

Example 11. *PibPebVp macro contains the individuals pi of type PremiseIndicator, pe of type People and vp of type VerbRelatedToPremise. pi is located before Pe and pe is located before vp (line 55).*

$$pi : PremiseIndicator, pe : People, vp : VerbRelatedToPremise \quad (54)$$
$$(pi, pe) : before, (pe, vp) : before \quad (55)$$
$$(sentence, pi) : hasToken, (sentence, vp) : hasToken \quad (56)$$
$$(sentence, pe) : hasToken \quad (57)$$
$$(pi, "evidenced") : hasText, (pe, "people") : hasText \quad (58)$$
$$(vp, "received") : hasText \quad (59)$$

The macro *PremiseIndicator before Verb (MI_PibVp)* contains expressions that are instances of the concept *PremiseIndicator* succeeded by verbs related to premise:

MI : PremiseIndicator before Verb $\doteqdot$ *MI_PibVp*

pi : *PremiseIndicator*
vp : *VerbRelatedToPremise*
(pi, vp) : *before*
$(sentence, pi)$: *hasToken*
$(sentence, vp)$: *hasToken*

Ex_1 : ["[since]$_{pi}$ [according]$_{vp}$"]$_{PibVp}$
Ex_2 : ["[given]$_{pi}$ that [noted]$_{vp}$"]$_{PibVp}$
Ex_3 : ["[seeing]$_{pi}$ that [served]$_{vp}$"]$_{PibVp}$

The macro *Elements of Cancer before Cancer related words (MI_ElOfCnbCw)* contains the expressions that are composed by one or more words that are instances of the concept *ElementsOfCancer* plus a word from the cancer domain and optional can be succeeded by a verb related to premise (recall Example 4):

MI : Elements of Cancer before Cancer words $\doteqdot$ *MI_ElOfCnbCw*

$elOfCn$: *ElementsOfCancer*
cw : *CancerWords*
$(elOfCn, cw)$: *before*
$(sentence, elOfCn)$: *hasToken*
$(sentence, cw)$: *hasToken*

Ex_1 : "[the [risk]$_{elOfCn}$ of [breast cancer]$_{cw}$ [was noted]$_{vp}$"]$_{ElOfCnbCw}$

The macro *Domains affected before Verb (MI_DbVp)* contains expressions that are composed by domains affected by breast cancer followed by a verb related to premise.

MI : Domains affected before Verb $\doteqdot$ *MI_DbVp*

d : *Domains*
vp : *VerbRelatedToPremise*
(d, vp) : *before*
$(sentence, d)$: *hasToken*
$(sentence, vp)$: *hasToken*

Ex_1 : ["[Family history]$_d$ [regarding]$_{vp}$"]$_{DbVp}$
Ex_2 : ["[physical changes]$_d$ [resulting]$_{vp}$"]$_{DbVp}$

The expression Ex_1 is formalised as an instance of MI_DbVp as example 12 illustrates.

Example 12. *DbVp macro contains the individuals d of type Domains and vp of type VerbRelatedToPremise. Inside the*

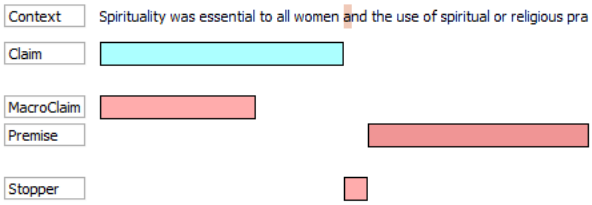

Fig. 5. Sample output from an ANNIE search

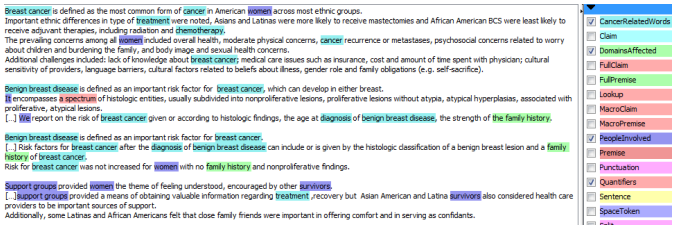

Fig. 6. An annotated text from breast cancer articles.

sentence (line 61), d is located before vp (line 60). The text of the individuals are presented via the role hasText (line 62).

$$d : Domains, vp : VerbRelatedToPremise, (d, vp) : before \quad (60)$$

$$(sentence, d) : hasToken, (sentence, vp) : hasToken \quad (61)$$

$$(d, "Familyhistory") : hasText, (vp, "regarding") : hasText \quad (62)$$

V. Running Experiments

To identify text fragments that can be used to instantiate the argumentation schemes, we use ANNIE to investigate the entire corpus. Figure 5 shows a result of a search for *Claim* and *Premise* annotation, where the "Context" represents a sentences from a document. Based on the Jape rules, the system need to know if the coordinating conjunction and the macro of claim or premise is present inside the sentence.

We can also look at annotations inside a text. Figure 6 shows one paragraph of a document, with a variety of annotation types, where are highlighted different annotation types; from this text the tool extracts word related to *Cancer*, *People involved*, *Domains affected* by breast cancer and *Qualifiers*.

The arguments were identified throw a corpus formed by six text documents related to breast cancer. In every text document are identified between five and ten arguments.

The quantitative evaluation is based on the measure of the *Precision*, *Recall* and *F-Measure* metrics. The system was evaluated on a manually annotated corpus containing six documents with different breast cancer articles. The quantitative metrics were obtained with the *Diff* plugin, integrated in GATE, applied to each document

To evaluate the performance of the system the documents were manually annotated having in total 135 annotations. This documents contain 203 sentences extracted from three articles of breast cancer. From the total of 135 arguments, 127 of them

TABLE II
Identifying claim of arguments (left) and premises (right).

#	*Recall*	*Precision*	*F*	#	*Recall*	*Precision*	*F*
1	1.00	0.77	0.862	1	0.925	0.805	0.845
2	0.94	0.78	0.845	2	0.915	0.8	0.85
3	0.905	0.92	0.91	3	0.925	0.945	0.935
4	0.975	0.975	0.975	4	0.79	1.00	0.875
5	1.00	0.845	0.91	5	0.8	0.845	0.785
6	0.975	0.81	0.875	6	0.94	0.79	0.85

are detected automatically based on the algorithm described in this paper. The percentage obtained for *Claim* and for *Premise* identification is presented in Table II. The first document contains 16 arguments with coordinating conjunction inside the structure and 11 without. Of them have been identified all 16 arguments with coordinating conjunction and 11, respectively 9 claims and premises separately. The second document contains in total 31 arguments. The number of determinate arguments was 29. The third text has 32 arguments, been identified 30. In the fourth document there were all arguments selected (17 from 17). In the fifth document there were identified nine from nine claims and premises separated by coordinating conjunction and five, respectively three claims and premises from five arguments. And in the last text 14 from the 14 arguments.

The incorrect identification is present when the argument is formed by claims and premises made of separated sentences. As example, the sentence *"When the family history is known, risk profiles can be refined."* is annotated as claim starting from the word *"the"*, not from the first word *"When"*. Other bad examples: *"Additionally, many women reflected some dissatisfaction with the quality of the relationship with their doctors."*; *"Almost all the women reported doctor recommendations of healthier diet, exercise to aid recovery."*, the claim starting in both cases from the word *"women"*.

The results obtained by the system are influenced by the performance of the correct identification of different parts of speech. Arguments are identified by the application according to lists of words and part of speech obtained by the MiniPar Parser included in GATE. If the annotations are not correctly identified this will limit the performance of the system.

VI. Conclusion

Here we integrated ontologies and NLP for identifying arguments from breast cancer articles. The contributions of this paper are: Firstly, we formalised an argumentation model in description logics. Hence, the arguments can be automatically classified, reasoning services of DL can be used on the model and the arguments can be retrieved by querying the ontology. Secondly, we developed o tool able to perform argumentation mining. During mining, the tool uses concepts and roles within a breast cancer ontology.

REFERENCES

[1] Cristian Tomasetti and Bert Vogelstein. Variation in cancer risk among tissues can be explained by the number of stem cell divisions. *Science*, 347(6217):78–81, 2015.

[2] Jennifer Couzin-Frankel. The bad luck of cancer. *Science*, 347(6217):12–12, 2015.

[3] Richard Doll and Richard Peto. The causes of cancer: quantitative estimates of avoidable risks of cancer in the united states today. *Journal of the National Cancer Institute*, 66(6):1192–1308, 1981.

[4] Dominik Wodarz and Ann G Zauber. Cancer: Risk factors and random chances. *Nature*, 517(7536):563–564, 2015.

[5] Raquel Mochales and Marie-Francine Moens. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22, 2011.

[6] Andreas Peldszus and Manfred Stede. From argument diagrams to argumentation mining in texts: A survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31, 2013.

[7] Anna Wróblewska, Pawel Kaplanski, Paweł Zarzycki, and Iwona Lugowska. Semantic rules representation in controlled natural language in FluentEditor. In *Human System Interaction (HSI), 2013 The 6th International Conference on*, pages 90–96. IEEE, 2013.

[8] Nancy L Green. Identifying argumentation schemes in genetics research articles. *NAACL HLT 2015*, page 12, 2015.

[9] Vanessa Wei Feng and Graeme Hirst. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 987–996. Association for Computational Linguistics, 2011.

[10] Fabrizio Macagno Douglas Walton, Chris Reed. *Argumentation Schemes*. Cambrige University Press, 2008.

[11] Jodi Schneider and Adam Wyner. Identifying consumers' arguments in text. In *SWAIE*, pages 31–42, 2012.

[12] Adam Wyner, Wim Peters, and David Price. Argument discovery and extraction with the argument workbench. *NAACL HLT 2015*, page 78, 2015.

[13] Franz Baader. *The description logic handbook: theory, implementation, and applications*. Cambridge university press, 2003.

[14] Hamish Cunningham. Gate, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254, 2002.

[15] Volker Haarslev, Kay Hidde, Ralf Möller, and Michael Wessel. The RacerPro knowledge representation and reasoning system. *Semantic Web Journal*, 3(3):267–277, 2012.