ACADEMIA ROMÂNĂ
Institutul de Cercetări pentru Inteligenţă Artificială

Metode de dezambiguizare semantică automată. Aplicaţii pentru limbile engleză şi română

Radu ION

Conducător: prof. dr. **Dan TUFIŞ**,
Membru Corespondent al Academiei Române

Bucureşti, mai 2007

Rezumat

Dezambiguizarea semantică automată (DSA) reprezintă un subdomeniu al Prelucrării Automate a Limbajului Natural (PLN) şi se referă la identificarea algoritmică a înţelesului unui cuvânt într-un context dat. Problema DSA a apărut ca o necesitate imediată a cercetărilor de traducere automată care au evidenţiat faptul că înţelesurile cuvintelor nu se traduc uniform pentru că la înţelesuri diferite corespund traduceri diferite. Astfel, pentru a selecta traducerea corectă a unui cuvânt, trebuie să existe o metodă de a alege acea traducere care conservă înţelesul cuvântului.

Adnotarea cu înţelesuri a devenit utilă şi pentru alte aplicaţii ale PLN. Dintre acestea, putem menţiona aplicaţiile de înţelegere a limbajului natural: generarea automată a răspunsurilor la întrebări, sisteme de recunoaştere a comenzilor în limbaj natural, etc. sau algoritmii de transcriere a vorbirii (pentru o listă mai cuprinzătoare se poate consulta [37]).

Problema DSA este recunoscută ca fiind una IA-completă. Ea nu poate fi rezolvată fără a rezolva în prealabil celelalte probleme complexe ale Inteligenţei Artificiale (IA) printre care pe primul loc se află Reprezentarea Cunoştinţelor (RC) cu un accent special pe reprezentarea cunoştinţelor implicite (aşa numitele cunoştinţe "de bun-simţ"). De aceea metodele de DSA existente aproximează capacitatea umană de a atribui înţelesuri cuvintelor modelând algoritmi evidenţiabili experiemental prin care se presupune că fiinţele umane înţeleg limbajul natural. Cel mai important dintre aceştia este exemplificat de axioma potrivit căreia înţelesul unui cuvânt este determinat de contextul de apariţie al acestuia[1] ([126, 26]).

Determinarea contextului de apariţie a unui cuvânt şi reprezentarea lui constituie principala dificultate în proiectarea de algoritmi de DSA. Există metode care reprezintă contextul ca pe o mulţime de cuvinte care apar în vecinătatea cuvântului studiat (ţintă). Altele impun restricţii pe aceste mulţimi cum ar fi ordinea în care apar cuvintele sau gradele de relevanţă a cuvintelor din mulţime asupra înţelesului cuvântului ţintă. Pe lângă acestea, metodele de DSA pe texte paralele beneficiază de un avantaj: câmpul semantic al cuvântului ţintă se restrânge[2] prin traducerea lui într-o altă limbă.

[1] [126, pag. 117]: "43. Pentru o clasă largă de cazuri de folosire a cuvântului "semnificaţie" ... semnificaţia unui cuvânt este folosirea lui în limbaj". Aici termenul "semnificaţie" este sinonim cu "înţeles".

[2] Avem în vedere faptul că traducerea conservă înţelesul cuvântului sursă şi că, în general, la traduceri diferite, corespund înţelesuri diferite.

Lucrarea de faţă îşi propune să studieze problema DSA atât pe texte simple cât şi pe texte paralele. Din perspectiva monolingvă ne interesează modelele sintactice ale contextului iar din cea multilingvă, traducerile ca şi cuantificări ale contextului.

Ideea de reprezentare sintactică a contextului de apariţie a unui cuvânt nu este nouă în peisajul cercetărilor de DSA (vezi de exemplu [97, 53, 98, 51]). În general, modelele sintactice ale contextelor au folosit gramaticile de constituenţi pentru a evidenţia corespondenţele dintre cuvinte. Prin însăşi natura lor, gramaticile de constituenţi sunt gramatici generative care încearcă, în ultimă instanţă, să explice realizarea formelor de suprafaţă[3] a propoziţiilor limbii fără a se preocupa de corespondenţa analizei sintactice cu cea a analizei semantice[4]. În contrast, formalismul sintactic al structurilor de dependenţă din [61] este conceput ca o etapă în reprezentarea semantică a propoziţiei. Mel'čuk observă faptul că ordinea cuvintelor este un mijloc expresiv universal al oricărei limbi şi care, tocmai din acest motiv, nu poate fi inclusă într-un formalism sintactic care ar trebui să fie independent de limbă[5].

Structura sintactică de dependenţe a unei propoziţii va fi aproximată de modele de atracţie lexicală ([131]) care sunt modele statistice ale structurii de dependenţă a unei propoziţii. Această structură simplifică definiţia din [61] prin eliminarea orientării arcelor şi a identificării lor cu numele relaţiilor sintactice din limbă. Din punctul de vedere al dezambiguizării semantice automate, simplificarea nu reduce complexitatea algoritmului de DSA dar, pe de altă parte, generarea grafului bazat pe modelul de atractie lexicală are propriile avantaje care nu pot fi neglijate.

În ce priveşte DSA pe texte paralele, se va prezenta un algoritm care utilizează traducerea cuvântului ţintă ca reprezentare a contextului acestuia. Înţelesurile diferite ale unui cuvânt se traduc de regulă diferit într-o altă limbă iar acest fapt se datorează cunoştinţelor pe care traducătorul le-a înglobat în traducerea cuvântului ţintă *prin examinarea contextului acestuia*. Dacă există inventare de înţelesuri compatibile[6] pentru cele două limbi, atunci prin intersecţia mulţimilor de înţelesuri ale cuvântului ţintă şi traducerii acestuia, obţinem o mulţime de înţelesuri redusă şi comună ambelor cuvinte.

[3]Forma observabilă a propoziţiei. Gramaticile generative conţin reguli de producţie din a căror aplicare ar trebui să rezulte propoziţii gramatical corecte.

[4]Pentru care nu există încă formalizări general acceptate.

[5]Lucru care nu se întâmplă cu gramaticile generative. Pentru o corespondenţă formală, vezi [69].

[6]Prin inventare de înţelesuri compatibile pentru două limbi, înţelegem inventare de înţelesuri între care înţelesurile unuia sunt echivalate la nivel sinonimic cu înţelesurile celui de-al doilea.

Cuprins

Listă de figuri

Listă de tabele

Capitolul 1

Introducere

Dezambiguizarea Semantică Automată[1] (o vom abrevia DSA de aici înainte) reprezintă un subdomeniu al Prelucrării Automate a Limbajului Natural (PLN) care şi-a câştigat recunoaşterea încă de la începuturile cercetărilor preocupate de procesarea limbajului natural. De exemplu, în traducerea automată, pentru a reda cu o cât mai mare acurateţe traducerea unui cuvânt din limba sursă, sensul acestuia trebuia cunoscut pentru a se putea alege acea lexicalizare care îl conservă în limba ţintă[2].

Un al doilea exemplu în sprijinul utilităţii unui proces de DSA ar fi acela în care se ia în considerare selectarea documentelor în care numai un anume sens al cuvântului cheie este căutat. În prezent, motoarele de căutare existente pe Internet nu fac distincţia de sensuri[3] pentru cheile căutate şi astfel, interogarea se soldează cu afişarea documentelor care conţin cuvintele cheie fără nici o altă procesare.

În [37], Ide şi Véronis dau o serie de alte argumente practice în favoarea DSA:

- inserţia de diacritice într-un cuvânt. De exemplu cuvântul "*fata*" ar fi

[1]Acronimul din limba engleză pentru aceasta problemă este WSD însemnând "*Word Sense Disambiguation*". Dezambiguizarea semantică se referă astfel la distincţia de sens.

[2]Unele ambiguităţi de sens se păstrează cu traducerea cum este exemplul engleză-română *country-ţară* în care ambiguitatea de sens teritoriu/naţiune capătă lexicalizări identice în cele două limbi. Astfel, sistemul de traducere automată trebuie să facă de fapt numai acele distincţii relevante pentru traducere.

[3]Nu se ţine de asemnea cont de categoriile gramaticale ale cuvintelor cheie. O căutare după "*book*" va returna documente care conţin atât verbul cât şi substantivul "*book*" chiar dacă suntem interesaţi numai de documentele care se referă la cărţi. În plus, cuvintele funcţionale sunt eliminate pentru că sunt prea frecvente şi deci fără relevanţă pentru căutare. Totuşi, pentru o cerere de tipul "*books about Peter Pan*", o simplă preprocesare la nivel morfosintactic a frazei de interogare ne-ar putea indica faptul că "*books*" este substantiv.

putut proveni din "*fată*", persoană tânără de sex feminin, "*faţă*" sau "*faţa*", chip, figură, sau din "*fată*", "*fătă*", proces prin care un mamifer dă naştere puilor săi;

- ataşarea grupurilor prepoziţionale în analiza sintactică. În

$$[[\text{John}]_{\text{NP}}\ [[\text{ate}]_{\text{V}}\ [\text{the cake}]_{\text{NP}}\ [\text{with a spoon}]_{\text{PP}}]_{\text{VP}}]_{\text{S}}$$

grupul prepoziţional "*with a spoon*" se ataşează verbului "*ate*" pentru că, de obicei, o prajitură se mănâncă cu linguriţa[4].

- transcrierea automată a vorbirii şi segmentarea cuvintelor într-o secvenţă vorbită;

- clasificarea tematică a documentelor.

În [48], Kilgarriff studiază aplicabilitatea DSA în următoarele 4 domenii ale PLN:

1. extragerea informaţiilor[5];
2. traducerea automată;
3. analiza sintactică;
4. înţelegerea limbajului natural[6].

Meritul DSA în traducerea automată este recunoscut şi aşa cum am afirmat mai sus un algoritm de DSA lucrând pentru un sistem de traducere automată, trebuie să identifice numai acele distincţii de sens care sunt relevante pentru traducere. În ce priveşte analiza sintactică, Kilgarriff observă că nu există studii care să indice clar dacă DSA ar îmbunătăţi performanţele unui analizor sintactic. Argumentul său care sugerează că DSA nu ar fi necesară analizei sintactice se bazează pe pe următorul exemplu:

(1.1) I love **baking** cakes **with friends**.

(1.2) I love baking **cakes with butter icing**.

[4]Asta nu înseamnă că analiza $[[\text{John}]_{\text{NP}}\ [[\text{ate}]_{\text{V}}\ [\text{the cake}\ [\text{with a spoon}]_{\text{PP}}]_{\text{NP}}]_{\text{VP}}]_{\text{S}}$ nu este posibilă. Este probabil mai puţin plauzibilă decât cealaltă.

[5]În engleză, "*Information Retrieval (IR)*".

[6]În engleză, "*Natural Language Understanding (NLU)*".

unde în 1.1 grupul prepoziţional "*with friends*" se ataşează verbului "*baking*" iar în 1.2 grupul prepoziţional "*with butter icing*" se ataşează substantivului "*cakes*". Motivaţia ataşamentului corect se află în informaţia lexicală de combinare disponibilă analizorului şi pentru că centrul[7] grupului nominal "*friends*" aparţine clasei semantice a oamenilor iar centrul grupului nominal "*butter icing*" aparţine clasei semantice a ingredientelor pentru prăjituri, nu există nici o ambiguitate semantică care să împiedice ataşamentul corect.

Putem răspunde acestui argument cu un alt exemplu:

(1.3) Am *cumpărat* un *plic* **pentru copii**.

Propoziţia 1.3 are două interpretări: fie plicul a fost cumpărat pentru un grup de copíi (lemă "*copil*") care probabil că au cerut acest lucru (grupul prepoziţional se ataşează la verb), fie s-a cumpărat un plic special conceput pentru cópii presupunând că există un astfel de plic (lemă "*copie*", grupul prepoziţional se ataşează la substantiv). Putem observa de asemenea că un analizor sintactic care se bazează pe cadrele de valenţă ale verbului şi nu pe informaţia de coocurenţă, are nevoie de determinarea claselor semantice ale argumentelor sale iar această problemă este echivalentă cu problema de DSA cu un inventar de sensuri în care acestea sunt grupate în astfel de clase semantice (sensul unui cuvânt devine astfel egal cu clasa semantică)[8].

În ce priveşte aplicaţiile de înţelegere a limbajului natural, Kilgarriff conchide că DSA are o aplicabilitate limitată în acest domeniu pentru că, în general, aceste aplicaţii sunt proiectate pentru domenii relativ restrânse unde ambiguitatea de sens nu există (sau nu interesează). În plus, conceptele ontologiilor implicate au corespondenţe stabilite prin metode ad-hoc cu domeniul de discurs iar DSA nu ar ajuta la stabilirea acestor corespondenţe. Totuşi se acceptă ideea evoluţiei acestor sisteme în direcţia depăşirii barierelor de domeniu, caz în care DSA devine necesară.

Dincolo de toate aplicaţiile imediate ale dezambiguizării semantice, credem că DSA este un domeniu al PLN care pune bazele celei mai importante cercetări a PLN: înţelegerea limbajului natural. Cu siguranţă că cercetările în DSA vor conduce la crearea de resurse lingvistice computaţionale foarte complexe construite special pentru a veni în ajutorul procesului. WordNet ([25]) este un prim exemplu (deşi această reţea semantică nu a fost construită special pentru DSA ea este responsabilă de apariţia unui număr foarte mare de algoritmi de DSA care îi exploatează direct structura în procesul de dezambiguizare).

[7]În engleză, "*head* (of a phrase)".

[8]Suntem de părere că cele două procese, analiza sintactică şi DSA, sunt interdependente. DSA are de beneficiat de pe urma procesului de analiză sintactică aşa cum se arată în [33, 53, 97].

1.1 O clasificare a metodelor de DSA

Rezolvările problemei de DSA au urmat căi diferite dând astfel naştere mai multor tipuri de rezolvări posibile. Distincţia cea mai importantă care s-a făcut între tipurile de rezolvări a fost cea de metodă *asistată*[9] faţă de metodă *neasistată*[10]. Metodele de DSA asistată (vezi de exemplu [33, 29, 19, 97]) folosesc în general texte în care fiecare cuvânt de interes[11] este adnotat la nivel de sens, pentru a se "antrena" în recunoaşterea sensurilor acestor cuvinte. Antrenarea presupune construcţia unui clasificator pentru fiecare cuvânt adnotat din textul de antrenament care va fi folosit apoi pentru a clasifica ocurenţele cuvântului dintr-un nou text (numit "de test") într-una din clasele "învăţate". Un impediment real în folosirea acestei metode de DSA este costul mare de timp pentru a produce texte adnotate cu sensuri. În general, într-un text oarecare nu toate sensurile unui cuvânt din inventarul de sensuri sunt reprezentate în text. De aceea, un text de antrenare care să furnizeze un număr suficient de exemple pentru fiecare sens al fiecărui cuvânt din inventarul de sensuri considerat este aproape imposibil de realizat "manual". O metodă de DSA asistată are nevoie de un astfel de text de antrenare pentru a putea fi aplicată, iar la rândul lui textul de antrenare ar putea beneficia de o metodă de DSA pentru a se uşura crearea sa. Acest fenomen este cunoscut sub denumirea de "*knowledge acquisition bottleneck*"[12] şi el a generat apariţia metodelor DSA intermediare de "*bootstrapping*". "*bootstrapping*" înseamnă în contextul DSA, adnotarea manuală a unor ocurenţe ale cuvântului ţintă în textul de antrenare şi aplicarea DSA asistate pentru restul de ocurenţe ale cuvântului. Exemple în acest sens sunt [33, 93].

Metodele de DSA neasistată sunt toate cele care nu sunt asistate (nu au nevoie de texte de antrenament). În mod tradiţional, aceste metode grupează ocurenţele cuvântului ţintă (de dezambiguizat) în clase de echivalenţă în care ocurenţele cuvântului dintr-o clasă au acelaşi sens (vezi de exemplu [93, 130, 94]). Identitatea sensurilor într-o clasă de echivalenţă este justificată de modul de construcţie al clasei şi anume, clasa conţine acele ocurenţe ale cuvântului ţintă care apar în contexte similare. Similaritatea contextelor este dependentă de metodă dar de obicei, un context este dat de o fereastră de cuvinte centrată în cuvântul ţintă[13].

[9] În engleză, "*supervised* (WSD method)".

[10] În engleză, "*unsupervised* (WSD method)".

[11] Cuvintele vizate în DSA sunt aşa-numitele cuvinte conţinut, adică substantivele, verbele, adjectivele şi adverbele.

[12] În IA "*knowledge acquisition bottleneck*" se referă la imposibilitatea de a descrie şi stoca cunoştinţe cu caracter enciclopedic.

[13] Formalizare a contextului cunoscută sub numele de "*bag of words*".

O altă clasificare a metodelor de DSA consideră sursele de informaţii folosite de algoritm. La o extremă se află algoritmul lui Lesk ([52]) care foloseşte textul de dezambiguizat (fără vreo adnotare prealabilă) şi inventarul de sensuri care este o simplă listă de definiţii pentru fiecare sens. La cealaltă extremă se află algoritmul lui Stevenson şi Wilks ([98]) care utilizează diverse adnotări ale textului de dezambiguizat şi un inventar de sensuri structurat. În general, un algoritm care foloseşte fie adnotări suplimentare ale textului de dezambiguizat cum ar fi de exemplu analiza sintactică, fie inventare de sensuri structurate (WordNet este un exemplu în acest sens) şi/sau ontologii (vezi SUMO, [75]) este un algoritm de DSA cu surse externe de informaţie (SEI)[14].

Dacă judecăm soluţiile problemei de DSA după acurateţea pe care algoritmii de DSA o obţin, putem afirma că cele mai bune metode de DSA sunt cele hibride. Metodele hibride de DSA folosesc de obicei în procesul de dezambiguizare orice sursă de informaţie la care au acces şi pe lângă acest lucru, se folosesc de procesările metodelor de DSA neasistată pentru a găsi clasele de echivalenţă ca un pas foarte util în activitatea de dezambiguizare. Hinrich Schütze afirmă în [94] că DSA este procesul de atribuire a etichetelor semantice ocurenţelor unui cuvânt ambiguu iar această problemă poate fi împărţită în două subprobleme:

- *sense discrimination*: gruparea ocurenţelor cuvântului ambiguu în clase de ocurenţe în care toate ocurenţele au acelaşi sens;
- *sense labeling*: identificarea sensurilor (etichetelor semantice) aplicabile claselor de ocurenţe.

Este un mod de a privi DSA ca pe un proces compus în care întâi se aplică o metodă de DSA neasistată pentru a se stabili clasele de echivalenţă ale sensurilor cuvântului ţintă după care intervine procedura de atribuire a unei etichete de sens unei clase de echivalenţă şi deci implicit fiecărei ocurenţe a cuvântului din clasă.

În figura 1.1 se află o reprezentare ierarhică a metodelor de DSA în viziunea autorului. Metodele care folosesc inventare de sensuri structurate sunt considerate cu surse externe de informaţie (+ SEI). Algoritmii pe care îi vom prezenta în această lucrare implementează următoarele tipuri de metode de DSA după această clasificare:

- WSDTool este o metodă de DSA multilingvă, neasistată cu surse externe de informaţie;

[14] În engleză, "*knowledge-based* WSD".

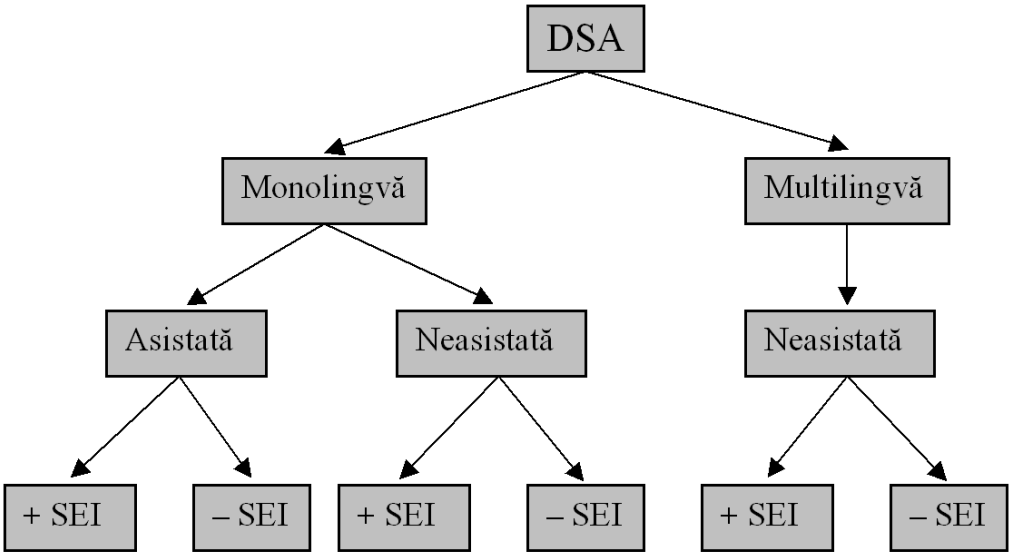

Figura 1.1: O clasificare a metodelor de DSA.

- SynWSD este o metodă de DSA monolingvă, neasistată cu surse externe de informaţie.

1.2 Despre sensuri şi înţelesuri

1.2.1 Sens şi denotaţie. Analiza limbajului

În încercarea sa de a descrie o teorie completă a înţelesului, Frege (vezi [15]) face o distincţie clară între sensul unui cuvânt şi referinţa[15] (sau denotaţia) sa. În concepţia fregeană, sensul unui cuvânt precizează referinţa cuvântului într-un context dat[16]. Astfel, două expresii pot avea aceeaşi referinţă dar două sensuri diferite:

(1.4) The Morning Star is The Evening Star.

[15]Un corespondent în logica predicatelor de ordinul I pentru referinţă ar fi extensiunea dată unui predicat de modelul de interpretare.

[16]Sau sensul este acea componentă semantică a unui cuvânt cu ajutorul căreia putem preciza referinţa cuvântului într-un context. În logica predicatelor de ordinul I, predicatele ar putea fi concepute ca sensuri.

(1.5) The Morning Star is The Morning Star.

În timp ce 1.5 este o relaţie de indentitate tautologică de tipul $a = a$[17] fără conţinut informativ, 1.4 este o relaţie care ne informează despre o identitate de tipul $a = a$ în care fiecare a a fost obţinut altfel. Această modalitate de denotare este numită de Frege "sens". Prin conectarea noţiunilor de "informaţie" (deci "cunoaştere") şi "sens", Frege îşi justifică punctul de vedere conform căruia sensul este o componentă a înţelesului unui cuvânt unde înţelesul unui cuvânt este "ceea ce cunoaşte/ştie cineva atunci când înţelege un cuvânt".

Pentru Frege, componentele înţelesului unui cuvânt sunt: tonul (sau conotaţia), forţa (cum este definită de teoria actelor de vorbire) şi sensul cuvântului. Referinţa cuvântului nu intră deloc în această descriere şi la prima vedere, ea nu joacă nici un rol în determinarea înţelesului cuvântului. Dar pentru că sensul cuvântului îi precizează referinţa şi pentru că sensul este componenta semantică a înţelesului care ajută la stabilirea valorii de adevăr a propoziţiei, concluzionăm că, deşi referinţa nu este menţionată explicit în schema de înţeles a unui cuvânt, ea este prezentă totuşi în determinarea înţelesului lui.

Frege a fost interesat de rolul semantic al expresiilor în compunerea înţelesului şi de clasificarea lor în acest scop. Despre înţelesurile expresiilor, Frege formulează două principii, principii care reflectă concepţia sa asupra analizei limbajului:

- sensul unei expresii se compune din sensurile expresiilor constituente[18];
- referinţa unei expresii se construieşte din referinţele expresiilor componente[19].

O primă distincţie care se face între expresii este aceea de expresii complete şi expresii incomplete. Expresiile complete sunt de două tipuri: nume proprii şi enunţuri. Expresiile incomplete sunt definite pe baza celor complete (de nivel 0) prin intermediul celor incomplete de nivel imediat inferior după cum urmează:

- Expresii incomplete de nivelul 1 (sunt expresii din care se elimină expresii de nivel 0):
  - Conectorii logici unari şi binari: $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$;

[17] Dacă a este un designator rigid (a este o constantă căreia i se atribuie acelaşi obiect din domeniul de discurs indiferent de modelul de interpretare ales).

[18] Expresia minimală căreia i se poate atribui sens este cuvântul.

[19] Mai exact, referinţa întregului este denotată de sensul compus.

  - Predicate unare[20] obţinute din eliminarea unui nume propriu dintr-un enunţ: din "John loves Mary" obţinem "x loves Mary" sau din "Brutus killed Caesar", "x killed Caesar";
  - Predicate n-are[21] care se obţin din eliminarea a n nume proprii dintr-un enunţ: din "Brutus killed Caesar" rezultă predicatul binar "x killed y";
  - Descripţii definite din care se elimină unul sau două nume proprii: "the father of John and Mary" generează "the father of x and y".

- Expresii incomplete de nivelul 2 (sunt expresii din care se elimină expresii de nivel 1):
  - Cuantificatorii logici: de exemplu un cuantificator împreună cu un predicat unar formează un enunţ, deci o expresie completă. Altfel spus, din $\forall x$ şi $P(x)$ obţinem $\forall x P(x)$ care are valoare de adevăr şi este deci un enunţ;
  - Operatorul de descriere: "the x [such that] $\phi(x)$" unde $\phi(x)$ este o variabilă care ia valori în mulţimea predicatelor unare.

1.2.2 DSA şi noţiunea de sens

Dacă acceptăm definiţia lui Frege a înţelesului, putem afirma că DSA se referă la selectarea algoritmică a *înţelesului* (nu a sensului) unui cuvânt în contextul său de apariţie. În lucrarea de faţă vom folosi totuşi termenii de "sens" şi "înţeles" ca fiind sinonimi[22] şi vom considera că problema de DSA este definită în raport cu un inventar de sensuri[23] care este disponibil algoritmului de dezambiguizare şi despre care se presupune că face acele distincţii de sensuri relevante pentru textul procesat[24]. Noţiunea de sens al unui cuvânt a fost folosită în această împrejurare pentru a individualiza acea componentă semantică a cuvântului de care depinde o viitoare procesare a lui sau a contextului de apariţie a lui. De exemplu, în traducerea automată interesează lexicalizările diferite ale înţelesurilor cuvântului sursă. Noţiunea

[20] Denumite şi proprietăţi.

[21] Denumite şi relaţii.

[22] Unde nu sunt, se va sublinia diferenţa dintre ei (vezi secţiunea 2.3).

[23] Un sens este privit aici ca o definiţie de dicţionar şi din acest motiv punem semnul (aproximativ) egal între "sens" şi "înţeles" (vezi secţiunea 2.3 pentru detalii asupra egalităţii dintre "sens" şi "înţeles").

[24] În această lucrare nu vom considera problemele care apar în legătură cu incompletitudinea inventarului de sensuri.

de sens al unui cuvânt a suferit astfel modificările cerute de aplicaţia de procesare a limbajului natural care foloseşte DSA.

Sensul unui cuvânt este un concept neclar din punctul de vedere al reprezentării pe care o capătă pentru fiecare vorbitor. Neclaritatea este accentuată de opiniile diferite pe care le au diversele dicţionare în legatură cu sensurile unui cuvânt dat şi chiar s-a afirmat că sensurile există în cadrul unui domeniu de aplicaţie (vezi [48]). În plus, "problema" creativităţii limbajului natural este invocată în defavoarea considerării dicţionarelor ca inventare de sensuri pentru metodele de dezambiguizare semantică automată. Un cuvânt poate fi folosit practic în orice context, cu orice categorie gramaticală, pentru a satisface nevoile de comunicare ale vorbitorului. Din acest punct de vedere, putem fi siguri că orice inventar de sensuri nu va fi niciodată suficient de bogat pentru a acoperi descriptiv întreg fondul lexical la care un om are acces. Totuşi, formularea problemei dezambiguizării semantice automate este clară: găsirea acelui sens al cuvântului cu care acesta este folosit în contextul său de apariţie, sens extras din inventarul de sensuri care este disponibil algoritmului. De aceea, din punctul de vedere al problemei, faptul că un inventar de sensuri este incomplet, nu este relevant. În cazul cuvintelor/sensurilor necunoscute, un algoritm de DSA ar trebui să indice că ele nu sunt cunoscute[25].

Dezambiguizarea semantică automată a fost considerată ca fiind o procesare utilă altora. De aceea, datele de intrare (inventarul de sensuri, modul de adnotare) au fost modificate astfel încât rezultatul dezambiguizării să fie util procesărilor ulterioare. Printre acestea, aplicaţiile de înţelegere a limbajului natural sunt cele mai în măsură să ceară serviciile oferite de un modul de DSA pentru că, în conformitate cu postulatul de compoziţionalitate a înţelesului, înţelesurile părţilor trebuie cunoscute pentru a se putea compune din ele înţelesul întregului. O întrebare naturală care se poate pune în acest punct este dacă pentru a reuşi dezambiguizarea semantică, este necesară construirea înţelesului propoziţiei pe care vrem să o dezambiguizăm. Rezolvările propuse până acum răspund negativ.

[25]Acest lucru nu se întâmplă în prezent. Lucrările despre dezambiguizarea semantică automată nu precizează care/câte cuvinte/sensuri nu au fost recunoscute pentru că nu erau prezente în lexicon. În cele mai multe cazuri, algoritmii de DSA se concentrează pe mulţimi reduse de cuvinte de dezambiguizat pentru care se testează acurateţea dezambiguizării pentru un număr determinat de sensuri care se află în lexicon.

Capitolul 2

Preprocesarea textelor. Resurse lingvistice computaţionale

Algoritmii de DSA atribuie sensuri[1] cuvintelor unui text. Pentru a realiza acest lucru, ei au nevoie să identifice în text cuvintele[2] iar în funcţie de dicţionarul folosit, au de asemenea nevoie să cunoască categoriile gramaticale[3] şi lemele[4] cuvintelor. În consecinţă, pentru a putea face DSA pe un text acesta are nevoie de câteva procesări prealabile, procesări care se fac de obicei pe niveluri (fiecare nivel depinzând de cel anterior):

1. **segmentare la nivel de frază**[5]: cei mai mulţi algoritmi de DSA folosesc contexte care nu sunt egale cu fraza. Dar pentru cei care folosesc fraza ca limită a contextului, această operaţie este necesară. Operaţia de identificare a unei fraze poate întâmpina dificultăţi atunci când aceasta conţine abrevieri de exemplu. În acest caz, punctul final al unei abrevieri poate sau nu să fie şi marcator de sfârşit de frază (vezi [31]).

2. **segmentare la nivel de cuvânt**[6]: acest proces este absolut necesar

[1] Aceste "sensuri" sunt de fapt nişte etichete care identifică înţelesuri anume ale cuvintelor aşa cum sunt ele date de un dicţionar.

[2] Identificarea unui cuvânt poate să pară o operaţie foarte simplă care nu merită menţionată dar există ambiguităţi în segmentarea la nivel de cuvânt (vezi de exemplu [31]).

[3] Este vorba despre categoriile morfosintactice (sau părţile de vorbire) cum ar fi substantiv, adjectiv, verb, adverb.

[4] Vezi nota de subsol 8 din capitolul 4.

[5] În engleză, "sentence splitting".

[6] În engleză, "tokenizing".

fiecărui algoritm de DSA. Pentru a putea atribui un sens unui cuvânt, algoritmul trebuie să obţină întâi o listă a cuvintelor de dezambiguizat.

3. **adnotare cu etichete morfosintactice**[7]: există lucrări de DSA (vezi [98]) în care se consideră că ambiguitatea de categorie gramaticală este de asemenea şi ambiguitate semantică. De aceea, performanţa adnotării cu categorii gramaticale este creditată ca performanţă a adnotării semantice în cazul în care pentru o anumită categorie gramaticală, cuvântul are un singur sens în dicţionar. Cunoaştem totuşi faptul că algoritmii de adnotare cu etichete morfosintactice ajung la ora actuală la performanţe în intervalul de precizie $96\% - 98\%$ (vezi [100, 101, 7, 88]) iar în acest caz, "dezambiguizarea" cuvintelor cu un singur sens pe categorie gramaticală nu mai reprezintă o problemă pentru că aici meritul este al algoritmului de adnotare cu etichete morfosintactice. În practicile curente de DSA, acest proces de dezambiguizare morfosintactică este considerat ca o etapă standard premergătoare dezambiguizării semantice.

4. **lematizare**: operaţie de asemenea obligatorie pentru DSA. Asigură reducerea formelor flexionare ale cuvintelor la formele standard care sunt inventariate de dicţionare. Trebuie să observăm că această operaţie este dependentă de adnotarea cu etichete morfosintactice pentru că pentru o formă flexionară a unui cuvânt lema acestuia depinde de categoria gramaticală a cuvântului. De exemplu, "haina" poate să fie adjectiv feminin, singular, articulat cu lema "hain" sau substantiv comun, feminin, singular, articulat cu lema "haină".

Pe lângă informaţia necesară pentru dezambiguizare prezentată mai sus, un algoritm de DSA mai are nevoie şi de un inventar de sensuri[8] din care să aleagă sensul unui cuvânt dintr-un context dat. Aceste inventare de sensuri fac parte din categoria resurselor lingvistice computaţionale (alături de corpusuri, lexiconuri, gramatici, ş.a.) şi sunt indispensabile dezambiguizării semantice. De ele depinde într-o oarecare măsură chiar proiectarea algoritmilor de DSA. De exemplu, în [1], reţeaua semantică Princeton WordNet 2.0 (PWN2.0) a limbii engleze ([25, 24]) este folosită pentru a calcula o densitate conceptuală între înţelesurile cuvântului ţintă şi înţelesurile cuvintelor din context de aceeaşi categorie gramaticală iar această densitate este folosită apoi pentru a selecta înţelesul cuvântului ţintă.

[7]În engleză, "part-of-speech tagging". Vezi şi nota de subsol 9 din capitolul 4.

[8]Sau dicţionar într-o accepţiune largă a termenului. Acest dicţionar trebuie să existe în format electronic astfel încât să poată fi interogat de un algoritm de DSA.

În acest capitol vom prezenta un modul de preprocesare a textelor care efectuează toate operaţiile menţionate anterior, vom continua cu o prezentare a unui corpus paralel englez-român în care partea engleză este adnotată cu etichete de sens din PWN2.0 şi care constituie un corpus de referinţă în cercetările de DSA ([67]). Odată cu traducerea în limba română a acestui corpus am reuşit să transferăm adnotările de sens din engleză în română folosind reţeaua semantică a limbii române (Romanian WordNet sau pe scurt, ROWN2.0, [105, 106]) care este aliniată (vezi secţiunea 2.3 pentru definiţia acestei operaţii) la PWN2.0.

2.1 Modulul de preprocesare a textelor TTL

TTL[9] este un modul Perl ([125]) care a fost dezvoltat din dorinţa de a dispune de un singur program care să producă nişte adnotări care altfel ar fi trebuie obţinute separat prin invocarea mai multor programe. O problemă suplimentară care apare din folosirea mai multor programe care nu sunt compatibile din punctul de vedere al formatelor datelor de intrare şi ieşire este conversia între aceste formate. De asemenea se dorea o interfaţă programabilă[10] cu acest modul, anume posibilitatea de a incorpora diverse proceduri de adnotare în alte programe.

TTL este capabil în versiunea sa curentă (6.7) să producă independent de limbă[11] următoarele adnotări:

- **recunoaşterea entităţilor denumite**[12]: în general acest proces se referă la adnotarea numelor proprii de persoane, oraşe, ţări, instituţii, etc. dintr-un text dar şi la depistarea unor entităţi cum ar fi datele, numerele (întregi, reale), ş.a.m.d. (vezi de exemplu [6] pentru o introducere). În majoritatea limbilor aceste entităţi au o grafie proprie diferită de cea a cuvintelor comune iar acest aspect este folosit în principal la identificarea lor. De exemplu, în română ca şi în engleză substantivele proprii se scriu cu majusculă. TTL foloseşte o listă de expresii regulate pentru fiecare entitate pe care o recunoaţe iar pentru fiecare expresie regulată există o etichetă care-i specifică tipul (dată, număr real, etc.). Urmează să descriem ce dificultăţi există la acest nivel şi cum procedează exact TTL pentru a identifica entităţile.

[9] În engleză, "Tokenizing, Tagging and Lemmatizing free running texts".

[10] Termen cunoscut în engleză ca API (Application Programming Interface).

[11] Cu excepţia resurselor de care are nevoie şi care sunt evident folositoare unei limbi anume.

[12] "Named Entity Recognition (NER)" în engleză.

- **segmentare la nivel de frază**: se folosesc o serie de şabloane pentru identificarea sfârşitului de frază şi de asemenea o listă de abrevieri frecvente pentru o limbă dată pentru a putea judeca natura punctului final al unei fraze.

- **segmentare la nivel de cuvânt**: a fost inspirată de segmentatorul MtSeg ([2]) şi foloseşte liste de expresii pe care le recunoaşte în text şi liste de prefixe şi sufixe care dacă fac parte dintr-un cuvânt, sunt despărţite de acesta din motive care vor deveni clare în cele ce urmează.

- **adnotare cu etichete morfosintactice**: implementează adnotatorul TnT ([7]) pe care îl îmbunătăţeşte cu câteva euristici noi.

- **lematizare**: funcţia de lematizare foloseşte un model de leme extras automat dintr-un lexicon care conţine pentru fiecare formă ocurentă a unui cuvânt, lema şi eticheta morfosintactică a ei.

2.1.1 Recunoaşterea entităţilor denumite

Această problemă a fost rezolvată prin diverse metode (pentru exemple vezi [6, 16]) însă o etapă care apare de obicei este etapa antrenării clasificatorului pe un corpus în care entităţile sunt deja recunoscute[13] (învăţare asistată[14]).

Pentru a rezolva problema entităţilor denumite, TTL apelează la expresiile regulate ca la o metodă facilă, uşor de implementat şi care nu necesită antrenare. Astfel, un expert codifică într-o gramatică nerecursivă care suportă operatorii de repetiţie Kleene $\{*, +, ?\}$[15] câte o regulă pentru fiecare tip de entitate care trebuie să fie recunoscută. Regulile gramaticii sunt apoi traduse automat prin expandarea lor[16] în expresii regulate Perl. Ordinea în care aplicarea lor este verificată este dată de un fişier de control care se numeşte filtru şi care specifică prioritatea fiecărei producţii de start din gramatică cât şi faptul dacă producţia se ia în calcul sau nu în procesul de recunoaştere. Prioritatea de aplicare este necesară pentru că o entitate poate să fie un subşir de caractere al altei entităţi iar dacă subşirul este recunoscut primul, entitatea mai cuprinzătoare rămâne astfel nerecunoscută. Din acest motiv, expresiile regulate care recunosc şiruri mai lungi de caractere primesc prioritate mai mare decât restul expresiilor regulate.

[13] Adnotate ca atare de un expert.

[14] În engleză, "*supervised training*".

[15] $a+ = aa*$, $a? = (\epsilon|a)$.

[16] Din acest motiv gramatica nu trebuie sa fie recursivă pentru că, în caz contrar, expandarea ar dura la nesfârşit.

```
LMarker -> ( '(^|\s|\(|\[|\{|\"|\'|,|\.|:|;|\?|\!)' )
RMarker -> ( '($|\s|\)|\]|\}|\"|\'|,|\.|:|;|\?|\!)' )
Abbrev  -> ( '(?:[A-Z]\.){1,4}' )

AbbrevS -> LMarker '(' Abbrev ')' RMarker
```

Figura 2.1: Gramatică pentru recunoaşterea unei abrevieri

```
apply  AbbrevS  priority 100  ctag Y  msd Yn  emsd Ed
```

Figura 2.2: Filtru pentru gramatica din figura 2.1

Un prim dezavantaj al recunoaşterii entităţilor denumite cu expresii regulate este acela că dacă există două entităţi de tipuri diferite care sunt recunoscute de o aceeaşi regulă de start a gramaticii avem un conflict de tipuri. În acest caz nu avem nicio metodă de a selecta un singur tip pentru entitatea respectivă şi din acest motiv suntem obligaţi fie să generalizăm tipul entităţii, fie să ajustăm regulile gramaticii astfel încât acest lucru să nu se întâmple.

Pentru exemplificare, fie gramatica din figura 2.1, filtrul din figura 2.2 şi fraza (formată dintr-o singură propoziţie) din 2.1.

(2.1) Serviciul Român de Informaţii (S.R.I.) este o instituţie similară cu C.I.A.

Şirurile de caractere aflate între apostrofuri reprezintă simboluri terminale ale gramaticii (sunt simboluri ale expresiilor regulate Perl) iar simbolul de start al gramaticii este `AbbrevS`. Prin expandarea producţiei `AbbrevS`, vom obţine o expresie regulată Perl care va putea fi verificată pentru aplicare pe propoziţia 2.1. Ea recunoaşte şirurile de caractere "S.R.I." şi "C.I.A." ca fiind abrevieri pentru că filtrul permite recunoaşterea abrevierilor (`apply`). Tipul entităţii este dat de mai multe etichete morfosintactice aflate în corespondenţă (vezi prezentarea adnotării cu etichete morfosintactice pentru detalii). După adnotare, propoziţia dată va conţine informaţia din figura 2.3.

2.1.2 Segmentarea la nivel de frază

Problema identificării sfârşitului unei fraze se reduce la a dezambiguiza punctuaţia finală. Dacă semnul întrebării ('?') sau semnul exclamării ('!') nu

Serviciul Român de Informaţii (
<entity nerss="AbbrevS" ctag="Y" ana="Yn" eana="Ed">S.R.I.</entity>
) este o instituţie similară cu
<entity nerss="AbbrevS" ctag="Y" ana="Yn" eana="Ed">C.I.A.</entity>

Figura 2.3: Rezultatul operaţiei de recunoaştere a entităţilor

```
î.e.n.    ABBREVIATION  înaintea_erei_noastre
ş.a.m.d.  ABBREVIATION  şi_aşa_mai_departe
ş.a.      ABBREVIATION  şi_altele
```

Figura 2.4: Câteva abrevieri uzuale în română

sunt aproape niciodată ambigue (ele termină fraze în marea majoritate a cazurilor), interpretarea punctului ('.') este ambiguă între marcajul de final de frază sau finalul unei abrevieri (sau poate primi ambele interpretări simultan, vezi de asemenea [31]). Există cazuri în care punctul apare şi în componenţa unor entităţi cum ar fi în engleză numerele reale: $1,234.543$ de exemplu.

Ca şi recunoaşterea entităţilor denumite, identificarea sfârşitului unei fraze este o problemă care a fost studiată şi pentru care există de asemenea algoritmi care învaţă din corpusuri adnotate ([89]). Abordarea noastră este iarăşi una mai simplă şi anume aceea bazată pe reguli: mai exact, aceste reguli definesc de fapt şabloanele sfârşitului de frază.

Segmentarea la nivel de frază se desfăşoară după analiza precedentă din cauză că entităţile pot conţine simboluri de sfârşit de frază. Odată cu recunoaşterea entităţilor, se elimină cazul în care se putea segmenta textul în interiorul unei entităţi. Rămâne de rezolvat problema judecării semnificaţiei punctului. TTL foloseşte o listă de abrevieri uzuale pentru fiecare limbă pentru a putea identifica abrevierile. Punctul final după un cuvânt care se află în această listă reprezină finalul unei abrevieri. Dacă după abreviere se întâlneşte un cuvânt care începe cu majusculă, punctul este de asemenea şi final de frază. Cazul rămas (punct după un cuvânt care nu este în lista de abrevieri) este considerat ca fiind sfârşit de frază.

Pentru limba română lista de abrevieri pe care o utilizează TTL conţine 731 de abrevieri (vezi figura 2.4 pentru formatul listei de abrevieri) iar pentru engleză, 186 de abrevieri (figura 2.5).

Şabloanele de sfârşit de frază rezolvă problemele care apar atunci când fraza se încheie cu punctuaţie pereche aşa cum sunt parantezele (deschise

```
m.p.h.   ABBREVIATION  miles_per_hour
vs.      ABBREVIATION  versus
i.e.     ABBREVIATION  id_est
a.m.     ABBREVIATION  ante_meridiem
p.m.     ABBREVIATION  post_meridiem
```

Figura 2.5: Câteva abrevieri uzuale în engleză

$\langle, (, \{, [$ şi închise $], \},), \rangle$), ghilimelele (deschise " şi închise ") sau apostrofurile (deschis ‘ şi închis ’). Dacă după punctuaţia de final de frază (’.’, ’?’, ’!’, ’...’, ’?...’ sau ’!...’) apare de exemplu o paranteză închisă, atunci ea trebuie păstrată în fraza curentă. În schimb, o paranteză deschisă nu trebuie păstrată dacă apare după punctuaţia de final.

2.1.3 Segmentarea la nivel de cuvânt

Se face în mod necesar după segmentarea la nivel de frază din acelaşi motive menţionate mai sus: entităţile pot conţine punctuaţie care nu trebuie segmentată iar abrevierile conţin la rândul lor punctuaţie finală care iarăşi nu trebuie separată. La acest nivel trebuie să avem deci garanţia că entităţile cât şi abrevierile sunt marcate în stilul prezentat în figura 2.3.

Dacă la segmentarea frazelor punctul nu era întotdeauna marcaj de sfârşit de frază, aici putem afirma că spaţiul nu este întodeauna marcaj de sfârşit de cuvânt. Mai mult, marcajul de sfârşit de cuvânt poate fi chiar şirul vid (ϵ) !

În orice limbă există expresii idiomatice al căror înţeles este nedecompozabil[17] şi din acest motiv ele trebuie considerate ca unităţi lexicale de sine stătătoare[18]. Chiar dacă înţelesul unei secvenţe de cuvinte poate fi construit din înţelesurile cuvintelor care o compun[19], există diverse motive pentru care putem totuşi considera secvenţa ca fiind o expresie: expresia se află într-un dicţionar, expresia este identificată ca o colocaţie[20] (pentru depistarea

[17]De exemplu “a arunca o vorbă”, “a-şi arunca ochii”, în română sau “to take a look”, “to catch one’s breath” în engleză

[18]Spaţiul nu este delimitator de cuvânt în acest caz.

[19]“Ecuaţie diferenţială” de exemplu.

[20]Manning şi Schütze, [57]: o colocaţie este o secvenţă de două sau mai multe cuvinte folosită în mod uzual pentru a exprima ceva.

mai_cu_seamă	COMPOUND
peste_poate	COMPOUND
peste_tot	COMPOUND
praf_de_puşcă	COMPOUND
punct_de_vedere	COMPOUND
punctul_de_vedere	COMPOUND

Figura 2.6: Compuşi româneşti ca unităţi lexicale

într-	LEFTSPLIT	*prepoziţie*
le-	LEFTSPLIT	*pronume*
ne-	LEFTSPLIT	*pronume*
-ţi	RIGHTSPLIT	*pronume*
-şi	RIGHTSPLIT	*pronume*
-o	RIGHTSPLIT	*pronume*

Figura 2.7: Prefixe (LEFTSPLIT) şi sufixe (RIGHTSPLIT) care trebuie separate în română.

colocaţiilor într-un corpus, vezi [57, pag. 151]). Din păcate[21], o secvenţă de cuvinte poate să constituie o expresie într-un context iar în altul nu (vezi exemplele 2.2 şi 2.3 cât şi [91]).

(2.2) Au venit *cu_miile* în piaţă.

(2.3) Negocierea se face **cu miile** de euro.

Din acest motiv, TTL foloseşte o listă de secvenţe de cuvinte[22] care indiferent de context constituie expresii şi care astfel pot fi recunoscute ca atare într-o frază[23] (vezi figura 2.6).

O altă problemă cu care se confruntă segmentarea la nivel de cuvânt este aceea că există situaţii în care dintr-o secvenţă de caractere care nu conţine spaţiu pot fi extrase două sau mai multe cuvinte[24]. Asta înseamnă că şirul vid este separator de cuvinte. Dar pentru că şirul vid apare între fiecare două

[21]Pentru prelucrarea automată a limbajului natural.

[22]Sau cuvinte compuse sau compuşi.

[23]TTL nu rezolvă deci problema dezambiguizării înţelesului expresiei în context ca unic mijloc de a identifica expresia.

[24]De exemplu, în engleză, “cannot” se separă în “can” şi “not”.

caractere consecutive ale unei secvenţe de caractere, nu putem să separăm pur şi simplu secvenţa după şirul vid. Din acest motiv, TTL păstrează o listă de prefixe şi sufixe care trebuie separate dacă sunt identificate într-un şir de caractere care nu conţine spaţiul. Această listă precizează astfel poziţiile în care şirul vid este separator de cuvânt (figura 2.7).

La acest nivel de segmentare facem uz de expresii regulate (ca şi Karttunen în [47]) pentru a separa punctuaţia de la stânga şi de la dreapta unui cuvânt. Sumar, algoritmul de segmentare la nivel de cuvânt parcurge următorii paşi pentru a obţine o listă de cuvinte dintr-o frază S (ca şir de caractare cu spaţii):

1. fiecare adnotare de tipul `<entity ...>...</entity>` devine un cuvânt în lista finală de cuvinte; se prelucrează S astfel încât segmentarea după spaţiu să nu distrugă aceste adnotări;

2. se segmentează fraza S după spaţiu şi se obţine astfel o primă listă tentativă de cuvinte L_1;

3. pentru fiecare cuvânt w_i din L_1, se elimină punctuaţia de la începutul şi de la sfârşitul lui w_i şi se construieşte astfel o nouă listă L_2 în care punctuaţia are intrări separate aflate pe poziţiile corespunzătoare (fie înaintea sau după w_i);

4. pentru fiecare cuvânt w_i din L_2 se elimină prefixe şi sufixe dacă w_i le conţine, extrase dintr-o listă ca cea din figura 2.7 construindu-se astfel o nouă listă L_3 în care prefixele şi sufixele eliminate apar pe poziţiile lor corespunzătoare (fie înaintea sau după w_i);

5. se construieşte lista finală de cuvinte L_4 în care fiecare secvenţă de N cuvinte consecutive[25] devine o singură unitate lexicală dacă secvenţa se află într-o listă similară cu cea din figura2.6.

2.1.4 Adnotarea cu etichete morfosintactice

Adnotarea cu etichete morfosintactice este o problemă a Prelucrării Automate a Limbajului Natural care s-a bucurat de o mare atenţie din partea comunităţii ştiinţifice[26]. La ora actuală există diverse metode de a rezolva această problemă printre care amintim câteva: Modelele Markov Ascunse

[25]Unde N reprezintă numărul maxim de cuvinte care pot să apară într-o expresie. Acest număr este calculat din lista de expresii.

[26]Vezi articolul despre "Part of Speech Tagging" de la adresa Internet `http://en.wikipedia.org/wiki/Part-of-speech_tagging`.

([7]), Principiul Entropiei Maxime ([88]) sau Reţelele Neurale ([92]). Nivelul de performanţă actual al algoritmilor de adnotare cu etichete morfosintactice se încadrează în intervalul $96\% - 98\%$ ceea ce înseamnă că dintr-un text oarecare primit la intrare, cel puţin 96% din unităţile lexicale care-l compun vor primi automat eticheta morfosintactică corectă în context. Cu un asemenea nivel de încredere în performanţelele algoritmului, adnotarea cu etichete morfosintactice a devenit o procesare standard în aproape orice prelucrare automată de text.

O etichetă morfosintactică[27] este o codificare a unei părţi de vorbire împreună cu valori ale variabilelor morfosintactice aplicabile ei. De exemplu, în română, substantivul are asociate următoarele variabile morfosintactice:

- **tipul**, valori: *propriu*, *comun*;
- **genul**, valori: *masculin*, *feminin*;
- **numărul**, valori: *singular*, *plural*;
- **cazul**, valori: *nominativ*, *acuzativ*, *genitiv*, *dativ*, *vocativ*;
- **articolul**, valori: *articulat*, *nearticulat*.

Dacă ar fi să construim mulţimea de etichete morfosintactice pentru substantiv în română am avea $2 \times 2 \times 2 \times 5 \times 2 = 80$ de etichete morfosintactice numai pentru substantiv.

Adnotarea cu etichete morfosintactice implică existenţa unei mulţimi de etichete morfosintactice[28] pentru o limbă dată. Acest inventar de etichete este în general proiectat astfel încât să se obţină un maxim de performanţă în adnotare relativ la cantitatea de informaţie conţinută în fiecare etichetă din inventar ([109], vezi de asemenea [110] pentru un experiment în proiectarea automată de inventare de etichete morfosintactice).

Pentru limba română cât şi pentru engleză au fost proiectate[29] câte două inventare de etichete morfosintactice aflate în corespondenţă (vezi şi tehnica adnotării stratificate, [100, 110] şi anexa A): primul inventar de etichete respectă specificaţiile MULTEXT-East ([23], similare cu exemplul pe care l-am dat mai sus) iar cel de-al doilea este derivat din primul eliminându-se din fiecare etichetă morfosintactică variabilele morfosintactice care nu sunt

[27] Vezi şi nota de subsol 9 din capitolul 4.

[28] În engleză, "tagset".

[29] În cadrul proiectului MULTEXT-East, [21].

dependente de context[30]. Astfel, pentru o etichetă morfosintactică din primul inventar avem o singură etichetă din al doilea iar unei etichete morfosintactice din al doilea inventar îi corespund una sau mai multe etichete din primul. O etichetă morfosintactică compatibilă MULTEXT-East se numeşte MSD (din engleză, "Morpho-Syntactic Descriptor") iar o etichetă derivată CTAG ("Corpus TAG").

TTL implementează adnotatorul cu etichete morfosintactice TnT ([7]) care este un adnotator probabilistic bazat pe Modele Markov Ascunse (MMA, "Hidden Markov Models" în engleză, vezi [87, 57]). Foloseşte un corpus adnotat pentru a-şi estima probabilităţile de tranziţie iar stările automatului sunt trigrame de etichete morfosintactice CTAG. Pentru a evita probabilităţile de tranziţie nule dintr-o stare în alta care nu a fost găsită la antrenament, probabilitatea tranziţiei în orice stare este dată de interpolarea liniară Jelinek-Mercer:

$$p(t_{k+1}|t_{k-1},t_k) = \lambda_1\, p(t_{k+1}) + \lambda_2\, p(t_{k+1}|t_k) + \lambda_3\, p(t_{k+1}|t_{k-1},t_k)$$
$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

Probabilităţile de emisie sunt de asemenea estimate din corpusul de antrenare la care se adaugă un lexicon care conţine forme ocurente ale cuvintelor împreună cu etichetele morfosintactice corespunzătoare[31]. În cazul în care adnotatorul găseşte un cuvânt pe care nu l-a întâlnit la antrenare, euristicile de ghicire a etichetei intră în funcţiune iar în acest punct implementarea noastră diferă de descrierea originală prin:

- analiza de sufix (adică atribuirea unei etichete morfosintactice t unui cuvânt pe baza analizei ultimelor m caractere din cele n ale cuvântului:

$$i = \overline{0,\ldots,m-1}\,,$$
$$P(t|l_{n-i+1},\ldots,l_n) = \frac{\hat{P}(t|l_{n-i+1},\ldots,l_n) + \theta_i\, P(t|l_{n-i},\ldots,l_n)}{1+\theta_i}\,,$$
$$\hat{P}(t|l_{n-i+1},\ldots,l_n) = \frac{f(t,l_{n-i+1},\ldots,l_n)}{f(l_{n-i+1},\ldots,l_n)},\ P(t) = \hat{P}(t)\,,$$
$$\bar{P} = \frac{1}{s}\sum_{j=1}^{s}\hat{P}(t_j),\ \theta_i = \frac{1}{s-1}\sum_{j=1}^{s}(\hat{P}(t_j)-\bar{P})^2$$

[30]De exemplu, pentru substantiv, între gen şi număr, genul poate fi eliminat pentru că în afară de cazul în care determină acordul cu un adjectiv, genul substantivului nu mai determină nicio altă etichetă spre deosebire de număr care apare atât la acordul cu adjectivul cât şi la acordul cu un verb în situaţia de subiect-predicat.

[31]Pentru română acest lexicon conţine aproximativ 570000 de intrări iar pentru engleză, 126000. Pentru un exemplu, vezi figura 2.9.

unde s este numărul de CTAG-uri, $\hat{P}$ este probabilitate estimată din corpusul de antrenare, f este funcţie de frecvenţă iar θ_i sunt parametrii de ajustare a probabilităţii finale, vezi [7] pentru detalii) se face doar pentru etichetele morfosintactice aparţinând claselor deschise: substantive, adjective, verbe şi adverbe pentru că cel puţin pentru română şi engleză credem că am epuizat lista cuvintelor funcţionale astfel încât marea majoritate a lor se află în lexicoanele noastre;

- dacă cuvântul este necunoscut dar începe cu literă mare şi nu se află la început de frază, adnotatorul are opţiunea de a-l eticheta ca fiind substantiv propriu, o etichetă generică care se potriveşte pentru orice tip de entitate. Această euristică este un parametru configurabil şi funcţionează în cazul în care textul abundă în denumiri;

- în cazul în care recunoaşterea entităţilor denumite a fost rulată înaintea adnotării cu etichete morfosintactice, entităţile au deja eticheta morfosintactică asociată (vezi figura 2.2; dacă se aplică o regulă, entitatea va putea fi adnotată cu MSD-ul, CTAG-ul sau EMSD-ul[32] corespunzător: toate cele trei etichete sunt în corespondenţă) ceea ce este un câştig atât pentru decodorul Viterbi [123] (unele puncte sunt prestabilite în calea optimă) cât şi pentru analiza de sufix pentru că aceasta greşeşte mai mult în cazul entităţilor (nu au terminaţii regulate).

Tehnica adnotării stratificate ([100, 110]) se bazează pe faptul că adnotatoarele probabilistice dau rezultate foarte bune cu mulţimea CTAG[33] şi că funcţia de recuperare a unui MSD dintr-un CTAG împreună cu forma ocurentă a cuvântului adnotat este deterministă în cazul în care cuvântul adnotat apare în lexicon împreună cu MSD-ul corespunzător. TTL implementează de asemenea această tehnică a adnotării stratificate cu care poate să readnoteze cu etichete MSD un text adnotat cu etichete CTAG . În cazul în care există ambiguităţi la operaţia de recuperare, TTL foloseşte o listă de reguli pentru a elimina ambiguitatea. De exemplu în figura 2.8, în engleză avem o regulă care precizează că ambiguitatea de MSD `Di.../Pi...` se rezolvă astfel: se alege `Di...` dacă la poziţia `+1` în text[34] apare un substantiv comun care se acordă cu determinatorul după număr (poziţia marcată cu `#` desemnează acord) sau dacă la poziţia `+1` se află un substantiv propriu sau dacă la poziţia `+1` se află un adjectiv (`A`), adverb (`R`) sau numeral (`M`) şi la

[32]Etichetele EMSD sunt extensii ale etichetelor MSD şi au fost introduse de autor pentru a descrie entităţile recunoscute de TTL (vezi anexa A).

[33]Care are o cardinalitate cu mult redusă faţă de mulţimea MSD.

[34]Faţă de poziţia ambiguă. În această fază, textul este o listă de perechi unitate lexicală, MSD sau unitate lexicală, MSD-uri dacă avem ambiguitate.

```
choose ^Di..#$ if
   +1 Nc.# or
   +1 ^Np or
   +1 ^[ARM] and +2 Nc.# or
   +1 ^[ARM] and +2 ^Np or
   +1 ^[ARM] and +2 ^[ARM] and +3 Nc.#
end
```

Figura 2.8: Regulă pentru a rezolva ambiguitatea de MSD `Di.../Pi....`

poziţia +2 se află un substantiv comun care se acordă cu determinatorul după număr, etc.

Dacă un cuvânt nu se află în lexicon, TTL încearcă să îi ghicească MSD-ul construind un model de sufixe similar cu cel descris mai sus implementat de TnT. O metodă mai adecvată care utilizează Principiul Entropiei Maxime atât pentru a asigura MSD-urile cuvintelor necunoscute cât şi pentru a rezolva ambiguităţile de MSD-uri, este descrisă în [14].

2.1.5 Lematizarea

Lematizarea[35] este o operaţie de normalizare a formei ocurente a unui cuvânt, normalizare care elimină orice tip de flexiune din forma ocurentă a cuvântului. Este operaţia care transformă orice formă ocurentă a unui cuvânt într-o formă standard care de obicei este adoptată de dicţionare. Lematizarea nu trebuie confundată cu indentificarea rădăcinii cuvântului[36] (vezi de exemplu algoritmul din [85]) pentru că de exemplu "connecting" are lema "connect" dar "connections" are lema "connection" şi nu "connect" care este rădăcina cuvântului.

În literatura de specialitate lematizarea se face folosind reguli de eliminare a flexiunilor achiziţionate automat ([84, 68]) sau nu ([81]) care transformă forma ocurentă a unui cuvânt în lemă. În [68], regulile de lematizare sunt văzute ca nişte clase în care toate formele ocurente care se lematizează cu regula respectivă se încadrează. Totuşi în lucrarea menţionată nu se ţine cont de eticheta morfosintactică a cuvântului dar în schimb, contexul cuvântului, o fereastră de 7 cuvinte centrată în cuvântul de lematizat, se ia în conside-

[35] "Lemmatization" în engleză.

[36] Operaţie care în engleză se numeşte "stemming".

rare pentru operaţia de lematizare, lucru menit să suplinească parţial lipsa etichetei morfosintactice.

Operaţia de lematizare urmează adnotarea cu etichete morfosintactice pentru că lema unui cuvânt depinde de categoria gramaticală a acestuia iar categoria gramaticală a unui cuvânt depinde la rândul ei de contextul de apariţie a cuvântului. TTL se bazează pe un model de lematizare extras automat dintr-un lexicon care conţine pentru fiecare formă ocurentă a unui cuvânt, lema şi MSD-ul acestuia (care prin corespondenţa descrisă în secţiunea anterioară poate fi transformat în CTAG-ul corespunzător, vezi figura 2.9). Acest model conţine pentru fiecare etichetă CTAG t_j (din mulţimea de etichete care aparţin clasei părţilor de vorbire care flexionează) două componente:

- o mulţime de reguli care dacă sunt aplicate pe o formă ocurentă a unui cuvânt produc lema acestuia. Aceste reguli sunt extrase automat astfel: pentru fiecare triplu formă ocurentă, lemă, etichetă CTAG, $\langle w_i, l_i, t_j \rangle$, se determină secvenţa LCS ("Longest Common Subsequence", vezi [36]) între w_i şi l_i. Dacă LCS are lungimea mai mare sau egală cu jumătatea lungimii lui w_i[37] atunci aceasta se înlocuieşte cu şirul de caractere "{LCS}" atât în w_i cât şi în l_i rezultând regula

  $$p_{w_i}\{LCS\}s_{w_i} \rightarrow p_{l_i}\{LCS\}s_{l_i}$$

  Această regulă precizează că dacă din forma ocurentă (care a fost adnotată cu eticheta morfosintactică t_j) se elimină prefixul p_{w_i} şi/sau sufixul s_{w_i} şi se adaugă la şirul de caractere rămas prefixul p_{l_i} şi/sau sufixul s_{l_i}[38] se obţine o lemă posibilă a formei ocurente (vezi figura 2.10).

- un model Markov (vezi [57, pag. 317]) antrenat pe lemele formelor ocurente cu eticheta morfosintactică t_j. Stările automatului sunt date de secvenţe de 4 caractere iar probabilităţile de tranziţie se calculează cu relaţia de interpolare liniară Jelinek-Mercer (pentru calculul parametrilor $\lambda_i, i \in \{1, 2, 3, 4\}$, vezi [7]). Acest model va fi folosit pentru a ordona lemele candidate ale unei forme ocurente.

Algoritmul de lematizare aşteaptă la intrare o listă de cuvinte, fiecare cuvânt având asociat CTAG-ul corespunzător. Dacă S este o listă de perechi formă ocurentă, CTAG $\langle w_i, t_i \rangle$, algoritmul de lematizare urmează paşii:

[37]Dacă nu se întâmplă acest lucru, lematizarea nu se poate face printr-o regulă şi acest caz reprezintă deci o excepţie. De exemplu în engleză avem plurale neregulate cum ar fi "goose" singular, "geese" plural. Între cele două forme LCS este "se" care nu este rădăcina cuvântului. Această condiţie încearcă să garanteze că atunci când se găseşte LCS aceasta conţine rădacina cuvântului.

[38]Oricare din $p_{w_i}, s_{w_i}, p_{l_i}, s_{l_i}$ poate fi egal cu şirul vid.

Formă ocurentă	Lemă	MSD	CTAG
aramă	aramă	Ncfsrn	NSRN
arama	aramă	Ncfsry	NSRY
arame	aramă	Ncfp-n	NPN
arame	aramă	Ncfson	NSON
aramei	aramă	Ncfsoy	NSOY
aramele	aramă	Ncfpry	NPRY
aramelor	aramă	Ncfpoy	NPOY

Figura 2.9: Formele flexionare ale substantivului "aramă".

1. dacă forma ocurentă a cuvântului w_i este întâlnită în lexicon extrage lema din intrarea respectivă folosind cheia $\langle w_i, t_i \rangle$; dacă lema nu este unică, alege cea mai frecventă lemă dintr-o listă de frecvenţe pentru leme[39].

2. aplică toate regulile de lematizare specifice etichetei t_i şi obţine o listă de leme candidate. Ordonează descrescător această listă după probabilităţile furnizate de modelul Markov pentru eticheta t_i şi extrage lema din capul listei. De exemplu, pentru perechea "`cartea/NSRY`" lemele candidate date de regulile din figura 2.10 (în ordinea din figură de sus în jos) sunt: ⟨"`-`", "`carteă`", "`carte`", "`cartee`"⟩. Ne aşteptăm ca poziţiile 2 şi 4 din lista anterioară să fie foarte puţin probabile pentru limba română pentru că secvenţele de 4 caractere "`teă<`" şi "`tee<`" la sfârşit de lemă sunt practic inexistente în modelul Markov pentru CTAG-ul "`NSRY`" ("`<`" indică sfârşit de lemă şi este un caracter de control inserat).

La selecţia lemei canditate în pasul 2 de mai sus, modelul Markov tinde să aleagă lema cu lungimea cea mai mică pentru că pentru o lemă l cu n caractere, probabilitatea furnizată de model este:

$$p(l) = p(c_1, c_2, \ldots, c_n) = \prod_{i=1}^{n+1} p(c_i | c_{i-3}, c_{i-2}, c_{i-1})$$

unde c_{-2}, c_{-1}, c_0 şi c_{n+1} sunt caractere inserate (nu fac parte din lemă) care marchează începutul respectiv sfârşitul lemei. Se observă că cu cât n este

[39]Dacă lema nu este unică ca în cazul "copii" cu lemele "copil" şi "copie" atunci pentru a extrage lema potrivită avem nevoie de DSA iar pentru a putea face DSA avem nevoie de lemă. Această dependenţă mutuală este rezolvată la nivelul lematizării.

CTAG	Regulă	Frecvenţă	Exemplu
NSRY	{LCS}ul->{LCS}	6732	băiatul → băiat
NSRY	{LCS}a->{LCS}ă	4307	maşina → maşină
NSRY	{LCS}a->{LCS}	3525	cartea → carte
NSRY	{LCS}a->{LCS}e	1540	colecţia → colecţie

Figura 2.10: Reguli de lematizare pentru un substantiv singular, articulat, nominativ/acuzativ.

mai mare, $p(l)$ este mai mică pentru că avem un produs de numere subunitare. Pentru a atenua această tendinţă am introdus în procesul de selecţie şi frecvenţa regulii care a produs lema candidată (vezi figura 2.10) şi în consecinţă, am decis să introducem următoarea combinaţie de euristici în locul punctului 2 de mai sus:

1. generează două leme: una cu modelul Markov (pasul 2 de mai sus), l_{MM} şi cea de-a doua prin aplicarea celei mai frecvente reguli pentru eticheta t_i, l_{FRQ}; dacă $l_{MM} = l_{FRQ} = l$, alege lema l.

2. la antrenare, pentru fiecare etichetă t_j pentru care se construieşte modelul de lematizare, evaluează precizia euristicilor MM şi FRQ separat şi împreună pe N tripluri $\langle w_i, l_i, t_j \rangle$. Fie c numărul de cazuri în care MM şi FRQ au dat aceeaşi lemă corectă, a numărul de cazuri în care MM a dat lema corectă şi b numărul de cazuri în care FRQ a dat lema corectă. Atunci, $x = a - c$ este numărul de cazuri în care MM a dat lema corectă iar FRQ nu şi $y = b - c$ este numărul de cazuri în care FRQ a dat lema corectă iar MM nu.

3. fie probabilităţile:

   - probabilitatea ca MM să furnizeze lema corectă dacă FRQ nu a dat-o ($l_{MM} \neq l_{FRQ}$): $p(MM|\neg FRQ) = \frac{x/N}{(N-b)/N} = \frac{x}{N-c-y}$
   - probabilitatea ca MM să furnizeze lema greşită dacă FRQ a dat lema corectă ($l_{MM} \neq l_{FRQ}$): $p(\neg MM|FRQ) = \frac{y/N}{b/N} = \frac{y}{c+y}$

4. dacă $p(MM|\neg FRQ) > p(\neg MM|FRQ)$ alege lema l_{MM}; în caz contrar alege l_{FRQ}.

Acest mecanism asigură selecţia automată a euristicii optime pentru fiecare CTAG.

În tabelul 2.1 se află rezultatele lematizării statistice a câte 1000 (română) respectiv 200 (engleză) de forme ocurente care au un CTAG inclus în modelul de lematizare. Notăm că rezultatele date sunt lematizări ale unor cuvinte necunoscute (care nu se află în lexicon[40]). Preciziile de 100% apar din cauză că exemplele sunt puţine (până în 10) şi se rezolvă toate cu aceeaşi regulă. De asemenea performanţa slabă a algoritmului de lematizare în engleză pentru verbele la participiu trecut (`PPAS`) poate fi explicată prin faptul că mulţimea de test a conţinut şi verbe neregulate pentru care lematizarea pe bază de prefixe şi sufixe este practic inaplicabilă. Un rezultat surprinzător se înregistrează la lematizarea participiilor prezente (`PPRE`) în engleză unde majoriatea erorilor se datorează faptului că regula cea mai frecventă în acest caz este înlăturarea sufixului "`ing`" în detrimentul celei care pe lângă această operaţie mai adaugă sufixul "`e`". Astfel verbe ca "breathing", "seizing" sunt lematizate incorect ca "breath" sau "seiz" în loc de "breathe" şi "seize".

În română remarcăm un prim rezultat slab în dreptul adjectivelor singulare, nearticulate (`ASN`) care poate fi explicat prin faptul că în lexiconul românesc adjectivele sunt lematizate întotdeauna la forma de masculin, singular, nearticulat. Deci "frumoasă" are lema "frumos" iar această transformare nu este atât de regulată pe cât ne-am fi aşteptat. Alte rezultate slabe apar în dreptul substantivelor comune, plural (`NPN, NPOY, NPRY`) în care regula cea mai frecventă furnizează leme incorecte cu lungime mai mică, leme care sunt agreate şi de modelul Markov conform observaţiilor pe care le-am făcut mai sus.

2.2 SemCor2.0: O versiune adnotată în limba română

SemCor ([67]) este un corpus de limbă engleză (americană) adnotat cu etichete de sens din WordNet 1.6 ([66]). Corpusul care a fost adnotat este corpusul Brown ([50]), un corpus balansat care conţine articole din presă, literatură, texte ştiinţifice şi religioase. Este un corpus de mici dimensiuni (1014312 de cuvinte, [28]) după standardele actuale dar care este foarte important pentru cercetările de DSA pentru că este practic singurul corpus disponibil pentru antrenarea algoritmilor de DSA asistată şi pentru testarea oricărui tip de algoritm de DSA. Există numeroase lucrări de DSA care fie raportează rezultate de precizie calculate pe acest corpus, fie îl folosesc la antrenare (de exemplu, [74, 97, 19, 63, 65]).

[40]De fapt am antrenat modelele de lematizare pe lexicoanele existente din care am extras în prealabil datele de test.

Română	
CTAG	**Precizie**
APN	93.4%
APOY	96.5%
APRY	96%
ASN	50.2%
ASON	93.6%
ASOY	97.8%
ASRY	95.8%
ASVN	100%
ASVY	99%
NPN	56.5%
NPOY	52.9%
NPRY	56.3%
NPVY	100%
NSON	88.5%
NSOY	80.2%
NSRY	75.8%
NSVN	50%
NSVY	82.9%
V1	90.4%
V2	82.6%
V3	75.8%
VG	84.8%
VPPF	88%
VPPM	89.5%
VPSF	86.2%
VPSM	91.9%
Media	82.869%

Engleză	
CTAG	**Precizie**
NNS	83.5%
PAST	71.5%
PAST1	100%
PAST2	100%
PAST3	75%
PPAS	65.5%
PPRE	66.5%
VERB3	89%
Media	81.375%

Tabela 2.1: Rezultatele lematizării pentru română şi engleză.

Aşa cum este descris în [67], SemCor a fost adnotat cu etichete morfosintactice cu adnotatorul Brill ([8]) iar adnotarea semantică a fost realizată urmând metoda secvenţială: cuvânt cu cuvânt, frază cu frază în ordinea apariţiei în corpus. Lexicografii au folosit o interfaţă grafică (ConText) dezvoltată special pentru activitatea de adnotare semantică. Această interfaţă afişează pentru fiecare cuvânt conţinut[41], sensurile din WordNet[42] corespunzătoare părţii sale de vorbire în context. Rezultatul adnotării îl reprezintă pointerul către sensul din WordNet care se potriveşte în context.

Traducerea în limba română ([55, 60]) a SemCor-ului a fost realizată astfel încât să se obţină o exprimare cursivă în limba română respectându-se pe cât posibil ordinea cuvintelor din engleză. Acest lucru a fost impus traducătorilor pentru a facilita o aliniere lexicală cât mai bună între engleză şi română. Corpusul a fost aliniat la nivel de cuvânt cu aliniatorul lexical **YAWA** (pentru o descriere a lui vezi capitolul 3 şi [118]) pentru a se putea trece la transferul adnotărilor de sens din engleză în română. Paşii de prelucrare a corpusului paralel englez-român sunt următorii:

- adnotarea morfosintactică şi lematizarea ambelor parţi cu TTL întrucât YAWA funcţionează pe texte paralele adnotatate morfosintactic (cu etichete compatibile MULTEXT-East) şi lematizate.
- alinierea lexicală cu YAWA în vederea transferului de sensuri.
- transferul de sensuri folosind alinierea lexicală şi corespondenţa dintre reţeaua semantică a limbii engleze PWN2.0 şi cea a limbii române ROWN2.0.

2.2.1 Adnotarea textului englezesc din SemCor2.0

Aşa cum am afirmat, partea engleză a corpusului a fost adnotată iniţial cu adnotatorul morfosintactic al lui Brill însă la o primă inspecţie a acestei adnotări s-au observat multe greşeli de adnotare (vezi [83] pentru observaţii similare). De exemplu, în primul fişier al corpusului, `br-a01`, în propoziţia numărul 2, întâlnim adnotările din figura 2.11. Totuşi, aşa cum se raportează şi în [83], erorile cele mai frecvente sunt observate la nivelul cuvintelor funcţionale[43] lucru care este explicabil prin faptul că adnotarea cu sensuri s-a făcut la nivelul cuvintelor conţinut ale căror etichete morfosintactice au fost

[41]Substantiv, adjectiv, verb sau adverb.

[42]Se afişează sinseturile în care cuvântul este un literal împreună cu glosele asociate. Vezi secţiunea următoare pentru definiţiile acestor termeni.

[43]Un cuvânt care nu este cuvânt conţinut, adică substantiv, adjectiv, verb sau adverb.

```
                  <wf cmd=ignore pos=IN>for</wf>
                  <wf cmd=ignore pos=DT>the</wf>
                  <wf cmd=done pos=NN lemma=manner
                     wnsn=1 lexsn=1:07:02::>manner</wf>
->                <wf cmd=done pos=RB ot=notag>in</wf>
->                <wf cmd=done pos=RB ot=notag>which</wf>
                  <wf cmd=ignore pos=DT>the</wf>
                  <wf cmd=done pos=NN lemma=election
                     wnsn=1 lexsn=1:04:01::>election</wf>
                  <wf cmd=done pos=VBD ot=notag>was</wf>
                  <wf cmd=done pos=VB lemma=conduct
                     wnsn=1 lexsn=2:41:00::>conducted</wf>
                  <punc>.</punc>
```

Figura 2.11: "in" este adnotat ca adverb (RB) când ar fi trebuit să fie prepoziţie (IN); "which" este adverb (!) când această parte de vorbire nici nu se află în clasa sa de ambiguitate. Aici ar fi trebuit să fie pronume relativ (WP).

probabil corectate acolo unde a fost nevoie pentru a se putea atribui corect sensul.

O a doua problemă cu privire la partea de limbă engleză a corpusului a constituit-o formarea expresiilor. Sunt trei tipuri de expresii în engleză: unele care există în PWN2.0 şi sunt recunoscute corect în contextul lor de apariţie, altele care există în PWN2.0 dar care nu formează expresii în contextul dat şi expresii care deşi sunt marcate ca atare în text, nu există în PWN2.0. În total, în engleză există 14561 de expresii unice recunoscute (marcate cu "_"). În tabelul 2.2 sunt expuse primele 62 de expresii din engleză după rangul de frecvenţă. Dintre acestea, de exemplu, "of_this", "in_which", "of_it" corespunzătoare exemplelor

(2.4) ... the size of_this city ...

(2.5) ... a resonant circuit in_which the capacitor ...

(2.6) The name of_it (RB!) is Gore_Court, ...

nu există în PWN2.0. Putem găsi o justificare la nivel sintactic pentru "of_this" ca fiind un determinator cu cazul marcat de prepoziţie (s-ar traduce în română cu "acestui/acestei") şi pentru "of_it" ca fiind un determinator posesiv (în română "lui/ei") dar care este adnotat ca adverb.

Oricum pentru că etichetele morfosintactice ale acestor expresii sunt în marea majoritate a cazurilor greşite şi pentru că aceste echivalenţe sunt lăsate spre rezolvare aliniatorului lexical, am decis să eliminăm toate expresiile care nu se regăsesc în PWN2.0 exceptând acele expresii care erau adnotate ca entităţi: grup, locaţie, persoană (de exemplu "`du_Pont`" care are eticheta de sens "`group(1)`", adică "`any number of entities (members) considered as a unit`" – definiţie din PWN2.0).

Tot în tabelul 2.2 remarcăm expresia "`in_this`" al cărei unic sens de adverb în PWN2.0 este

```
therein, in_this, in that -- ((formal) in or into that thing or place;
    "they can read therein what our plans are")
```

dar care în exemple de tipul

(2.7) ... I ever saw `in_this` county ...

(2.8) ... which is so vividly real `in_this` play.

nu reprezintă expresia cu sensul citat ci succesiunea de prepoziţie, determinator demonstrativ. Pe parcursul a două luni, autorul a verificat toate "expresiile" suspecte de a nu fi expresii în contextul lor care aveau o frecvenţă de apariţie de cel puţin 10. Candidaţii s-au selectat din expresii care conţineau cel puţin un cuvânt funcţional (prepoziţie, determinator, etc.).

După ce etapa de corectare a expresiilor s-a încheiat, s-a trecut la adnotarea morfosintactică şi lematizarea textului. Adnotatorul morfosintactic din TTL s-a antrenat pe fracţiunea în limba engleză a corpusului multilingv 1984 ([100]). În plus, toate cuvintele necunoscute din SemCor2.0 (care nu apar în 1984) au fost adăugate la lexionul de engleză (extras iniţial din 1984) fiecare împreună cu lema şi MSD-ul asociat. Informaţia de lemă este dată de PWN2.0 iar MSD-ul este derivat cu ajutorul unui analizor morfologic rudimentar cu expresii regulate care asociază terminaţiile formelor ocurente cu MSD-urile corespunzătoare. De exemplu dacă "`booking`" nu apare în lexicon, PWN2.0 ne dă două leme: "`booking`" ca substantiv şi "`book`" ca verb. Pentru că substantivul nu se termină în "`s|es`", avem MSD-ul "`Ncns`" (singular, neutru) iar verbul, pentru că se termină în "`ing`" primeşte MSD-ul "`Vmg`" (gerunziu). Cu aceste adăugiri, lexiconul de engleză s-a mărit cu aproximativ 64000 de forme ocurente necunoscute şi a fost şi el inclus în antrenarea adnotatorului morfosintactic şi a lematizorului.

Adnotarea morfosintactică pe partea de engleză a corpusului SemCor2.0 (să o denumim SemCor2.0-en) a fost apoi corectată de autor pe parcursul a aproximativ 4 săptămâni[44] folosind tehnica antrenării pe datele de test[45]

[44]SemCor2.0-en are 778400 de unităţi lexicale incluzând aici şi punctuaţia.

[45]În engleză, "biased evaluation".

Frecvenţă	Expresie	Frecvenţă	Expresie
229	United_States	64	had_to
217	a_few	63	many_of
210	of_this	59	that_is
201	more_than	59	as_well_as
194	at_least	58	think_of
168	New_York	58	In_addition
163	in_this	56	Of_course
156	of_course	55	fiscal_year
151	going_to	53	up_to
147	a_little	52	too_much
134	not_only	52	in_order
128	as_well	49	President_Kennedy
126	such_as	47	thought_of
126	per_cent	47	of_that
126	in_which	47	in_front
117	U._S.	47	a_couple_of
115	at_all	46	sort_of
114	the_most	44	of_which
110	and_then	44	in_terms_of
99	all_of	44	United_Nations
98	most_of	43	set_up
96	so_that	43	du_Pont
96	rather_than	42	all_right
86	Rhode_Island	42	a_bit
78	for_example	41	at_the_same_time
77	have_to	41	all_over
76	of_it	40	so_much
76	kind_of	40	old_man
74	no_longer	40	high_school
67	in_fact	40	find_out
64	less_than	40	at_once

Tabela 2.2: Primele 62 de expresii ca rang de frecvenţă din SemCor-ul englezesc.

([121]). În final precizia adnotatorului morfosintactic antrenat şi testat pe SemCor2.0-en a fost de aproximativ 99%. Concomitent cu corecţiile de etichete morfosintactice s-au corectat şi lemele formelor ocurente care aveau eticheta morfosintactică greşită.

2.2.2 Adnotarea textului românesc din SemCor2.0

Au fost traduse 81 de fişiere din SemCor2.0 din totalul de 352 după cum urmează:

- `br-a01`, `br-a02`, `br-a11` până la `br-a15` inclusiv,
- `br-b13`, `br-b20`,
- `br-c01`, `br-c02`, `br-c04`,
- `br-d01` până la `br-d04` inclusiv,
- `br-e01`, `br-e02`, `br-e04`, `br-e21`, `br-e24`, `br-e29`,
- `br-f03`, `br-f10`, `br-f19`, `br-f43`,
- `br-g01`, `br-g11`, `br-g15`,
- `br-h01`,
- `br-j01` până la `br-j20` inclusiv, `br-j23`, `br-j37`, `br-j52` până la `br-j60` inclusiv, `br-j70`,
- `br-k01` până la `br-k19` inclusiv.

Traducerea românească a corpusului (să o numim SemCor2.0-ro) s-a făcut pe varianta iniţială a textului SemCor2.0 şi din această cauză toate expresiile forţate în engleză au fost traduse ca atare în limba română. În plus, traducătorii au introdus compuşi în limba română pentru a facilita alinierea lexicală:

- timpuri compuse, moduri ale verbelor: "`a_spus`", "`au_făcut`", "`va_pleca`", "`ar_veni`", "`să_fie`", etc.
- grade de comparaţie ale adjectivelor/adverbelor: "`mai_frumos`", "`cel_mai_tare`", etc.

- alţi compuşi: "`un_personal cleric`", "`la_1 ianuarie`", "`comisia din_Fulton`", etc. din care unii pot fi justificaţi din punct de vedere sintactic, alţii nu[46].

Pentru că pe SemCor2.0 s-au operat corecturile menţionate în secţiunea anterioară după ce traducerea românească a fost făcută (rezultând SemCor2.0-en), a fost necesară o procesare similară a SemCor2.0-ro astfel încât:

- expresiile din engleză să rămână traduse cu expresii româneşti acolo unde acest lucru s-a întâmplat;
- expresiile forţate din română să dispară.

Traducătorii nu au fost familiarizaţi cu procedeul de aliniere lexicală iar din această cauză, am fost obligaţi să eliminăm toţi compuşii pe care aceştia i-au introdus crezând că uşurează procesul de aliniere: am eliminat de exemplu compuşii care exprimau timpuri compuse şi moduri ale verbelor pentru că YAWA aliniază automat verbele auxiliare din română la corespondentele lor din engleză sau în cazul în care acestea nu există, la verbul predicativ. Am eliminat de asemenea toţi compuşii care în opinia noastră nu aveau o justificare de natură morfologică şi/sau sintactică fie pentru limba română, fie pentru traducerea lor în engleză. O menţiune specială o facem la adresa expresiilor din engleză care reprezintă nume de entităţi (persoană, grup sau locaţie) care în limba română fie au fost traduse ca expresii[47], fie au fost lăsate neschimbate[48], fie au fost traduse dar nu ca expresii[49].

După corecţiile operate pe SemCor2.0-ro cu privire la expresii şi compuşi[50], dintr-un număr de 8685 de compuşi[51] unici care apăreau în varianta iniţială a SemCor2.0-ro au mai rămas 3433 în varianta editată. Toţi aceşti compuşi au fost adăugaţi la resursele de limba română ale modulului TTL pentru a putea fi recunoscuţi la viitoarele procesări.

Adnotarea morfosintactică şi lematizarea corpusului SemCor2.0-ro au fost obţinute de asemenea cu modulul TTL. În ce priveşte adnotarea morfosintactică, TTL s-a antrenat pe corpusurile româneşti 1984, Republica şi Ziare ([100]) şi pe partea în limba română a corpusului NAACL ([64]). Lematizorul a fost antrenat pe lexiconul românesc de forme ocurente care conţine

[46] Cel puţin în opinia autorului.

[47] i.e. "`Fulton_Superior_Court`" a fost tradus cu "`Curtea_Superioară-din-Fulton`".

[48] În general numele de persoane şi de locaţii nu au fost traduse.

[49] Vezi "`Fulton_County_Grand_Jury`" cu "`Marele Juriu din Fulton`".

[50] Orice expresie este un compus, însă reciproca nu e valabilă. Compuşii sunt acele unităţi lexicale care sunt marcate în text cu caracterul "_".

[51] În formă ocurentă. Aici se includ şi expresiile.

aproximativ 570000 de înregistrări (o fracţiune din acest lexicon este expusă în figura 2.9). Corectura adnotării morfosintactice s-a bazat pe aceeaşi tehnică a antrenării pe datele de test iar rezultatul testării a indicat o precizie de peste 99%.

2.2.3 Transferul sensurilor din engleză în română

SemCor2.0 a fost iniţial adnotat morfosintactic cu adnotatorul lui Brill (să numim această variantă a corpusului SC20-en-Brill) iar aceste etichete morfosintactice au stat apoi la baza adnotării semantice. Textele SemCor2.0-en şi SemCor2.0-ro au fost însă prelucrate cu TTL şi apoi corectate (fie aceste corpusuri SC20-en-TTL şi SC20-ro-TTL) astfel încât pe corpusul paralel să se poată rula aliniatorul lexical YAWA. Readnotarea morfosintactică şi relematizarea corpusului SC20-en-Brill a introdus astfel o nouă problemă pe lângă problema de transfer interlingual: transferul de sensuri din SC20-en-Brill în SC20-en-TTL. În efectuarea acestui prim transfer de sensuri ne interesează numai cuvintele conţinut (numai ele sunt adnotate semantic) iar identitatea de etichete morfosintactice este dată de o echivalenţă de clase ale părţilor de vorbire:

- substantive: Brill `NN` $\Leftrightarrow$ MSD `N`,
- verbe: Brill `VB` $\Leftrightarrow$ MSD `Vm`,
- adjective: Brill `JJ` $\Leftrightarrow$ MSD `Af` şi
- adverbe: Brill `RB` $\Leftrightarrow$ MSD `R`.

În consecinţă, pentru frazele s^i_{Brill} din SC20-en-Brill şi corespondenta acesteia din SC20-en-TTL, s^i_{TTL}, am întâlnit următoarele cazuri:

1. cuvintele w^j_{Brill} din s^i_{Brill} şi w^j_{TTL} din s^i_{TTL} au aceeaşi lemă şi aceeaşi etichetă morfosintactică. În acest caz, copiază pur şi simplu eticheta semantică de la w^j_{Brill} la w^j_{TTL};
2. cuvintele w^j_{Brill} din s^i_{Brill} şi w^j_{TTL} din s^i_{TTL} au aceeaşi etichetă morfosintactică dar nu au aceeaşi lemă. În acest caz, copiază lema şi eticheta semantică de la w^j_{Brill} la w^j_{TTL};
3. cuvintele w^j_{Brill} din s^i_{Brill} şi w^j_{TTL} din s^i_{TTL} au aceeaşi lemă dar nu au aceeaşi etichetă morfosintactică. În acest caz, copiază eticheta morfosintactică şi eticheta semantică de la w^j_{Brill} la w^j_{TTL} transformând eticheta morfosintactică Brill în etichetă morfosintactică MSD cu o tabelă de corespondenţă.

Caz de transfer	**SC20-en-Brill**	**Procent**	**Transfer?**
1. $l^j_{Brill} = l^j_{TTL}, t^j_{Brill} = t^j_{TTL}$	204386	88.90%	da
2. $l^j_{Brill} \neq l^j_{TTL}, t^j_{Brill} = t^j_{TTL}$	12020	5.23%	da
3. $l^j_{Brill} = l^j_{TTL}, t^j_{Brill} \neq t^j_{TTL}$	3813	1.66%	da
4. $l^j_{Brill} \neq l^j_{TTL}, t^j_{Brill} \neq t^j_{TTL}$	2614	1.14%	nu
5. $w^j_{Brill} \notin s^i_{TTL}$	7061	3.07%	nu
TOTAL	220219	95.79%	da

Tabela 2.3: Transferul de etichete semantice SC20-en-Brill–SC20-en-TTL

4. cuvintele w^j_{Brill} din s^i_{Brill} şi w^j_{TTL} din s^i_{TTL} nu au nici aceeaşi lemă, nici aceeaşi etichetă morfosintactică. În acest caz am creditat adnotarea TTL şi nu am transferat eticheta semantică a lui w^j_{Brill};

5. cuvântul w^j_{Brill} nu se află în fraza s^i_{TTL}. Acest lucru se întâmplă datorită faptului că din SC20-en-Brill au fost eliminate expresii (vezi subsecţiunea 2.2.1) care nu se mai regăsesc în SC20-en-TTL.

În tabela 2.3 este cuantificat transferul de etichete semantice din corpusul SC20-en-Brill în varianta TTL a lui, SC20-en-TTL. În total, în SC20-en-Brill există 229894 de adnotări cu etichete semantice ale cuvintelor conţinut. Din acestea, urmând paşii de mai sus s-au importat în SC20-en-TTL 220219 de adnotări care reprezintă un procent de aproximativ 95.79% din mulţimea iniţială de adnotări. Trebuie notat că numărul mare de leme diferite în cazul de transfer 2 se datorează faptului că în SC20-en-Brill, entităţile sunt lematizate ca "`group`", "`person`" sau "`location`" – în total 8608 de astfel de adnotări, pe când TTL le lematizează automat la forma ocurentă.

În figura 2.13 sunt prezentate primele 10 diferenţe de lemă (ca rang de frecvenţă) care au pus probleme transferului de sens. Cele mai multe se datorează modului de lematizare al modulului TTL care pentru comparativele adjectivelor şi adverbelor furnizează ca lemă forma pozitivă a acestora. În PWN2.0, există atât formele pozitive ale adjectivelor şi adverbelor cât şi formele lor de comparativ şi superlativ. De exemplu pentru "`much`", avem intrările din figura 2.12. Dacă adjectivul "`more`" apare în text şi este comparativul lui "`much`", el primeşte sensul numărul 1 al literalului "`more`" şi nu al literalului "`much`".

```
much(1) -- ((quantifier used with mass nouns)
  great in quantity or degree or extent)
more(1), more_than(1) -- ((comparative of 'much' used with mass nouns)
  a quantifier meaning greater in size or amount or extent or degree)
most(1) -- (the superlative of 'much' that can be used with mass nouns
  and is usually preceded by 'the'; a quantifier meaning the greatest in
  amount or extent or degree)
```

Figura 2.12: Adjectivul "much" în Princeton WordNet 2.0.

Frecvenţă	SC20-en-Brill	SC20-en-TTL
112	more/more(1)/RB	more/much/Rmc
72	best/best(1)/JJ	best/good/Afs
50	better/better(1)/JJ	better/good/Afc
48	larger/larger(1)/JJ	larger/large/Afc
44	more/more(1)/JJ	more/much/Afc
41	greater/greater(1)/JJ	greater/great/Afc
31	smaller/smaller(1)/JJ	smaller/small/Afc
28	services/services(1)/NN	services/service/Ncnp
27	steps/steps(1)/NN	steps/step/Ncnp
26	words/words(1)/NN	words/word/Ncnp

Figura 2.13: Exemple de diferenţe în cazul de transfer 2 (leme diferite).

Frecvenţă	SC20-en-Brill	SC20-en-TTL
42	today/today(1)/NN	today/today/Rmp
42	more/more(1)/JJ	more/more/Rsc
40	much/much(1)/JJ	much/much/Rmp
38	latter/latter(1)/JJ	latter/latter/Ncns
34	only/only(1)/JJ	only/only/Rmp
30	else/else(1)/RB	else/else/Afp
30	alone/alone(1)/RB	alone/alone/Afp
25	such/such(1)/RB	such/such/Afp
25	much/much(1)/NN	much/much/Rmp
24	public/public(1)/NN	public/public/Afp

Figura 2.14: Exemple de diferenţe în cazul de transfer 3 (etichete morfosintactice diferite).

După ce am obţinut corpusul SC20-en-TTL adnotat cu sensuri[52], am aliniat lexical cele două jumătăţi ale corpusului, SC20-en-TTL şi SC20-ro-TTL, în vederea efectuării celui de-al doilea transfer semantic: transferul sensurilor din engleză în română. Aliniatorul lexical folosit a fost YAWA, program care furnizează alinieri multicuvânt de $m : n$. Nu vom descrie aici procedeul de aliniere lexicală (acesta se poate afla din [118] şi este de asemenea dat în capitolul 3) şi vom considera că alinierea lexicală ne funizează o listă de alinieri pentru fiecare pereche de fraze din corpusul paralel. Fie două fraze s_i^{en} şi s_i^{ro} din unitatea de traducere[53] i a corpusului paralel SemCor2.0. Fiecare cuvânt w_j^{en} din s_i^{en} are asociată o listă L_j^{ro} de cuvinte româneşti care se aliniază la el. Această listă poate să fie vidă sau nu. Dacă l_j^{en}, t_j^{en} şi n_j^{en} sunt lema, eticheta morfosintactică şi respectiv eticheta de sens a cuvântului w_j^{en} (adoptăm aceleaşi notaţii şi pentru română), atunci algoritmul de transfer semantic din engleză în română funcţionează astfel:

1. extrage lista A_j^{ro} din L_j^{ro} astfel încât pentru fiecare cuvânt w_k^{ro} din A_j^{ro}, $t_k^{ro} = t_j^{en}$ (egalitatea semnifică de fapt numai identitatea părţilor de vorbire);

2. pentru fiecare triplu $\langle l_j^{en}, l_k^{ro}, t_j^{en} \rangle$, ($l_k^{ro}$ fiind lema cuvântului w_k^{ro} din A_j^{ro}) aplică algoritmul **WSDTool** (vezi [119, 42, 116] şi capitolul 3, pagina 61, punctul 2) şi obţine o mulţime $S_k, k = \overline{1 \ldots |A_j^{ro}|}$ de etichete de sens aplicabile atât lui w_j^{en} cât şi lui w_k^{ro}; aici întâlnim mai multe cazuri (vezi următoarea secţiune şi capitolul 3 pentru detalii şi terminologie):

   - l_j^{en} şi l_k^{ro} aparţin unor sinseturi care sunt în corespondenţă având aceeaşi etichetă de sens – numim acest lucru o *corespondenţă directă* sau CD;
   - l_j^{en} şi l_k^{ro} aparţin unor sinseturi care nu sunt în corespondenţă directă dar între care se poate găsi o cale de cel mult N legături ($0 \leq N \leq 2$) în ierarhia semantică – avem atunci o *corespondenţă indirectă* sau CI;
   - eticheta de sens n_j^{en} nu se află în reţeaua semantică lexicală a limbii române ROWN2.0, caz în care sinsetul corespunzător *nu este implementat* în română: SSNEI;

[52] Am folosit termenii de "(etichetă de) sens" şi "etichetă semantică" ca sinonimi în această subsecţiune. Ei se referă la pointerul către înţelesul din PWN2.0 care este aplicabil în context.

[53] O unitate de traducere este un fragment al corpusului care conţine fraza sau paragraful sursă şi traducerile acesteia/acestuia în limba/limbile corpusului.

- deşi eticheta de sens n_j^{en} se află în reţeaua semantică lexicală a limbii române ROWN2.0, lema l_k^{ro} nu se află în sinsetul românesc corepunzător acestei etichete, caz în care avem un *sinset incomplet*: SSINC.

3. dacă $n_j^{en} \in S_k$, adaugă eticheta de sens n_j^{en} la adnotarea semantică a cuvântului w_k^{ro}.

Pentru că folosim corespondenţa dintre reţelele semantice lexicale ale limbii engleze şi române şi pentru că structurile acestora sunt diferite în funcţie de categoria gramaticală, avem nevoie de perechi de cuvinte aliniate care să aibă aceeaşi categorie gramaticală. La pasul 2, eticheta de sens poate fi comună pentru că ea reprezintă de fapt un identificator de înţeles (vezi următoarea secţiune pentru detalii). În pasul 3 putem întâlni situaţii în care mai multe cuvinte englezeşti să se alinieze la unul românesc şi astfel să avem mai multe etichete de sens pentru cuvântul românesc deşi acest lucru nu s-a întâmplat în practică.

Câteva statistici ale corpusului paralel englez-român SemCor2.0 cât şi ale transferului de sensuri din SemCor2.0-en-TTL în SemCor2.0-ro-TTL sunt redate în tabelele 2.4 şi 2.5. În tabela 2.4 avem o statistică a corpusului paralel englez-român SemCor2.0 în ansamblu şi pe categoriile gramaticale ale cuvintelor conţinut. Frecvenţele sunt absolute iar procentul este calculat ca numărul de cuvinte conţinut adnotate pe numărul de cuvinte conţinut din corpus. În tabela 2.5, procentele sunt calculate din numărul total de cuvinte conţinut existente în română: 88874. Trebuie menţionat faptul că alinierea lexicală de la engleză la română nu a fost corectată de un expert iar în această situaţie, frecvenţele de sinset incomplet, respectiv sinset neimplementat pot fi eronate. Totuşi tinând cont de faptul că YAWA are o măsură a preciziei de aliniere în jurul procentului de 80% ([118]), putem afirma că cel puţin acest procent din cele două frecvenţe sunt reale erori de tipul SSNEI şi SSINC[54].

2.3 Reţeaua semantică a limbii române

Reţeaua semantică lexicală a limbii române, ROWN2.0 ([105, 106]) a luat naştere odată cu proiectul BalkaNet ([108]) finanţat de Comisia Europeană (IST-2000-29388) care îşi propunea să urmeze demersul EuroWordNet ([124]) şi să dezvolte astfel o bază de date de reţele semantice lexicale pentru cinci

[54]Această afirmaţie este susţinută şi de observaţia conform căreia precizia aliniatorului lexical YAWA *pe cuvintele conţinut* (aliniere din română în engleză) este de fapt de aproximativ 90%.

	Unităţi lexicale	Cuvinte conţinut	Adnotate	%
SemCor2.0				
engleză	178499	85552	79595	93.03%
română	175603	88874	48392	54.45%
Engleză				
substantive	41007	41007	38799	94.61%
adjective	12885	12885	12313	95.56%
adverbe	7973	7973	7264	91.10%
verbe	20634	20634	19435	94.18%
numerale	2994	2994	1738	58.04%
abrevieri	59	59	46	77.96%
TOTAL	85552	85552	79595	93.03%
Română				
substantive	41652	41652	26666	64.02%
adjective	12314	12314	2078	16.87%
adverbe	9420	9420	3249	34.49%
verbe	21915	21915	14396	65.69%
numerale	3324	3324	1858	55.89%
abrevieri	249	249	145	58.23%
TOTAL	88874	88874	48392	54.45%

Tabela 2.4: Corpusul paralel englez-român SemCor2.0.

	Ocurenţe	Procent de transfer
Transfer reuşit		
corespondenţă directă (CD)	37816	42.55%
corespondenţă indirectă (CI)	4487	5.05%
entităţi: group, person, location	4231	4.76%
numerale	1858	2.09%
TOTAL	48392	54.45%
Transfer nereuşit		
alinieri nule	12814	14.42%
sinset incomplet (SSINC)	12044	13.55%
sinset neimplementat (SSNEI)	11930	13.42%
etichete morfosintactice diferite	3694	4.16%
TOTAL	40482	45.55%

Tabela 2.5: Situaţia transferului de sensuri în română.

limbi balcanice: bulgară, greacă, română, sârbă şi turcă. În plus, reţeaua semantică a limbii cehe creată în proiectul EuroWordNet, avea să se extindă.

Prima reţea semantică lexicală a fost cea a limbii engleze şi a fost dezvoltată de o echipă de cercetători de la universitatea Princeton din Statele Unite ale Americii coordonată de George Miller ([66]). A fost denumită sugestiv "WordNet"[55] pentru că simplificând lucrurile până la extrem, este în cele din urmă o reţea de (mulţimi de) cuvinte. WordNet adaugă încă trei principii la principiul de bază al semanticii lexicale oferind astfel o nouă viziune asupra organizării informaţiei semantice din lexiconul mental:

1. (semantica lexicală) există o corespondenţă de $a : b$, $a, b \geq 1$ între cuvinte şi înţelsuri (vezi figura 2.15);

2. (WordNet) *înţelesurile* cuvintelor sunt definite de serii sinonimice (sinseturi), adică de mulţimi de cuvinte care într-o serie anume au *sensuri* similare[56]. Distincţia înţeles/sens este aceeaşi deci cu cea dintre concept (general) şi interpretare/accepţiune (viziune particulară a conceptului). De exemplu, în DEX98 ([18]), conceptul de "*vehicul pe patru roţi propulsat de un motor cu ardere internă*" este exprimat prin: maşină/sens 3, autovehicul/sens 1 şi automobil/sens 1. Definiţiile sensurilor sunt:

   - **maşină(3)**: autovehicul, automobil.
   - **autovehicul(1)**: vehicul autopropulsat suspendat pe roţi, şenile sau tălpi de alunecare, care serveşte la transportul oamenilor sau al bunurilor.
   - **automobil(1)**: vehicul cu patru (rar, trei, şase) roţi pneumatice, mişcat de un motor cu explozie interna, cu aburi, cu electricitate sau aer comprimat.

   Cele trei sensuri sunt astfel plasate în sinsetul {*maşină*(3), *autovehicul*(1), *automobil*(1)} pentru a desemna înţelesul (conceptul) de "*vehicul pe patru roţi propulsat de un motor cu ardere internă*".

3. (WordNet) înţelesurile cuvintelor sunt în relaţie unele cu altele;

4. (WordNet) relaţiile conceptuale se diferenţiază în funcţie de categoriile gramaticale ale cuvintelor.

[55]Princeton WordNet 2.0 (PWN2.0) este versiunea 2.0 a implementării conceptului WordNet.

[56]Similaritatea se stabileşte pentru o clasă de contexte şi există dacă principiul substituţiei sinonimelor funcţionează pentru oricare element al sinsetului.

Înţelesuri	**Cuvinte**				
	C_1	C_2	C_3	$\ldots$	C_n
S_1	$E_{1,1}$	$E_{1,2}$			
S_2		$E_{2,2}$			
S_3		$E_{3,2}$	$E_{3,3}$		
$\vdots$				$\ddots$	
S_m					$E_{m,n}$

Figura 2.15: Matricea de corespondenţă între înţelesuri şi cuvinte.

Înainte de a descrie ROWN2.0, este util să fixăm o anumită terminologie care va fi folosită în această secţiune:

- *literal*: este un cuvânt component al unui sinset. În matricea lexicală din figura 2.15 un literal este oricare din cuvintele $C_1, C_2, \ldots, C_n$. De notat este faptul ca literalul este de fapt forma standard de dicţionar a unui cuvânt, deci lema acestuia.

- *sens* sau *înţeles*: în WordNet, spre deosebire de un dicţionar convenţional, putem pune semnul egal între sens şi înţeles pentru că orice sens al unui literal se identifică cu sinsetul din care acesta face parte (sinsetul defineşte un concept şi este o mulţime de sensuri similare, sensuri care la rândul lor se individualizează prin perechea literal, identificator de sens). Această egalitate elimină astfel deficienţa dicţionarelor convenţionale care exprimă un acelaşi înţeles prin definiţii (deci sensuri) diferite.

- *sinset* sau *concept*: o mulţime de literali în care fiecare literal este indexat de identificatorul său de sens. Din egalitatea sens, înţeles deducem că în WordNet putem identifica un concept şi prin termenii: înţeles, sinset, sens al unui literal.

- *ILI*: din englezescul "Inter-Lingual Index", este cheia unică cu care se indexează fiecare concept în WordNet[57]. Reţeaua lexicală semantică poate astfel fi vazută şi ca o tabelă a unei baze de date indexată după ILI.

[57]În secţiunea anterioară am făcut referire la termenii "(eticheta de) sens" sau "etichetă semantică". O astfel de etichetă este de fapt un ILI.

POS:	**n**
ILI:	**ENG20-02853224-n**
Synonyms:	*automobil*(1), *autovehicul*(1), *maşină*(4)
Definition:	Vehicul cu patru (rar trei, şase) roţi pneumatice, mişcat de un motor cu explozie internă, cu aburi, cu electricitate sau aer comprimat.

Figura 2.16: Conceptul de "*vehicul pe patru roţi propulsat de un motor cu ardere internă*" în ROWN2.0.

POS:	**n**
ILI:	**ENG20-02853224-n**
Synonyms:	*car*(1), *auto*(1), *automobile*(1), *machine*(4), *motorcar*(1)
Definition:	4-wheeled motor vehicle; usually propelled by an internal combustion engine.

Figura 2.17: Conceptul de "*vehicul pe patru roţi propulsat de un motor cu ardere internă*" în PWN2.0.

- *relaţie*: relaţiile se stabilesc între conceptele reţelei semantice lexicale. În funcţie de categoria gramaticală, există diverse relaţii care leagă între ele conceptele reţelei. De exemplu, pentru substantive între concepte se stabileşte o relaţie de tip **subsumare** (relaţia $\sqsubset$ din logicile descriptive) numită *hiperonimie*: $hypernym(a, b)$ indică faptul că a este hipernimul lui b, adică b moşteneşte toate proprietăţile lui a la care adaugă proprietăţi caracteristice pentru a se individualiza ca şi concept. Exemplu: $\{arbore(1), copac(1), pom(1)\}$ este hipernimul lui $\{stejar(1)\}$.

Prezentăm în figurile 2.16 şi 2.17 intrările corespunzătoare conceptului de "*vehicul pe patru roţi propulsat de un motor cu ardere internă*" din PWN2.0 şi ROWN2.0. Categoria gramaticală a literalilor din sinset este substantiv (**n**). Cheia unică a acestui concept este **ENG20-02853224-n** iar sinsetul corespunzător poate fi extras dacă interogăm PWN2.0 sau ROWN2.0 după această cheie. În ROWN2.0, sensul 1 al literalului *automobil*, sensul 1 al literalului *autovehicul* şi sensul numărul 4 al literalului *maşină* au toate aceeaşi definiţie: "Vehicul cu patru (rar trei, şase) roţi pneumatice, mişcat de un motor cu explozie internă, cu aburi, cu electricitate sau aer comprimat" (la fel se întâmplă bineînţeles şi în PWN2.0).

Înţelesurile cuvintelor sunt independente de limbă dar acest lucru nu garantează că în două limbi diferite conceptualizarea unei entităţi din universul de discurs se face în mod necesar la fel. Cu alte cuvinte, teoretic nu există înţelesuri *identice* dar pentru că oricine poate învăţa şi vorbi o limbă străină, există deci înţelesuri *similare* iar aceste similarităţi sunt independente de limbă. Două înţelesuri similare din limbi diferite *asociate aceluiaşi ILI*[58] formează un concept iar operaţia de asociere[59] le conferă acestora statutul de înţelesuri echivalente. După ce înţelesurile au fost asociate conceptul identificat de ILI-ul respectiv devine o generalizare a înţelesurilor din cele două limbi şi este astfel *egalul* acestora[60]. În consecinţă, aceeaşi cheie de identificare a sinseturilor în PWN2.0 şi ROWN2.0 arată faptul că vorbim despre acelaşi concept (înţeles). Astfel ILI reprezintă o codificare independentă de limbă a unui concept care se realizează (lexicalizează) diferit în română şi engleză (vezi figurile 2.16 şi 2.17).

O trăsătură importantă (poate cea mai importantă) a WordNet-ului este existenţa relaţiilor semantice între conceptele reţelei. Această structură ne îndreptăţeşte să privim WordNet-ul ca pe o *ontologie lexicală*, ontologie care specifică astfel o conceptualizare a structurii lexiconului mental. Există două tipuri de relaţii în WordNet ([66]):

- relaţii lexicale: apar între literali şi nu între sensurile lor. Exemple: sinonimia[61], antonimia şi relaţiile morfologice.
- relaţii semantice: apar între conceptele reţelei. Exemple: hiperonimia/hiponimia, meronimia/holonimia, ş.a.

Pentru că un sinset conţine o mulţime de literali care pot fi folosiţi interşanjabil într-o clasă de contexte, într-un sinset literalii sunt toţi de aceeaşi categorie gramaticală. În WordNet, fiecare categorie gramaticală grupează conceptele în structuri diferite cu proprietăţi diferite[62]. Unul din principiile de dezvoltare a reţelei semantice lexicale a limbii române ROWN2.0 a

[58]ILI este cheia care identifică înţelesul din engleză. Asocierea se face prin atribuirea aceluiaşi ILI înţelesului din română.

[59]Numim această operaţie aliniere conceptuală de unde noţiunea de reţele semantice aliniate la nivel de concept.

[60]Prin operaţia de asociere, cele două înţelesuri îşi împrumută unul altuia trăsături semantice astfel încât conceptul să reprezinte o descriere suficientă pentru identificarea oricăruia din ele.

[61]Este relaţia definitorie a conceptului în WordNet şi este o relaţie lexicală pentru că "apare între literalii unui sinset" ([66]). Totuşi sinsetul este o colecţie de sensuri similare ale literalilor componenţi, caz în care sinonimia trebuie să fie o relaţie semantică. Atribuim astfel sinonimiei calificativul de relaţie lexico-semantică.

[62]Există de asemenea relaţii care leagă sensuri ale unor literali care nu au aceeaşi categorie gramaticală dar acestea nu formează ierarhii de tipul celei a hipernimelor de exemplu.

fost acela de a conserva pe cât posibil structurile din PWN2.0 având dată alinierea conceptuală. Ca o consecinţă directă a acestui fapt, relaţiile din PWN2.0 pot fi grupate în două categorii: relaţii dependente de limbă (engleză) şi relaţii independente de limbă. Evident că relaţiile semantice sunt independente de limbă şi de aceea pot fi transferate automat în română (vezi figurile 2.18 şi 2.19 pentru o aliniere structurală automată – vizualizări cu VisDic ([34])). Antonimia a fost şi ea parţial transferată în română dar cu verificarea perechilor de antonime rezultate.

În cele ce urmează, vom da o descriere sumară a relaţiilor din PWN2.0 care au fost transferate automat în ROWN2.0 (vezi tabelul 2.6). Considerăm că C este mulţimea conceptelor din PWN2.0 şi ştim de asemenea că fiecare concept are asociat un ILI unic, $i \in I$[63]. Există deci o funcţie bijectivă f, $f : C \rightarrow I, f(c) = i$ astfel încât putem identifica un concept prin ILI-ul său asociat. Relaţiile semantice şi lexicale din PWN2.0 sunt relaţii binare R definite pe $I \times I$ care au proprietăţi specifice (desemnăm prin $R(i_1, i_2)$ sau $i_1 \stackrel{R}{\rightarrow} i_2$ o pereche din relaţia R). Ele sunt:

- relaţia *hypernym*: se aplică substantivelor şi verbelor. $hypernym(i_1, i_2)$ exprimă faptul că i_1 este hipernimul lui i_2 adică i_2 are toate proprietăţile lui i_1 la care adaugă proprietăţi caracteristice pentru a se individualiza ca şi concept. De exemplu, $stilou(1)$ este hipernimul lui $pix(1)$ în ROWN2.0 (figura 2.19). Relaţia inversă relaţiei *hypernym* este *hyponym* şi se defineşte astfel:

  $$hyponym(i_2, i_1) \Leftrightarrow hypernym(i_1, i_2)$$

  Ambele relaţii sunt asimetrice, ireflexive şi tranzitive.

- relaţia *holonym*: leagă substantive şi este relaţia de tip "*alcătuit(ă) din*". Relaţia inversă, *meronym*, este relaţia de tip "*parte/porţiune/ membru din*". Între cele două relaţii există echivalenţa

  $$meronym(i_2, i_1) \Leftrightarrow holonym(i_1, i_2)$$

  Ambele relaţii sunt asimetrice şi parţial tranzitive (vezi [105, pag. 118] pentru detalii). Exemple din PWN2.0:

  $$engine(1) \stackrel{holo\_part}{\longrightarrow} camshaft(1) \Leftrightarrow camshaft(1) \stackrel{mero\_part}{\longrightarrow} engine(1)$$
  $$timber(1) \stackrel{holo\_portion}{\longrightarrow} wood(1) \Leftrightarrow wood(1) \stackrel{mero\_portion}{\longrightarrow} timber(1)$$
  $$forest(1) \stackrel{holo\_member}{\longrightarrow} tree(1) \Leftrightarrow tree(1) \stackrel{mero\_member}{\longrightarrow} forest(1)$$

[63]Pentru că literalii unui sinset au toţi aceeaşi categorie gramaticală, ILI-ul conţine de asemenea şi această informaţie. Considerăm următoarele notaţii pentru categoriile gramaticale: substantiv N, verb V, adjectiv A şi adverb R.

- relaţia *subevent*: se stabileşte între verbe. $subevent(i_1, i_2)$ denotă faptul că intervalul de timp în care se desfăşoară evenimentul propriu conceptului i_1 este inclus în intervalul de timp în care are loc evenimentul i_2: $subevent(dream(2), sleep(1))$.

- relaţia *causes*: apare între verbe. $causes(i_1, i_2)$ asignează conceptului i_1 cauza iar conceptului i_2 efectul: $causes(kill(1), die(1))$.

- relaţia *verb_group*: se stabileşte de asemenea între verbe[64] şi grupează câteva înţelesuri similare într-o clasă: $verb\_group(i_1, i_2)$ şi perechea $verb\_group(i_1, i_3)$ plasează conceptele $i_{1,2,3}$ în aceeaşi mulţime de înţelesuri[65] iar i_1 este înţelesul reprezentativ al clasei[66].

- relaţia *be_in_state*: categoriile gramaticale implicate sunt substantivul şi adjectivul iar relaţia specifică ce valoare are o anumită proprietate: $be\_in\_state(stature(2), tall(1))$ atribuie valoarea adjectivului "*tall*" proprietăţii exprimate de substantivul "*stature*".

- relaţia *similar_to*: se verifică între adjective şi ca *verb_group* grupează nişte înţelesuri într-o clasă de similaritate, clasă care conţine un înţeles reprezentativ. Relaţia de antonimie a înţelesului reprezentativ cu un alt înţeles este preluată astfel indirect şi de membrii clasei.

- relaţia *also_see*: apare între categoriile gramaticale din tabelul 2.6 şi este o relaţie care indică o legătură semantică (specifică dicţionarelor convenţionale) între înţelesurile din PWN2.0. Este o relaţie simetrică: $also\_see(tall(1), high(2)) \Leftrightarrow also\_see(high(2), tall(1))$.

- relaţia *category_domain*: este o relaţie care clasifică un înţeles din PWN2.0 în termenii altui înţeles din PWN2.0. Apare între categoriile gramaticale din tabelul 2.6 iar categoria este dată întotdeauna de un sinset de substantive. De exemplu, tancul este un vehicul specific armatei: $category\_domain(tank(1), military(1))$.

În afară de relaţii, din PWN2.0 s-au mai transferat automat corespondenţele sinseturilor cu conceptele ontologiei SUMO ([75, 76]) şi cu domeniile IRST ([56]). Astfel, fiecare concept din PWN2.0 are asociat unul sau mai multe concepte SUMO şi unul sau mai multe domenii IRST. Asocierea cu conceptele SUMO se face la nivel de sinonimie, hipernimie sau instanţiere (vezi [76]):

[64] Între ILI-uri de verbe.

[65] $i_{1,2,3}$ nu sunt sinonime totuşi pentru că altfel am fi avut un singur concept în loc de trei.

[66] În engleză acesta se numeşte "*head*".

Relaţie	Categorii gramaticale	Transfer?
hypernym	$\langle N, N\rangle$; $\langle V, V\rangle$	da
holo_part	$\langle N, N\rangle$	da
holo_portion	$\langle N, N\rangle$	da
holo_member	$\langle N, N\rangle$	da
subevent	$\langle V, V\rangle$	da
causes	$\langle V, V\rangle$	da
verb_group	$\langle V, V\rangle$	da
be_in_state	$\langle A, N\rangle$	da
similar_to	$\langle A, A\rangle$	da
also_see	$\langle V, V\rangle$; $\langle A, A\rangle$	da
category_domain	$\langle N, N\rangle$; $\langle V, N\rangle$; $\langle A, N\rangle$; $\langle R, N\rangle$	da
near_antonym	$\langle N, N\rangle$; $\langle V, V\rangle$; $\langle A, A\rangle$; $\langle R, R\rangle$	da cu restricţii
derived	$\langle A, A\rangle$; $\langle R, A\rangle$; $\langle A, N\rangle$	parţial

Tabela 2.6: Relaţii transferate automat din PWN2.0 în ROWN2.0 (tabel din [105]).

- sinonimie (=): {*animal*(1), *beast*(1), *brute*(2), *creature*(1), ...} este asociat cu conceptul SUMO sinonim *Animal*;
- hipernimie (+): {*measure*(3), *quantity*(1), *amount*(3)} are asociat conceptul SUMO hipernim *ConstantQuantity*;
- instanţiere (@): {*President_of_the_United_States*(1)} este o instanţă a conceptului SUMO *Position*.

Aceste asocieri ale sinseturilor din PWN2.0 cu concepte SUMO sau domenii IRST ne oferă inventare de sens alternative la inventarul dat de ILI. Pentru fiecare ILI $i \in I$, funcţia $sumo(i)$ ne dă conceptul SUMO asignat conceptului i iar $dom(i)$ ne dă domeniul IRST asignat aceluiaşi concept. Funcţiile *sumo* şi *dom* sunt surjective dar nu sunt injective. Această proprietate a lor face ca noile etichete de sens să grupeze conceptele PWN2.0 în mulţimi de înţelesuri, lucru care reduce dimensiunea inventarului de sens uşurând astfel sarcina programului de DSA (vezi capitolele următoare).

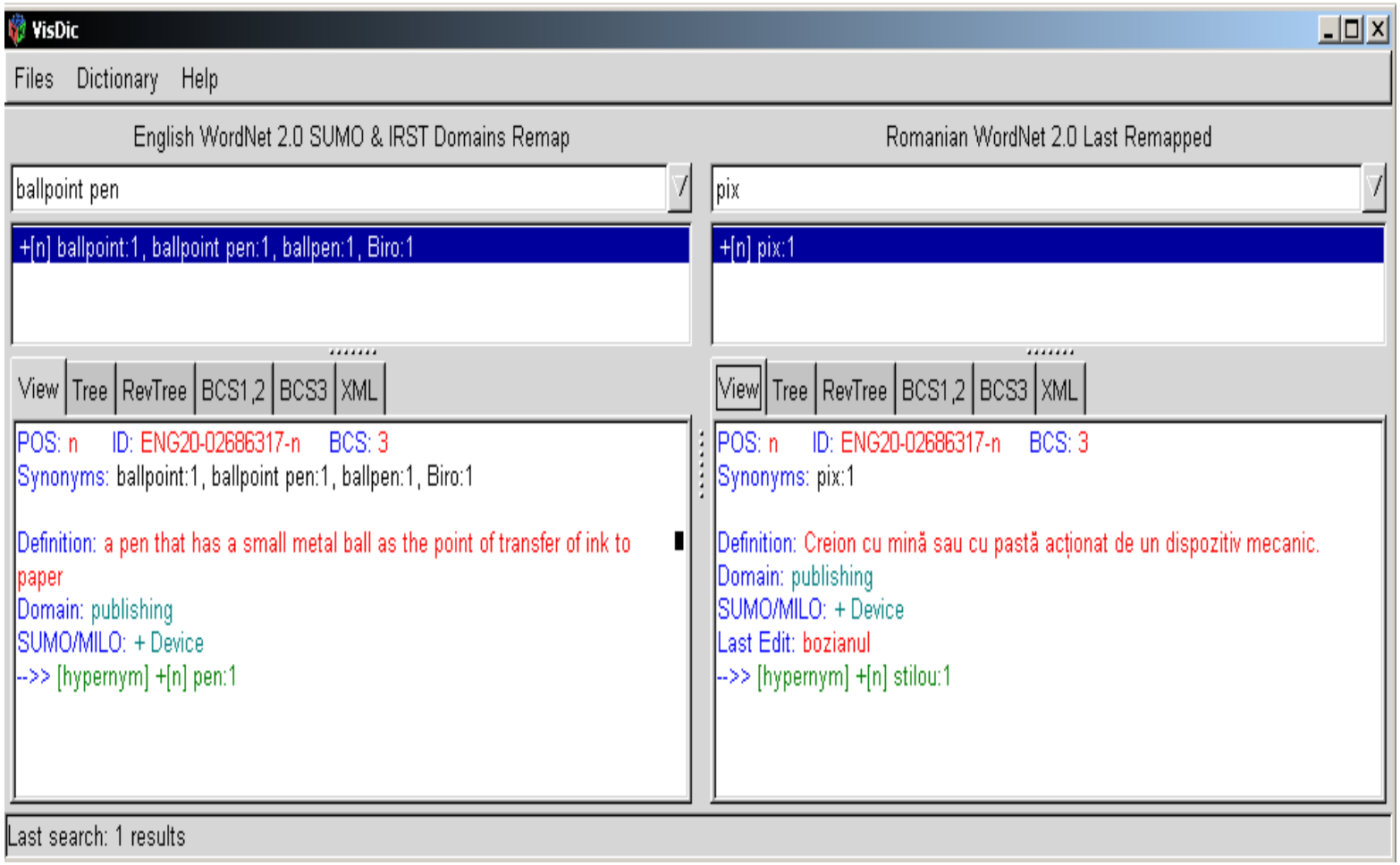

Figura 2.18: Alinierea înţelesurilor de "*pix* - instrument de scris" şi "*ballpoint pen*".

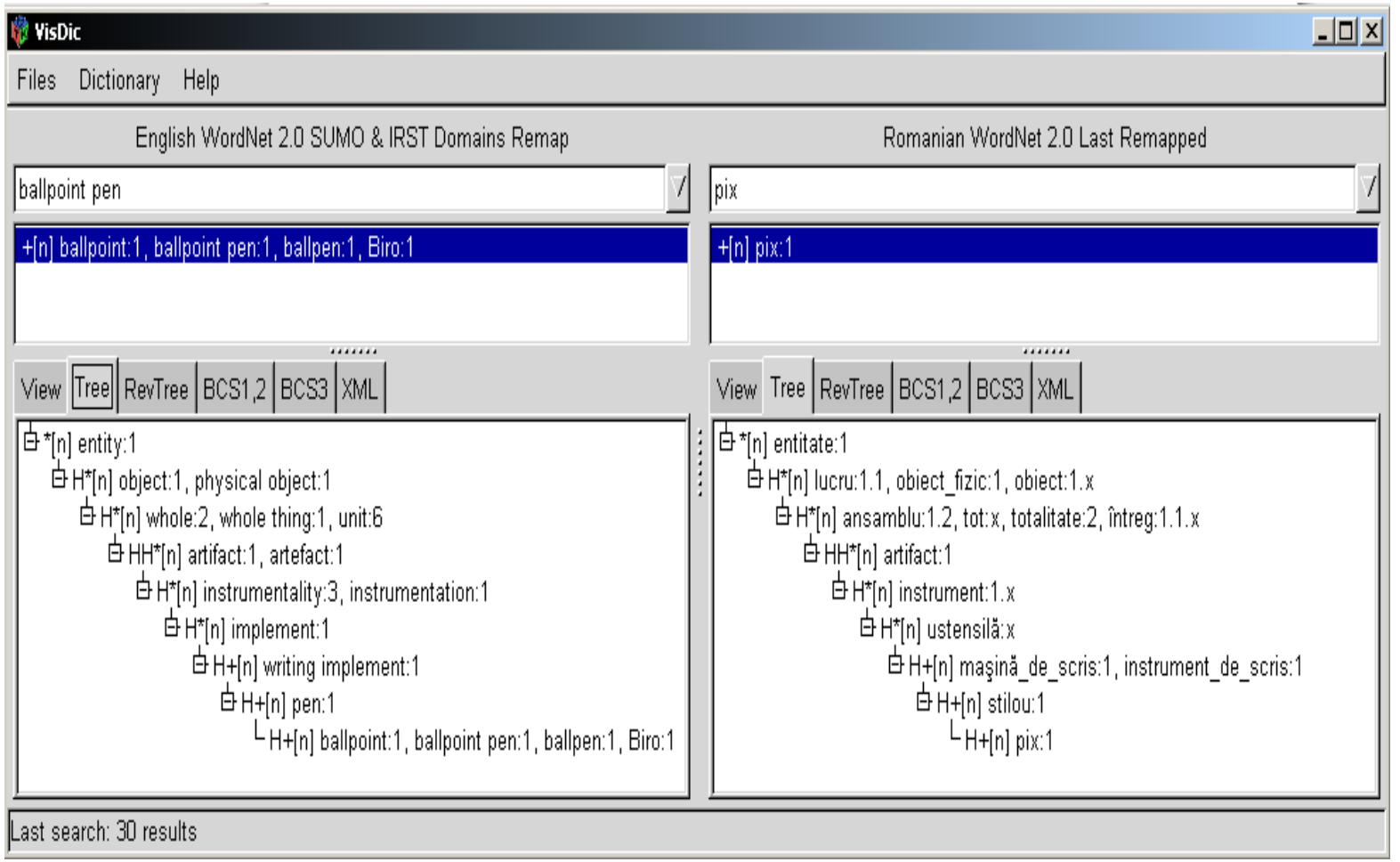

Figura 2.19: Echivalenţa conceptuală a arborilor de hipernimi pentru conceptul *pix*(1).

Capitolul 3

DSA pe texte paralele

Dezambiguizarea semantică automată a fost o problemă a cărei soluţie s-a căutat în mod tradiţional experimentând-se pe texte simple. Textele paralele[1], reprezintă colecţii de traduceri (le numim texte ţintă) ale unor texte (sursă) în una sau mai multe limbi, traduceri care oferă o nouă dimensiune noţiunii de context de apariţie al unui cuvânt. Pentru că o traducere conservă înţelesul textului sursă, principiul compoziţionalităţii înţelesului ne permite să apreciem că la nivel de cuvânt, perechea cuvânt sursă, cuvânt ţintă reduce ambiguitatea de înţeles în ambele direcţii (sursă-ţintă şi ţintă-sursă). Contextul de apariţie al cuvântului sursă este materializat prin însăşi traducerea cuvântului în limba ţintă (contextul de apariţie determină înţelesul cuvântului sursă iar înţelesuri diferite se traduc diferit[2]).

Dezambiguizarea semantică automată pe texte paralele a fost subiectul cercetărilor din [9, 17, 20]. Primele două lucrări se ocupă de generarea traducerii corecte în engleză a unui cuvânt într-o limbă sursă dată cum ar fi limba germană sau ebraică ([17]) sau limba franceză ([9]). Această problemă este tot una de DSA în care "inventarul de sensuri" este compus din traducerile posibile ale unui cuvânt[3]. A treia lucrare ([20]) foloseşte traducerile pentru a determina înţelesurile cuvintelor ţintă (în franceză) din inventarul de sensuri al limbii sursă (engleza). Prezentăm pe scurt algoritmul SALAAM din [20] pentru că posedă anumite similarităţi cu algoritmul nostru **WSDTool** ([119]) care va fi descris în acest capitol.

SALAAM atribuie etichete de sens cuvintelor în franceză dintr-un inventar de sensuri pentru engleză (Collins Cobuild English Dictionary, [95]). Paşii de dezambiguizare sunt următorii:

[1] Sau corpusuri paralele.

[2] Evident, nu în mod necesar.

[3] Este deci un dicţionar de traducere.

1. alinierea lexicală a bitextului englez-francez şi selecţia cuvintelor franţuzeşti de dezambiguizat;

2. pentru fiecare cuvânt de dezambiguizat w_k^{fr}, prin alinierea lexicală, obţine o mulţime E de echivalenţi de traducere ai lui w_k^{fr} din întreg corpusul;

3. se defineşte o măsură de similaritate sim (vezi [52] pentru definiţia acesteia) între textele definiţiilor sensurilor cuvintelor $w_j^{en} \in E$. Dacă notăm definiţia sensului n al cuvântului w^{en} cu $d_n(w^{en})$, această măsură, $sim(d_a(w_i^{en}), d_b(w_j^{en}))$, trebuie să fie maximă pentru a se selecta sensurile a respectiv b ale cuvintelor englezeşti w_i^{en} şi w_j^{en}, $\forall w_i^{en}, w_j^{en} \in E$. Astfel, un cuvânt $w_i^{en} \in E$ primeşte eticheta de sens a dacă şi numai dacă $d_a(w_i^{en})$ are o similaritate maximă cu restul definiţiilor sensurilor atribuite ale cuvintelor din E;

4. fiecare ocurenţă a cuvântului w_k^{fr} în textul francez primeşte eticheta de sens $w_j^{en}(a)$ unde $w_j^{en} \in E$ este echivalentul de traducere folosit în unitatea de traducere respectivă.

În ce priveşte pasul 3 de mai sus, decizia de a eticheta toţi echivalenţii de traducere englezeşti ai unui cuvânt francez cu sensuri similare presupune că un cuvânt în franceză apare în corpus într-un spectru semantic îngust, lucru care este adevărat numai în cazul corpusurilor nebalansate. Altfel spus, tacit, se adoptă o ipoteză în spiritul celei a "unui sens pe discurs" (Yarowsky, [130]).

WSDTool ([119]) este un algoritm de dezambiguizare semantică automată pe texte paralele care se bazează pe existenţa reţelelor semantice lexicale aliniate la nivel de concept. Aceste reţele asigură o reprezentare independentă de limbă a înţelesurilor, lucru care oferă o etichetare cu sensuri uniformă pentru orice cuvânt al corpusului paralel. Pe de altă parte, structurile de relaţii între conceptele reţelei favorizează definirea de măsuri de similaritate între înţelesuri mult mai exacte decât cele de tip Lesk. În acest capitol vom descrie algoritmul WSDTool împreună cu aliniatorul lexical **YAWA** ([118]) de care WSDTool are nevoie pentru a găsi perechile de echivalenţi de traducere (etapă identică cu pasul 1 al lui SALAAM).

3.1 Aliniatorul lexical YAWA

YAWA[4] ([118]) este un program de aliniere lexicală care pentru două fraze[5] în limbi diferite ale unui corpus paralel precizează la nivel de unitate lexicală

[4]Yet Another (simple) Word Aligner.

[5]Sau fragmente de text.

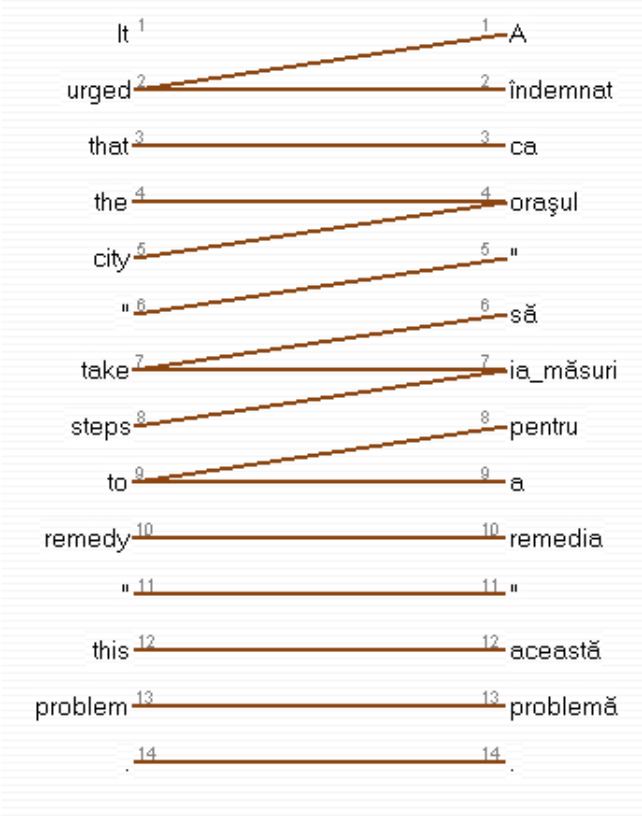

Figura 3.1: Exemplu de aliniere lexicală între o frază în engleză şi traducerea acesteia în română.

care sunt echivalenţele de traducere între ele. Frazele în limbile l_1 şi l_2 apar ca două mulţimi de unităţi lexicale împreună cu poziţia lor în frază, S_{l_1} şi S_{l_2} iar echivalenţele de traducere se pot descrie ca nişte corespondenţe între elemente din S_{l_1} şi S_{l_2}. În concluzie, alinierea lexicală a frazelor S_{l_1} şi S_{l_2} este o mulţime $\mathcal{A}$ de corespondenţe $w^i_{l_1} \leftrightarrow w^j_{l_2}$ (sau de perechi $\langle w^i_{l_1}, w^j_{l_2} \rangle$) cu $w^i_{l_1} \in S_{l_1}$ şi $w^j_{l_2} \in S_{l_2}$[6]. În figura 3.1 avem un exemplu de aliniere lexicală între două fraze, una în engleză (sursa) şi cea de-a doua în română (ţinta). Mulţimea $\mathcal{A}$ conţine în acest caz următoarele perechi:

$$\{\langle It^1_{en}, \emptyset \rangle, \langle urged^2_{en}, A^1_{ro} \rangle, \langle urged^2_{en}, indemnat^2_{ro} \rangle, \ldots\}$$

Pentru a putea alinia un corpus paralel cu YAWA, acesta are nevoie de preprocesare: segmentare la nivel de cuvânt, adnotare cu etichete morfo-sintactice compatibile MULTEXT-East ([21]) şi lematizare. După preprocesare, o frază în limba L a corpusului paralel este o mulţime S_L de tupluri

[6]Dacă un cuvânt $w^i_{l_1}$ nu se traduce în limba ţintă, acest lucru se reprezintă prin perechea $\langle w^i_{l_1}, \emptyset \rangle$. De asemenea dacă $w^j_{l_2}$ este inserat în traducerea în l_2, perechea $\langle \emptyset, w^j_{l_2} \rangle$ va fi adăugată la mulţimea de alinieri.

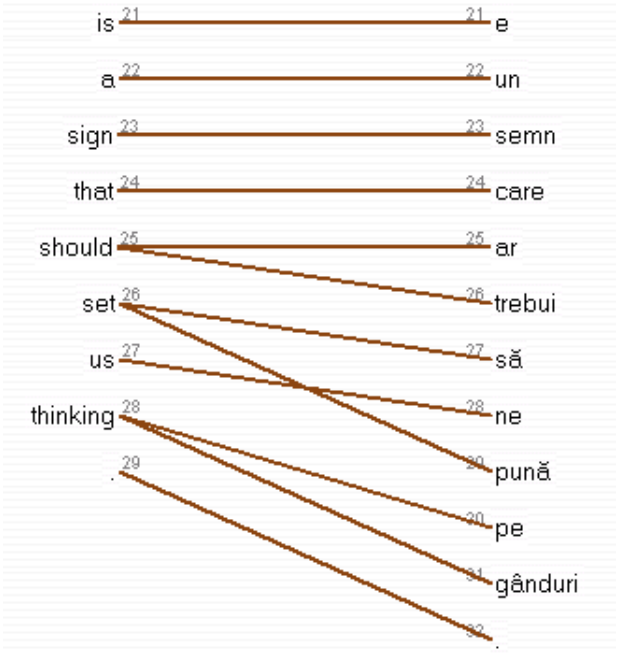

Figura 3.2: Exemplu de aliniere lexicală între două cuvinte de categorii gramaticale diferite: "*thinking*" şi "*gânduri*".

formă ocurentă, etichetă morfosintactică şi lemă a cuvântului în frază de forma $\langle w_L^i, t_L^i, l_L^i \rangle$ unde i este poziţia cuvântului în frază. Pe lângă aceste preprocesări, YAWA mai foloseşte de asemenea adnotarea cu metacategorii. O metacategorie este o clasă de etichete morfosintactice (vezi anexa A pentru lista exhaustivă a metacategoriilor pentru română şi engleză) identificată printr-un număr întreg care permite ca alinierea intercategorială să aibă loc[7]. De exemplu, în figura 3.2, "*thinking*" este un verb la gerunziu (`Vmg`) iar "*gânduri*" este un substantiv, plural, nearticulat (`Ncfs-n`). Dacă aceste etichete morfosintactice sunt puse în corespondenţă prin aceeaşi metacategorie (`1`), alinierea devine posibilă.

În versiunea sa curentă, YAWA aliniază perechile de limbi română şi engleză (în direcţia română-engleză[8]) şi este un aliniator lexical în patru faze. Fiecare fază asigură un schelet de aliniere pe care se va construi alinierea din următoarea fază. YAWA adaugă alinieri la fiecare fază (cu excepţia ultimei

[7]YAWA aliniază cuvintele considerând eticheta morfosintactică a acestora. Cel mai restrictiv caz este acela în care cuvintele au aceeaşi etichetă dar cum alinieri intercategoriale există, metacategoriile relaxează condiţia de egalitate nepermiţând totuşi orice combinaţie de etichete la aliniere (de exemplu, un adverb nu se va alinia niciodată cu un pronume).

[8]Pentru un alt program de aliniere pe aceeaşi direcţie, vezi [103].

faze) cu scopul de a mări recall-ul alinierii globale fără a deteriora (prea mult) precizia acesteia. În cele ce urmează considerăm că YAWA aliniază două fraze S_{ro} şi S_{en} de tupluri $\langle w^i_{ro}, c^i_{ro}, t^i_{ro}, l^i_{ro}\rangle \in S_{ro}$ şi $\langle w^j_{en}, c^j_{en}, t^j_{en}, l^j_{en}\rangle \in S_{en}$ de forma cuvânt (w), metacategorie (c), etichetă morfosintactică (t) şi lemă (l). De asemenea, rezultatele intermediare ale fazelor se notează $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ şi $\mathcal{A}_4$ care este şi rezultatul final ($\mathcal{A}$) şi care conţin corespondenţe de tupluri (română-engleză) de tipul celor de mai sus.

3.1.1 Faza 1

Faza 1 este faza cea mai importantă a alinierii pentru că pe scheletul de aliniere construit aici se vor genera alinierile următoare. În această fază YAWA aliniază 1:1 numai cuvintele cu metacategoriile `1`, `8`, `14`, `100`[9] (vezi anexa A). Alinierea se face la nivel de lemă printr-un dicţionar de echivalenţi de traducere extras automat[10] ([102, 104]) din corpusuri paralele şi augmentat cu dicţionarul extras din ROWN2.0 şi PWN2.0[11]. Pe lângă dicţionarul de echivalenţi de traducere, oricare două leme pentru care $cogn(l^i_{ro}, l^j_{en}) \geq 0.45$[12] sunt aliniate dacă au aceeaşi metacategorie. Dacă $cogn(l^i_{ro}, l^j_{en}) \geq 0.90$[13], cele două leme sunt aliniate indiferent de metacategoria pe care o are fiecare. Algoritmul de aliniere în această fază este următorul:

1. pentru fiecare $l^i_{ro} \in S_{ro}$ ($c^i_{ro} \in \{1, 8, 14, 100\}$), determină cu ajutorul dicţionarului şi a funcţiei de similaritate *cogn*, poziţiile din S_{en} pe care se află echivalenţii de traducere pentru l^i_{ro} şi alcătuieşte lista B^i_{en} de tupluri $\langle l^i_{ro}, l^j_{en}, s_{i,j}\rangle$ unde $s_{i,j}$ este scorul perechii extras din dicţionar (sau dacă e vorba de leme similare din punct de vedere ortografic, $s_{i,j} = 100 \cdot cogn(l^i_{ro}, l^j_{en})$ pentru a se obţine scoruri în intervalul $[45, 100]$);

[9] Se aliniază intercategorial substantivele comune, verbele şi adjectivele (`1`), adverbele (`14`), substantivele proprii (`8`) şi cuvintele cu categorie gramaticală necunoscută (`100`).

[10] Formatul acestui dicţionar este $\langle l_{ro}, l_{en}, c, scor\rangle$ unde l_{ro}, l_{en} este perechea de echivalenţi de traducere, c este metacategoria lemelor iar *scor* este un scor care indică măsura în care programul de extracţie "crede" că perechea este una de echivalenţi de traducere.

[11] Acest dicţionar se extrage prin generarea tuturor perechilor din două sinseturi cu acelaşi ILI. Scorul de echivalenţă de traducere este unul foarte mare (10000) pentru că o astfel de pereche este sigură.

[12] Limba română a împrumutat cuvinte din engleză şi din acest motiv, forma ortografică a acestor cuvinte este similară cu cea din engleză. Acest lucru este un indicator foarte puternic pentru echivalenţa de traducere. *cogn* este o funcţie de similaritate între şiruri de caractere cu valori în intervalul $[0, 1]$ iar pentru calculul acestei funcţii, YAWA foloseşte pachetul Perl `String::Similarity` (`http://search.cpan.org/~mlehmann/String-Similarity-1.02/Similarity.pm`) care implementează algoritmul din [72]. Cititorul poate consulta de asemenea [102] pentru detalii.

[13] Cele două praguri de similaritate 0.45 şi 0.90 au fost stabilite experimental.

```
Det  -> ( '<TS>'|'<DM>'|'<DMS>'|'<DMP>'|'<PSS>'|'<PS>'|'<PSP>'|'<PI>'|
          '<PZ>'|'<RELQ>' )
Adje -> ( '<ADJE>' )
Adve -> ( '<ADVE>' )
Noun -> ( '<NN>'|'<NNP>'|'<NNPS>'|'<NNS>'|'<NNSY>'|'<NNY>'|'<CD>'|'Y' )
Prep -> ( '<PREP>' )

Mod -> ( Adve* Adje+ )
Np  -> Det* Mod* Noun+
Pp  -> ( Prep+ Np )
```

Figura 3.3: Gramatică pentru recunoaşterea grupurilor nominale şi prepoziţionale (tipice) în engleză.

2. din produsul cartezian $\bigotimes_i B^i_{en}$ se extrag alinieri 1:1 G_k iar dintre acestea se alege ca aliniere finală cea pentru care $\sum_{\langle l^i_{ro}, l^j_{en}, s_{i,j}\rangle \in G_k} |i-j|$ este minimă iar $\sum_{\langle l^i_{ro}, l^j_{en}, s_{i,j}\rangle \in G_k} s_{i,j}$ este maximă. Cu alte cuvinte se presupune că ordinea cuvintelor se păstrează în traducere (suma modulelor diferenţelor poziţiilor cuvintelor este minimă) şi se alege alinierea a cărei sumă a scorurilor de traducere este maximă.

3.1.2 Faza 2

Această fază cere o preprocesare suplimentară a corpusului paralel atât pentru limba română cât şi pentru engleză. Este vorba de recunoaşterea grupurilor sintactice nominale (`Np`) şi prepoziţionale (`Pp`) nerecursive cât şi a complecşilor verbali (`Vp`) şi adjectivali (`Ap`). Aceste grupuri sunt recunoscute cu ajutorul expresiilor regulate definite peste secvenţe de etichete morfosintactice. De exemplu expresia `/<TSR>(<NSRN>|<NSN>)<ASN>/` recunoaşte un grup nominal de tipul "*o*/`TSR` *fată*/`NSRN` *frumoasă*/`ASN`" sau "*un*/`TSR` *băiat*/`NSN` *curajos*/`ASN`" iar expresia regulată `/<TSR>?<R><ASN>/` recunoaşte complecşi adjectivali de tipul "*cel*/`TSR` *mai*/`R` *complicat*/`ASN`" sau "*foarte*/`R` *complicat*/`ASN`". Se foloseşte o gramatică similară cu cea cu care TTL recunoaşte entităţile denumite (vezi figura 2.1) ale cărei reguli se transformă automat în expresii regulate Perl. De exemplu, în figura 3.3, neterminalul `Mod` generează expresia regulată Perl `/(<ADVE>)*(<ADJE>)+/` (în figura 3.4 se află un extras din formatarea XML a corpusului paralel în care grupurile sunt adnotate cu atributul `chunk`).

```
- <tu id="50">
  - <seg lang="en">
    - <s id="br-a01.44.50.en">
        <w lemma="Caldwell" ana="8+,Np" chunk="Np#1" wns="ili:ENG20-00006026-n">Caldwell</w>
        <w lemma="'s" ana="21+,St" chunk="Np#1" head="0">'s</w>
        <w lemma="resignation" ana="1+,Ncns" chunk="Np#1" wns="ili:ENG20-06109386-n" head="0">resignation</w>
        <w lemma="have" ana="3+,Vais" chunk="Vp#1" head="5">had</w>
        <w lemma="be" ana="3+,Vaps" chunk="Vp#1" head="5">been</w>
        <w lemma="expect" ana="1+,Vmps" chunk="Vp#1,Ap#1" wns="ili:ENG20-00695861-v" head="2">expected</w>
        <w lemma="for" ana="5+,Sp" chunk="Pp#1" head="8">for</w>
        <w lemma="some" ana="22+,Di3" chunk="Pp#1,Np#2" head="8">some</w>
        <w lemma="time" ana="1+,Ncns" chunk="Pp#1,Np#2" wns="ili:ENG20-14265546-n" head="5">time</w>
        <c>.</c>
      </s>
    </seg>
  - <seg lang="ro">
    - <s id="br-a01.44.50.ro">
        <w lemma="demisie" ana="1+,Ncfsry" chunk="Np#1">Demisia</w>
        <w lemma="lui" ana="21+,Tf-so" chunk="Np#1" head="2">lui</w>
        <w lemma="Caldwell" ana="8+,Np" chunk="Np#1" wns="ili:ENG20-00006026-n" head="0">Caldwell</w>
        <w lemma="fi" ana="3+,Vail3s" chunk="Vp#1" head="4">fusese</w>
        <w lemma="aştepta" ana="1+,Vmp--sf" chunk="Vp#1,Ap#1" wns="ili:ENG20-00695861-v" head="0">aşteptată</w>
        <w lemma="de" ana="5+,Spsa" chunk="Pp#1" head="7">de</w>
        <w lemma="ceva" ana="22+,Di3-sr---e" chunk="Pp#1,Np#2" head="7">ceva</w>
        <w lemma="timp" ana="1+,Ncms-n" chunk="Pp#1,Np#2" wns="ili:ENG20-14265546-n" head="4">timp</w>
        <c>.</c>
      </s>
    </seg>
  </tu>
```


Figura 3.4: Exemplu de codificare XML din corpusul paralel SemCor2.0.

În faza 2 a algoritmului cu ajutorul mulţimii $\mathcal{A}_1$ se aliniază 1:1 grupurile/complecşii *de acelaşi tip* ($\texttt{Np}_{ro} \leftrightarrow \texttt{Np}_{en}$, etc.) din română şi engleză astfel: dacă o submulţime $a_1^k \subset \mathcal{A}_1$ conţine alinieri între poziţii conţinute de grupuri de acelaşi tip, aliniază grupurile respective. De exemplu, în figura 3.4 dacă mulţimea a_1^1 ar fi conţinut alinierea $\langle Caldwell_{ro}^3, 8_{ro}^3, Np_{ro}^3, Caldwell_{ro}^3\rangle \leftrightarrow \langle Caldwell_{en}^1, 8_{en}^1, Np_{en}^1, Caldwell_{en}^1\rangle$ atunci am fi putut alinia grupurile nominale $\texttt{Np\#1}_{ro}$ şi $\texttt{Np\#1}_{en}$ întrucât indexul 3 este conţinut de grupul $\texttt{Np\#1}_{ro}$ (limite 1,3) iar indexul 1 este conţinut la rândul său de grupul $\texttt{Np\#1}_{en}$ (limite 1,3).

După ce s-au aliniat grupurile, cuvintele componente trebuie şi ele aliniate. În acest punct intră în funcţiune un modul de aliniere bazat pe reguli care este dependent de perechea de limbi. Pentru fiecare pereche de grupuri aliniate, g_{ro}^a şi g_{en}^b cuvintele componente se aliniază în felul următor:

1. indiferent de tipul grupurilor, dacă în acestea există un număr egal de cuvinte cu aceeaşi metacategorie, aliniază 1:1 cuvintele de aceeaşi metacategorie în ordinea apariţiei lor. În exemplul nostru din figura 3.4, grupul nominal $\texttt{Np\#1}_{ro}$ conţine cuvintele *demisia*, *lui*, *Caldwell* cu metacategoriile 1, 21, 8 iar grupul nominal $\texttt{Np\#1}_{en}$ conţine cuvintele *Caldwell*, *'s*, *resignation* cu metacategoriile 8, 21, 1. În acest caz, cuvintele se aliniză 1:1 pentru că avem acelaşi număr de metagorii (8 apare o dată în engleză şi în română, etc.);

2. după ce se aplică pasul 1, pentru cuvintele rămase nealiniate se caută alinieri care să respecte anumite reguli de traducere. De exemplu, în compecşii verbali aliniaţi verbele auxiliare se aliniază 1:1 sau 1:2 (vezi aceeaşi figură 3.4, grupurile $\texttt{Vp\#1}_{ro}$ şi $\texttt{Vp\#1}_{en}$) sau dacă în română nu avem verb auxiliar, auxiliarul din engleză se aliniază la verbul predicativ românesc. De asemenea, marcajele de mod conjunctiv (*să*) şi condiţional-optativ (*aş*, *ai*, etc.) se aliniază prin convenţie pe verbul predicativ/modal englezesc (vezi figura 3.2).

Pentru că YAWA a fost dezvoltat într-un timp relativ scurt (aproximativ două săptămâni) regulile de aliniere în faza 2 nu au fost descrise separat într-un fişier ci au fost incorporate într-un plugin[14] Perl care este folosit de YAWA pentru a genera alinierile specifice perechii de limbi. Din acest motiv nu putem da aici o listă de reguli de traducere pentru că acestea sunt codificate în plugin[15].

[14]Un modul cu o interfaţă anume care îndeplineşte o funcţie a unui program şi care poate fi detaşat/înlocuit foarte uşor fără a modifca programul.

[15]Următoarea dezvoltare a acestui aliniator va include o descriere formală a regulilor de traducere care vor fi incluse într-un fişier separat.

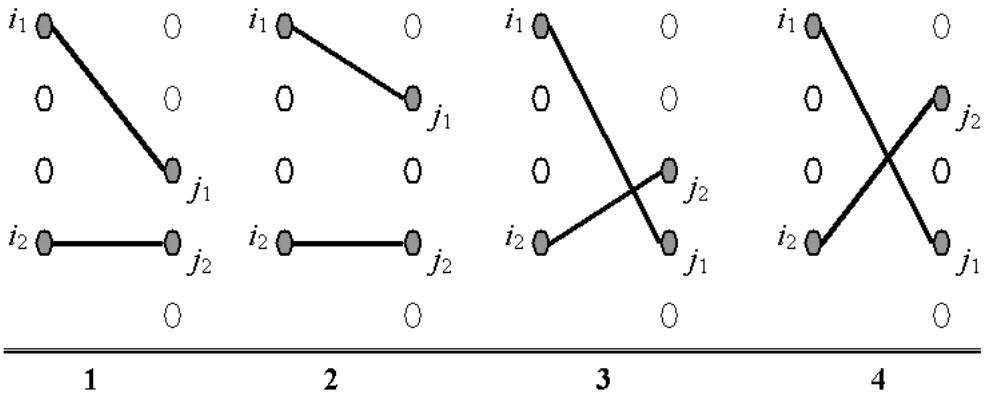

Figura 3.5: Situaţii posibile în alinierea de blocuri.

Algoritmul fazei 2 de aliniere este rulat în mod repetat până când la mulţimea $\mathcal{A}_2$ nu se mai adaugă nicio aliniere nouă. În acest moment se reunesc cele două mulţimi $\mathcal{A}_1$ şi $\mathcal{A}_2$ iar rezultatul se depune în $\mathcal{A}_2$ care este rezultatul alinierii fazelor 1 şi 2.

3.1.3 Fazele 3 şi 4

În faza 3 se încearcă alinierea euristică a secvenţelor de cuvinte consecutive (blocuri) rămase nealiniate. Întâi se caută corespondenţa blocurilor întocmai ca în faza 2 când se aliniau grupurile sintactice. Un bloc în română sau engleză este determinat de două cuvinte care sunt deja aliniate. Fie i_1 şi i_2 ($i_1 < i_2, i_2 - i_1 > 1$) poziţiile acestor cuvinte în română iar j_1 şi j_2 poziţiile cuvintelor din engleză care se aliniază la cele româneşti. Avem patru cazuri (vezi figura 3.5):

1. $j_1 < j_2, j_2 - j_1 \in \{0, 1\}$; în acest caz blocul englesc corespunzător este vid iar blocul românesc rămâne nealiniat;

2. $j_1 < j_2, j_2 - j_1 > 1$; în acest caz blocul englesc corespunzător se aliniază cu cel românesc;

3. $j_1 > j_2, j_1 - j_2 \in \{0, 1\}$; bloc românesc izolat (rămâne nealiniat);

4. $j_1 > j_2, j_1 - j_2 > 1$; blocuri izolate (rămân nealiniate);

Precizie (P)	Recall (R)	F-Measure (F)
91.32%	69.58%	78.98%

Tabela 3.1: Performanţele YAWA pe corpusul HLT-NAACL 2003.

După ce blocurile au fost puse în corespondenţă, dintr-o pereche de blocuri se aliniază 1:1 toate cuvintele din engleză şi română care au fie aceeaşi categorie gramaticală, fie aceeaşi metacategorie[16]. Se aplică apoi repetitiv faza 2 pe acest schelet de aliniere[17] până când mulţimea de alinieri $\mathcal{A}_3$ nu mai primeşte alinieri noi. Mulţimea $\mathcal{A}_3$ se reuneşte cu $\mathcal{A}_2$ iar rezultatul este depus în $\mathcal{A}_3$ care astfel conţine alinierea finală a acestei faze.

Ultima fază, faza 4, constă într-o procedură de corecţie a alinierii din faza anterioară. Se elimină corespondenţele care traversează un număr prestabilit de alinieri[18] care au înclinaţii simiare[19]. Înclinaţia unei alinieri, *obl* se calculează cu relaţia

$$obl(i,j) = 1 - \left| \frac{i}{|S_{ro}|} - \frac{j}{|S_{en}|} \right|$$

unde i şi j sunt poziţiile cuvintelor aliniate în română şi engleză iar $|S_{ro}|$ şi $|S_{en}|$ sunt dimensiunile în număr de cuvinte ale frazelor. Mulţimea $\mathcal{A}_4$ conţine alinierile finale (corectate) pentru perechea de fraze S_{ro}, S_{en}.

În tabelele 3.1 şi 3.2 se află evaluările aliniatorului YAWA pe corpusurile paralele de test[20] din competiţiile de aliniere lexicală română-engleză desfăşurate în cadrul workshop-urilor "HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond" ([64]) şi "ACL 2005 workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond" ([59]).

[16]Alinierea se face în ordinea apariţiei. Numărul de cuvinte din engleză de o parte de vorbire/metacategorie dată trebuie să fie *egal* cu numărul de cuvinte din română de aceeaşi parte de vorbire/metacategorie. Folosirea categoriei gramaticale sau a metacategoriei este un parametru configurabil al aliniatorului. Rezultatele cele mai bune au fost obţinute cu metacategorii.

[17]Pentru fiecare pereche de blocuri aflate în corespondenţă.

[18]Configurabil ca parametru al aliniatorului. Rezultatele cele mai bune au fost obţinute cu acest număr egal cu 4.

[19]O aliniere care traversează alte alinieri "paralele" (după funcţia de înclinare *obl*) este intuitiv greşită (fapt observat experimental) pentru că nu respectă regularitatea traducerii.

	Precizie (P)	Recall (R)	F-measure (F)
Faza 1	94.08%	34.99%	51.00%
Faza 2	89.90%	53.90%	67.40%
Faza 3	88.82%	73.44%	80.40%
Faza 4	88.80%	74.83%	81.22%

Tabela 3.2: Performanţele YAWA pe corpusul ACL 2005.

Fie G alinierea de referinţă şi A alinierea produsă de YAWA[21]. Valorile de precizie (**P**), recall (**R**) şi F-measure (**F**) se calculează cu relaţiile:

$$P = \frac{|A \cap G|}{|A|}$$
$$R = \frac{|A \cap G|}{|G|}$$
$$F = \frac{2PR}{P + R}$$

O precizie bună indică faptul că cele mai multe alinieri generate de YAWA sunt corecte pe când un recall bun indică faptul că cele mai multe alinieri care trebuiau să fie găsite au fost. F-measure este media armonică a celor două evaluări fiind astfel o măsură care le combină într-o singură valoare. În tabelul 3.2 se observă că în faza 1, YAWA generează un schelet de aliniere cu precizie mare (în defavoarea recall-ului) pentru că pe acesta se vor construi alinierile din fazele următoare. Cu cât înaintăm prin paşii de aliniere, prin adăugarea de alinieri noi, precizia scade uşor dar recall-ul creşte semnificativ asigurându-se astfel o creştere monotonă a performanţei de ansamblu a aliniatorului (F-measure).

Acest lucru poate să nu fie valabil pentru altă pereche de limbi.

[20]Alinierile de referinţă (eng. "*gold standard alignments*") au fost modificate pentru că textele erau segmentate necorespunzător la nivel de cuvânt. De asemenea, au fost eliminate alte alinieri care erau considerate ca fiind foarte greu de realizat automat (cum ar fi rezoluţia interlinguală a anaforei).

[21]Ambele alinieri nu includ alinierile nule adică perechile de forma $\langle w^i_{ro}, \emptyset \rangle$ sau $\langle \emptyset, w^i_{en} \rangle$ (vezi nota de subsol 6).

3.2 WSDTool

WSDTool ([119, 42]) este un algoritm care a fost proiectat iniţial pentru validarea alinierii conceptuale între ROWN2.0 şi PWN2.0 ([115, 112, 120]). Conceptele reţelei semantice a limbii române au fost aliniate cu cele ale limbii engleze fără ca lexicografii să verifice aplicabilitatea alinierii pe traduceri efective din engleză în română. Experimentul de transfer al adnotării semantice din SemCor2.0 demonstrează convingător (vezi tabelul 2.5) că introspecţia lexicografilor trebuie completată de analize ale alinierilor conceptuale aplicate pe traduceri reale[22] ca o măsură *necesară* în validarea semantică a reţelei semantice lexicale a limbii române ROWN2.0.

WSDTool este un algoritm de DSA care operează pe texte paralele. Ideea esenţială pe care se bazează WSDTool este aceea că dacă admitem că înţelesul unei propoziţii este o funcţie a înţelesurilor unităţilor lexicale care o compun, atunci o pereche de echivalenţi de traducere $\langle w_S, w_T \rangle$ (S limba sursă, T limba ţintă) ar trebui să indice un înţeles comun sau o mulţime de înţelesuri comune (din totalitatea înţelesurilor lui w_S şi w_T) ştiind că traducerea conservă înţelesul sursă. Existenţa reţelelor semantice lexicale aliniate la nivel de concept ne permite să exprimăm precis intuiţia de mai sus cu ajutorul operaţiilor pe mulţimi.

3.2.1 Descrierea algoritmului de bază

Fie C un corpus paralel care conţine N unităţi de traducere. Fiecare unitate de traducere conţine la rândul ei $k+1$ fraze din care k reprezintă traduceri în k limbi diferite ale frazei rămase. În această secţiune prin *cuvânt ţintă* vom înţelege un cuvânt care este dezambiguizat iar prin *cuvânt sursă* un echivalent de traducere al acestuia[23]. Unitatea de traducere devine astfel un tuplu de fraze $\langle S_T, S_{L_1}, S_{L_2}, \ldots, S_{L_k} \rangle$ în care S_T este fraza ţintă (în care există cuvinte de dezambiguizat) iar $S_{L_i}, i = \overline{1,k}$ sunt frazele sursă (cele care conţin echivalenţii de traducere ai cuvintelor de dezambiguizat). Înainte de a trece la descrierea algoritmului este util să mai fixăm cateva notaţii care vor fi folosite de aici înainte:

- fiecare frază S_L^n din unitatea de traducere $n, n \leq N$ în limba L este o mulţime de tupluri $\langle w_L^{n,i}, t_L^{n,i}, l_L^{n,i} \rangle$ de forma cuvânt (w), etichetă morfosintactică (t) şi lemă (l) unde i este poziţia cuvântului în frază iar n este identificatorul unităţii de traducere în care apare fraza;

[22] La momentul scrierii acestei secţiuni, cele mai multe erori de tipul SSINC raportate în tabelul 2.5 au fost corectate.

[23] Termenii de "sursă" şi "ţintă" nu mai indică astfel direcţia de traducere.

- funcţia $ili(l_L, t_L)$ furnizează mulţimea de ILI care corespund sinseturilor în care l_L apare şi are categoria gramaticală[24] identică cu t_L;
- funcţia $occ(l_L, S_L^n)$ numără ocurenţele lemei l_L în fraza S_L^n.

Pentru a putea rula, WSDTool are nevoie de reţele semantice lexicale aliniate la nivel de concept pentru toate limbile care sunt implicate în procesul de dezambiguizare. De asemenea, se prespune că pentru fiecare pereche de fraze în limbile $T, L_i, i = \overline{1,k}$ se dispune de o aliniere lexicală a acestora astfel încât pentru fiecare cuvânt ţintă să se poată identifica cuvântul sursă care se aliniază la el. Algoritmului i se poate da o listă de cuvinte conţinut[25] pentru a fi dezambiguizate sau acesta poate dezambiguiza toate cuvintele conţinut ale corpusului. Acestea fiind spuse, paşii de dezambiguizare ai lui WSDTool sunt următorii:

1. pentru fiecare ocurenţă $l_T^{n,i}, 1 \leq n \leq N, 1 \leq i \leq |S_T^n|$ a lemei ţintă l_T se extrag lemele sursă ale acesteia (alinierile 1:1) $l_{L_i}^{n,j}, 1 \leq j \leq |S_{L_i}^n|$ din fiecare frază $S_{L_i}^n, i = \overline{1,k}$ şi se construieşte matricea echivalenţilor de traducere (MTEQ) din figura 3.6. Putem simplifica notaţia redenumind ocurenţele $l_T^{n,i}$ şi etichetele morfosintactice ale acestora prin

$$M = \sum_{n=1}^{N} occ(l_T, S_T^n)$$
$$l_T^i, i = \overline{1,M}$$
$$t_T^i, i = \overline{1,M}$$

   şi ocurenţele $l_{L_i}^{n,j}$ prin $e_{L_i}(l_T^i)$ adică echivalentul de traducere al lemei l_T^i în limba L_i. Trebuie subliniat faptul că acest echivalent de traducere are aceeaşi etichetă morfosintactică[26] cu l_T^i. În cazul în care l_T^i nu are echivalent de traducere sau acesta are o etichetă morfosintactică diferită, $e_{L_i}(l_T^i) = \epsilon$ (şirul vid).

2. matricea echivalenţilor de traducere este transformată în matrice de dezambiguizare (MSET, figura 3.7), matrice cu acelaşi număr de linii şi coloane cu MTEQ. În această matrice fiecare celulă este ocupată de mulţimea

$$s(i,j) = ili(l_T^j, t_T^j) \cap ili(e_{L_i}(l_T^j), t_T^j)$$
$$ili(\epsilon, t_T^j) = \emptyset$$

   Există două cazuri în ce priveşte mulţimea $s(i,j)$:

[24] De substantiv, adjectiv, verb sau adverb.
[25] Lista conţine lemele acestor cuvinte şi nu forma lor ocurentă.
[26] Identitatea se face numai la nivelul categoriilor gramaticale.

	l_T^1	l_T^2	...	l_T^M
L_1	$e_{L_1}(l_T^1)$	$e_{L_1}(l_T^2)$	...	$e_{L_1}(l_T^M)$
L_2	$e_{L_2}(l_T^1)$	$e_{L_2}(l_T^2)$	...	$e_{L_2}(l_T^M)$
$\vdots$			...	
L_k	$e_{L_k}(l_T^1)$	$e_{L_k}(l_T^2)$	...	$e_{L_k}(l_T^M)$

Figura 3.6: Matricea echivalenţilor de traducere (MTEQ).

(a) $|s(i,j)| \geq 1$; adaugă mulţimea la matricea MSET pe poziţia i, j;

(b) $|s(i,j)| = 0$ sau $s(i,j) = \emptyset$; din diverse motive (descrise mai jos) mulţimea $s(i,j)$ poate fi vidă, caz în care, dacă t_T^j este etichetă de substantiv sau verb, $s(i,j)$ va conţine conceptele c_T făcând parte din perechi $\langle c_T, c_{L_i} \rangle \in ili(l_T^j, t_T^j) \otimes ili(e_{L_i}(l_T^j), t_T^j)$[27] pentru care măsura de similaritate calculată pe graful relaţiei de hipernimie are o valoare de cel puţin 0.33 ($0 \leq K \leq 2$)[28]:

$$sim(c_T, c_{L_i}) = \frac{1}{1+K}$$

(K este numărul de legături între cele două concepte; $K = 0$ dacă şi numai dacă $c_T = c_{L_i}$).

3. mulţimea D_T^j de etichete de sens (ILI) pentru lema l_T^j se obţine prin intersecţia mulţimilor $s(i,j)$:

$$D_T^j = \bigcap_{i=1}^{k} s(i,j), |s(i,j)| \geq 1$$

Cu alte cuvinte, dacă $s(i,j) = \emptyset$ atunci această mulţime nu poate contribui la dezambiguizarea lemei l_T^j fiind astfel exclusă de la intersecţia finală. În cazuri excepţionale (vezi comentariile următoare), mulţimea D_T^j poate să fie vidă caz care se rezolvă prin gruparea ocurenţelor l_T^j (secţiunea 3.2.2).

În pasul 2b mulţmea $s(i,j)$ poate să rămână vidă din oricare din următoarele motive:

[27] $\otimes$ este produsul cartezian.

[28] Valoare stabilită experimental.

	l_T^1	l_T^2	...	l_T^M
L_1	$s(1,1)$	$s(1,2)$	...	$s(1,M)$
L_2	$s(2,1)$	$s(2,2)$	...	$s(2,M)$
$\vdots$			...	
L_k	$s(k,1)$	$s(k,2)$	...	$s(k,M)$

Figura 3.7: Matricea de dezambiguizare (MSET).

- echivalentul de traducere $e_{L_i}(l_T^j)$ poate să fie greşit (vorbim de o eroare a aliniatorului lexical);
- l_T^j poate să nu fie tradus în limba L_i sau poate de asemenea să fie tradus greşit;
- $e_{L_i}(l_T^j)$ poate să îl traducă aproximativ pe l_T^j iar pragul de similaritate ($K \leq 2$) să fie prea ridicat. Similaritatea între două sinseturi de substantive sau verbe (pentru alte măsuri de similaritate sau distanţe semantice pe reţele semantice lexicale, vezi [12]) este o măsură care cuantifică înrudirea înţelesurilor pentru că într-o traducere reală adesea se folosesc hipernimii/hiponimii direcţi ai cuvântului de tradus. De exemplu, în traducerea

  (3.1) "It's the Golden Country - almost," he murmured.

  (3.2) "Parc-ar fi Tărâmul de Aur; în fine, aproape," şopti el.

  intersecţia de la punctul 2 este vidă pentru perechea de traducere $\langle country_{en}, t\check{a}r\hat{a}m_{ro}\rangle$. Figura 3.8 ne arată că înţelesul lui "*country*" a fost tradus în română printr-un hipernim lexicalizat în această limbă ca "*tărâm*" iar în acest caz măsura de similaritate are valoarea 0.5 ($K = 1$):

$$sim(country(5)_{en}, t\check{a}r\hat{a}m(1)_{ro}) = \frac{1}{1+1} = 0.5$$

- literalul $e_{L_i}(l_T^j)$ nu se află în sinsetul care se aliniază la sinsetul aplicabil în context al lui l_T^j. Avem in acest caz un sinset incomplet în reţeaua semantică a limbii L_i (vezi studiul de caz SemCor2.0, tabelul 2.5);

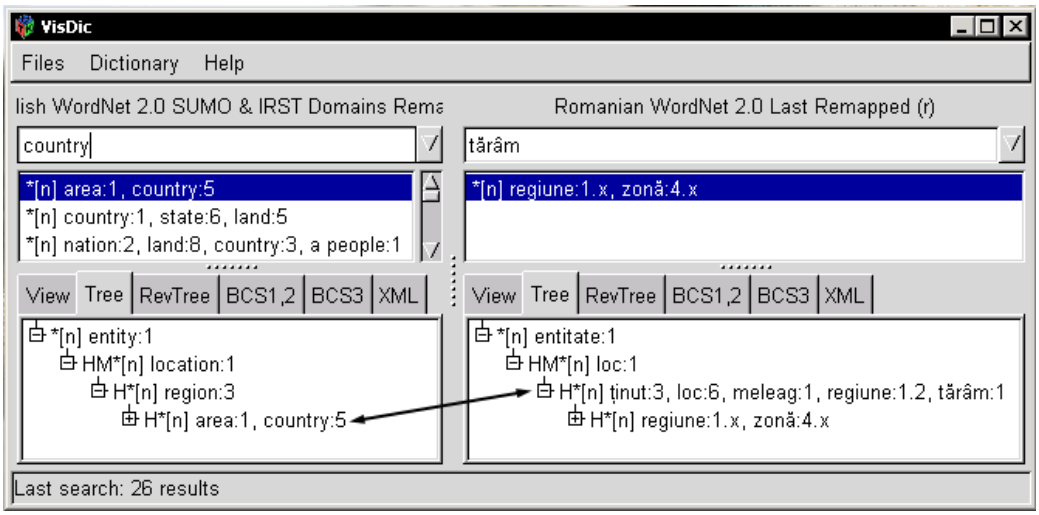

Figura 3.8: O traducere aproximativă (corespondenţă indirectă).

- sinsetul aplicabil în context al lui l_T^j nu este implementat în reţeaua semantică a limbii L_i (ILI-ul care-l identifică nu se află în reţeaua semantică a limbii L_i, vezi de asemenea tabelul 2.5)

3.2.2 O extensie a algoritmului de bază

Pasul 3 al algoritmului de dezambiguizare *nu asigură* pentru fiecare lemă l_T^j o singură etichetă de sens (ILI). Dar în cazul în care $k \geq 3$, sunt mici şansele ca intersecţia finală să conţină mai multe etichete de sens mai ales dacă limbile corpusului paralel au origini diferite şi dacă reţelele semantice ale limbilor L_i sunt corect aliniate la cea a limbii ţintă. Experimentele noastre pe corpusul paralel 1984 care conţine traducerea romanului "Nineteen Eighty-Four" al lui George Orwell în română, cehă şi bulgară au arătat că 4 limbi sunt suficiente pentru dezambiguizarea completă a oricărui cuvânt ţintă[29].

În cazul în care corpusul paralel conţine traduceri în mai puţin de 3 limbi (la limită poate să conţină o singură traducere cum ar fi cea în română din SemCor2.0) trebuie găsită o metodă de a reduce dimensiunea mulţimii D_T^j la 1 (acolo unde este cazul) adică ne trebuie o metodă care să aleagă din D_T^j eticheta de sens aplicabilă lemei l_T^j în contextul ei de apariţie[30].

[29]Cu condiţia ca sinseturile relevante să se afle în reţelele semantice ale celor 3 limbi sursă şi să conţină echivalenţii de traducere ai cuvântului ţintă.

[30]Se poate obiecta că D_T^j poate să nu conţină ILI-ul căutat. Excludem acest caz pre-

Pentru a reduce dimensiunea mulţimii D_T^j apelăm la un algoritm ierarhic de grupare[31] a ocurenţelor l_T^j după similaritatea traducerilor bazându-ne pe ipoteza că traduceri identice pentru două ocurenţe l_T^a şi l_T^b înseamnă sensuri identice pentru aceste ocurenţe. Fie E_{Li} lista ordonată alfabetic a echivalenţilor de traducere în limba sursă L_i pentru lema l_T[32] construită din linia L_i a matricii MTEQ (figura 3.6). Fie E lista obţinută prin concatenarea listelor $E_{Li}, i = \overline{1,k}$ şi $pos(E, e_{L_i}(l_T^j))$ poziţia în E a echivalentului de traducere $e_{L_i}(l_T^j)$. Pentru fiecare ocurenţă l_T^j se construieşte un *vector binar* v_T^j de dimensiune $|E| = \sum_{i=1}^{k} |E_{Li}|$ în care fiecare bit corespunde unui element din E. Acest vector are exact k poziţii egale cu 1 şi anume cele date de $pos(E, e_{L_i}(l_T^j)), i = \overline{1,k}$.

Algoritmul de grupare pe care-l utilizăm este descris în principiu în [46] (vezi de asemenea şi [111]). Acest algoritm este modificat astfel încât condiţia de oprire să permită determinarea claselor de ocurenţe ale lemei l_T care *au sensuri distincte*. Fiecare vector de traducere v_T^j intră în algoritmul de grupare cu mulţimea proprie D_T^j iar doi vectori v_T^a şi v_T^b se pot combina într-o clasă *doar dacă* $D_T^a \cap D_T^b \neq \emptyset$ (D_T^a şi D_T^b sunt diferite de mulţimea vidă $\emptyset$). În acest fel, *numai* ocurenţele care au sensuri comune se pot grupa într-o clasă de echivalenţă iar în momentul în care nu mai există două clase cu sensuri comune, algoritmul se opreşte[33].

Algoritmul de grupare porneşte cu o listă iniţială de clase V de dimensiune M (numărul de ocurenţe al lemei l_T). Fiecare clasă conţine o mulţime O de ocurenţe care au fost grupate în ea (iniţial se află doar ocurenţa l_T^j a lemei l_T), vectorul v_T^j corespunzător lemei l_T^j şi mulţimea de etichete de sens D_T^j. La fiecare iteraţie, două clase x_a şi x_b pentru care conjuncţia de enunţuri

- distanţa euclidiană

$$dist(x_a, x_b) = \sqrt{\sum_{i=1}^{|E|} (v_T^a[i] - v_T^b[i])^2}$$

  este minimă, $\forall a, b, 1 \leq a, b \leq M$ unde $v_T^a[i]$ este poziţia i (la prima iteraţie, 0 sau 1) a vectorului v_T^a şi

supunând că traducerile sunt corecte, reţelele semantice lexicale ale limbilor sursă sunt corect aliniate la cea a limbii ţintă şi că sinseturile relevante sunt implementate şi conţin echivalenţii de traducere din text. Cu WSDTool se pot dezambiguiza numai ocurenţele cuvintelor ţintă ale căror sensuri căutate sunt implementate în *toate* reţelele semantice ale limbilor sursă.

[31] "Hierarchical Clustering Algorithm" în engleză.

[32] Listă fără duplicate.

[33] Încercăm să determinăm practic câte sensuri ale lemei l_T (= numărul de clase finale) sunt prezente în textul paralel şi care sunt acelea (un sens pe clasă).

- $D_T^a \cap D_T^b \neq \emptyset$ sau $D_T^a = \emptyset$ sau $D_T^b = \emptyset$[34],

este adevărată, se unesc într-o clasă x rezultantă compusă din:

- $O = O_a \cup O_b$,
- un vector centroid (reprezentant al noii clase) care are pe poziţia i elementul
$$v_T[i] = \frac{|O_a| v_T^a[i] + |O_b| v_T^b[i]}{|O_a| + |O_b|},$$
- o mulţime de etichete comune ambelor clase D:

$$D = D_T^a \cap D_T^b \; dacă \; D_T^a \neq \emptyset \wedge D_T^b \neq \emptyset$$
$$D = D_T^a \cup D_T^b \; dacă \; D_T^a = \emptyset \vee D_T^b = \emptyset$$

Clasa x_a se suprascrie cu noua clasă x iar clasa x_b se elimină din lista V de clase. Se observă că dacă una din mulţimile de etichete D_T^a sau D_T^b este vidă, algoritmul de grupare poate corecta deficienţa versiunii de bază a lui WSDTool unind această clasă cu una a cărei mulţime de etichete nu este vidă.

Dimensiunea listei V scade cu 1 la fiecare iteraţie până în momentul în care fie

- $|V| = 1$; am ajuns la o singură clasă de ocurenţe caz în care toate ocurenţele $l_T^j, 1 \leq j \leq M$ primesc aceeaşi etichetă din mulţimea D a clasei, sau
- $|V| > 1$ şi $\forall x_a, x_b \in V, D_T^a \cap D_T^b = \emptyset, D_T^a, D_T^b \neq \emptyset$; am ajuns la o partiţie a sensurilor lemei l_T în text.

Oricare ar fi dimensiunea lui V, atribuirea etichetelor de sens se face astfel: pentru fiecare clasă $x_a \in V$, extrage mulţimea de etichete de sens D_a şi dacă:

- $|D_a| = 0$, atribuie tuturor ocurenţelor $l_T^j \in O_a$ eticheta de sens (ILI-ul) corespunzătoare celui mai frecvent sens[35];
- $|D_a| = 1$, atribuie tuturor ocurenţelor $l_T^j \in O_a$ eticheta de sens din D_a;

[34]Singurul caz în care acest enunţ este fals este $D_T^a \cap D_T^b = \emptyset$ şi $D_T^a \neq \emptyset$ şi $D_T^b \neq \emptyset$.

[35]Este vorba despre identificatorul de sens. Această informaţie se obţine din reţeaua semantică a limbii T iar în PWN2.0 de exemplu, identificatorii de sens sunt numere naturale care prin ele însele dau rangul de frecvenţă al sensului în limbă.

	Mărime		Engleză(en)				Română(ro)			
	en	ro	P(%)	R(%)	F(%)	S/C	P(%)	R(%)	F(%)	S/C
ILI	115424	33421	70.217	66.882	68.509	1	53.478	49.805	51.576	1
SUMO	2008	1774	76.788	73.144	74.921	1	65.059	60.572	62.735	1
IRST	168	164	87.636	83.463	85.498	1.092	85.015	79.124	81.964	1.11

Tabela 3.3: Performanţa WSDTool pe SemCor2.0.

- $|D_a| > 1$, ordonează etichetele de sens din D_a după frecvenţa sensului corespunzător şi atribuie eticheta din capul listei tuturor ocurenţelor $l_T^j \in O_a$.

WSDTool echipat cu extensia grupării ocurenţelor cuvântului ţintă are câteva avantaje notabile asupra versiunii de bază a algoritmului:

- prin gruparea claselor de ocurenţe ale cuvântului ţintă se rezolvă eventuala ambiguitate de înţeles din pasul 3 (pagina 62) al algoritmului de bază;
- gruparea claselor de ocurenţe asigură atribuirea de înţeles ocurenţelor pentru care mulţimea de înţelesuri din pasul 3 rămâne vidă;
- gruparea atenuează de asemenea lipsa traducerilor din corpusul paralel care ar fi ajutat procesul de dezambiguizare.

3.2.3 Evaluări

Evaluările algoritmului WSDTool (cu extensia grupării ocurenţelor cuvântului ţintă) sunt date în anexa B. Testele s-au făcut pe corpusul paralel englez-român SemCor2.0 (vezi secţiunea 2.2) care este adnotat cu etichete de sens atât în engleză cât şi în română, adnotare care s-a luat ca referinţă. Algoritmul a fost rulat pentru fiecare limbă în parte iar precizia (**P**), recallul (**R**) şi f-measure (**F**) au fost calculate pentru fiecare fişier al corpusului, pentru *toate* cuvintele conţinut dezambiguizate, cu relaţiile de la pagina 59 în care:

- mulţimea A conţine ocurenţele l_T^j ($T \in \{en, ro\}$) care au fost adnotate de WSDTool;
- mulţimea G conţine ocurenţele l_T^j ($T \in \{en, ro\}$) care sunt dezambiguizate în SemCor2.0. Din tabelul 2.4, ştim că $|G|_{en} = 79595$ iar $|G|_{ro} = 48392$.

În tabelul 3.3 sunt rezumate rezultatele din anexa B. Am folosit 3 inventare de sensuri diferite existente în reţelele semantice lexicale aliniate ale limbilor engleză (PWN2.0) şi română (ROWN2.0). Să notăm că pentru fiecare ILI, există una sau mai multe categorii SUMO corespunzătoare şi de asemenea unul sau mai multe domenii IRST[36] şi din acest motiv, numărul mediu de etichete semantice atribuite pe cuvânt (**S/C**) depăşeşte 1 în cazul domeniilor IRST. Valorile de precizie (**P**) şi recall (**R**) sunt valorile medii din toate fişierele corpusului iar valoarea f-measure (**F**) este calculată între aceste medii. Coloana **Mărime** indică dimensiunea inventarelor de sensuri în engleză şi română (numărul de categorii distincte cu care algoritmul operează).

Se observă că cu cât dimensiunea inventarului de sensuri este mai mică, performanţa algoritmului creşte, rezultat care confirmă afirmaţiile din [113, 114] dar la o scară mult mai mare. Atât domeniile IRST cât şi categoriile SUMO grupează sub o aceeaşi etichetă mai multe concepte (ILI) din reţeaua semantică lexicală. De aceea, putem afirma că cu cât granularitatea semantică[37] este mai mică, cu atât este mai uşoară sarcina algoritmului de dezambiguizare (lucru care este remarcat în majoritatea lucrărilor care tratează DSA). În acelaşi cadru putem afirma că pentru a compara obiectiv doi algoritmi de DSA este nevoie ca ei să folosească acelaşi inventar de sensuri şi cel puţin să dezambiguizeze volume comparabile de date (extrase aleator) dacă nu să ruleze pe acelaşi text.

În tabelul 3.3 performanţa pentru română este sistematic mai slabă decât cea pentru engleză. Acest lucru se explică prin faptul că în engleză toate ocurenţele l^j_{en} care nu au putut fi dezambiguizate[38] au primit automat ILI-ul corespunzător celui mai frecvent sens, informaţie care nu este (încă) disponibilă în ROWN2.0 (vezi şi nota de subsol 35). În ROWN2.0 identificatorii de sens îi respectă pe cei din DEX ([18]) dar în acest dicţionar, sensul numărul 1 nu este neapărat cel mai frecvent sens al cuvântului respectiv în română.

[36]Însă de regulă, un ILI are asociată o singură categorie SUMO şi un singur domeniu IRST. De exemplu, în PWN2.0 există 38 de ILI cu două sau mai multe categorii SUMO asociate şi 23838 de ILI cu două sau mai multe domenii IRST asociate dintr-un total de 115424 de ILI.

[37]Este vorba de distincţiile care se fac între înţelesuri şi care la nivel de ILI sunt foarte fine: nu puţine sunt cazurile în care este greu de precizat prin ce anume diferă două înţelesuri.

[38]Frecvenţă mică şi fără echivalenţi de traducere în română.

Capitolul 4

DSA cu structuri sintactice de dependenţe

Dezambiguizarea semantică automată cu structuri sintactice de dependenţe aderă la ideea conform căreia contextul unui cuvânt este dat de dependenţele sintactice ale lui de restul frazei. Această reprezentare a contextului impune o anumită structură acestuia şi anume, structura contextului este dată de arborele de dependenţe asociat. O astfel de reprezentare a contextului are următoarele avantaje:

1. înţelesul cuvântului ţintă este influenţat direct numai de înţelesurile cuvintelor cu care intră în relaţii de dependenţă sintactică. Acest lucru poate avea o influenţă benefică asupra procesului de dezambiguizare întrucât se elimină astfel zgomotele introduse de contextul de tip fereastră de cuvinte în care cuvântul ţintă intră în relaţie cu fiecare cuvânt din fereastră.

2. structura arborescentă a contextului favorizează atribuirea înţelesurilor cuvintelor fără a se considera o ordine de procesare a acestora.

3. structura contextului permite o cuantificare a interpretării semantice; la nivelul frazei ne putem imagina o măsură care să exprime numeric cât de plauzibilă este o interpretare[1] sau alta.

În lucrările care tratează DSA, dimensiunea[2] contextului nu este determinată. De exemplu, în [129, 130] Yarowsky foloseşte noţiunea de context ca o fereastră (mulţime) de $\pm k$ cuvinte din jurul cuvântului ţintă unde

[1]Prin interpretare înţelegem atribuirea a câte unui înţeles pentru fiecare substantiv, verb, adjectiv sau adverb din fraza dată.

[2]În număr de cuvinte.

$k \in [2, 10] \cap \mathcal{N}$ [3]. Schütze ([93]) foloseşte secvenţe de 3 sau 4 caractere ca unitate componentă a contextului pe care le selectează dintr-o fereastră de 1000 de caractere centrată în cuvântul ţintă. În general nu există un consens în ceea ce priveşte dimensiunea contextului dar un fapt unanim acceptat este acela că influenţele asupra înţelesului cuvântului ţintă ale cuvintelor din contexul său scad cu creşterea dimensiunii contextului[4]. În acest capitol vom considera că fraza în care apare cuvântul de dezambiguizat reprezintă contextul de apariţie al său[5]. Desigur că facem o presupunere care este discutabilă pentru că există cazuri în care cuvinte din afara frazei în care apare cuvântul de dezambiguizat pot ajuta la identificarea înţelesului acestuia. Totuşi trebuie să remarcăm aici că deşi determinarea înţelesului unui cuvânt poate fi favorizată de contexte ale lui care depăşesc barierele frazei, acest lucru nu este de natură *să schimbe structura de bază a algoritmului de dezambiguizat.* Dându-i-se o frază oarecare, un om poate determina în marea majoritate a cazurilor care sunt înţelesurile cuvintelor care o alcătuiesc. Deci, suntem înclinaţi să credem că un context egal cu fraza ar trebui să fie suficient pentru un algoritm de DSA iar eventualele performanţe mai slabe ale algoritmilor de DSA care folosesc un asemenea context ar trebui să fie puse pe seama unor simplificări de formalizare a lui.

O a doua problemă cu privire la contextul de apariţie a unui cuvânt este formalizarea acestuia. Cea mai simplă conceptualizare a contextului este reprezentarea lui ca o mulţime de cuvinte[6] de dimensiune nespecificată ([127, 129, 130, 29]). Marea majoritate a algoritmilor de DSA existenţi consideră că un context al cuvântului ţintă este echivalent cu o mulţime de atribute[7] (extrasă din acest context) relevantă pentru dezambiguizarea înţelesului cuvântului în contextul respectiv ([33, 98, 51, 82]). Câteva dintre aceste atribute sunt:

[3] $\mathcal{N}$ este mulţimea numerelor naturale.

[4] În sprijinul acestei afirmaţii putem aduce rezultatele obţinute în [5] care confirmă faptul că atracţia lexicală dintre două cuvinte scade exponenţial cu distanţa dintre ele. În continuare vom vedea de ce atracţia lexicală este esenţială pentru determinarea înţelesurilor cuvintelor.

[5] Fraza este context de apariţie al unui cuvânt al ei în [97, 63, 73] de exemplu.

[6] Termenul din engleză pentru acest model de context este "*bag of words*" prin care se sugerează ca un context este dat de cuvintele care apar la stânga şi la dreapta cuvântului studiat fără a se considera niciun nivel de segmentare al textului, altul decât cel la nivel de cuvânt. De remarcat este faptul că un cuvânt poate apărea de mai multe ori în această "mulţime" pentru că mulţimea este formată din perechi ⟨cuvânt,poziţie⟩ unde "poziţie" este poziţia la care apare cuvântul în text. Acest tip de context mai este denumit şi "*fereastră de cuvinte*" centrată în cuvântul studiat (numărul de cuvinte de la stânga cuvântului ţintă este egal cu cel de la dreapta lui).

[7] Termenul englezesc pentru atribut este "*feature*". 'Atribut' se referă la un atribut specific contextului şi de aceea îl vom numi şi atribut contextual.

- cuvinte din fereastra cuvântului studiat reduse sau nu la formele lor standard de dicţionar[8]. O opţiune posibilă este includerea sau nu a cuvintelor din clasa părţilor de vorbire neflexionare (cu excepţia adverbului).

- etichetele morfosintactice[9] ale cuvintelor din fereastra cuvântului studiat.

- colocaţiile cuvântului ţintă. Yarowsky lansează ipoteza conform căreia cuvintele care alcătuiesc o colocaţie au înţelesuri determinate în acest context minimal (vezi [128]).

- atribute morfosintactice cum ar fi de exemplu numărul plural la substantive ([11]).

- atribute de natură sintactică cum sunt centrul unui grup nominal care include cuvântul ţintă sau primul substantiv/verb care apare înaintea cuvântului ţintă sau după el ([62]).

Reprezentarea contextului cu ajutorul analizei sintactice la nivel de frază a mai fost realizată în [97, 122]. Structurile de dependenţe au fost folosite la dezambiguizarea înţelesurilor cuvintelor de [53]. În această lucrare, Lin defineşte contextul local al unui cuvânt W ca fiind mulţimea de relaţii sintactice de dependenţă la care acesta participă într-o frază dată. Metoda sa de dezambiguizare se bazează pe postulatul conform căruia "*cuvinte diferite în contexte locale identice tind să aibă înţelesuri similare*"[10] şi procedează la identificarea înţelesului unui cuvânt prin aflarea contextelor locale ale altor cuvinte identice cu cel al cuvântului ţintă.

Acelaşi principiu va fi folosit şi în capitolul de faţă. Algoritmul prezentat va diferi esenţial de cel din [53] prin faptul că va atribui o interpretare frazei

[8]Reducerea formelor ocurente ale cuvintelor la formele lor standard de dicţionar se numeşte "*lematizare*". O lemă este deci o formă morfologică "standard" a unui cuvânt care, pentru substantive de exemplu, este forma de nominativ, sigular, nearticulat. În consecinţă, lematizarea formelor ocurente "băiatul", "băiatului", "băieţii" produce lema "băiat".

[9]O etichetă morfosintactică reprezintă o codificare a unei părţi de vorbire împreună cu combinaţii ale atributelor morfosintactice proprii ei. De exemplu, pentru un substantiv comun se pot codifica numărul, genul, cazul şi articolul enclitic.

[10]O obiecţie perfect justificată formultată de Lin în [53] asupra metodelor de DSA asistate era aceea că cei mai mulţi algoritmi de DSA asistată funcţionează pe principiul similarităţii contextelor: "*cuvinte identice în contexte similare au acelaşi înţeles*", principiu care impune existenţa unor corpusuri de antrenare în care fiecare înţeles al fiecărui cuvânt să fie reprezentat suficient de bine din punct de vedere statistic. Astfel de corpusuri nu există şi pe lângă aceasta, cuvintele care nu au fost întâlnite în procesul de antrenare, nu pot fi dezambiguizate.

ca întreg. În cele ce urmează, vom prezenta pe scurt formalismul sintactic al dependenţelor expus în [61] urmat de o prezentare succintă a modelelor de atracţie lexicală din [131] cu îmbunătăţirile ce se impun iar în final, vom schiţa un algoritm de DSA neasistată pe texte adnotate cu un analizor de legături.

4.1 Formalismul dependenţelor sintactice

În această secţiune vom prezenta modelul de dependenţă sintactică introdus de Igor Mel'čuk în lucrarea sa [61], model care va evidenţia rolul structurii sintactice în construcţia înţelesului unei fraze. În primul rând se va caracteriza relaţia de dependenţă sintactică iar apoi se va prezenta locul pe care această relaţie îl ocupă într-un model lingvistic mai general denumit "Meaning Text Model".

4.1.1 Relaţia de dependenţă sintactică

Comparativ cu gramaticile generative (de constituenţi), formalismul dependenţelor sintactice (abreviat FDS) are următoarele caracteristici diferenţiatoare:

1. **relaţionare** (compară cu **constituenţă**): reprezentarea sintactică cu ajutorul dependenţelor se bazează pe construcţia unui arbore de *relaţii binare* între cuvintele componente ale unei fraze spre deosebire de arborele de constituenţă care reprezintă un mod de formare al frazei din grupuri de *cuvinte adiacente* ca poziţie în frază. De exemplu, în figura 4.1, vedem cum se construieşte un arbore de constituenţi prin formarea succesivă a grupurilor de cuvinte adiacente iar în figura 4.2 observăm relaţiile binare, asimetrice care se constituie între cuvintele aceleiaşi propoziţii. Trebuie subliniat faptul că grupurile de cuvinte pot fi identificate la fel de bine în reprezentarea sintactică cu dependenţe: un grup de cuvinte este un subarbore al arborelui sintactic al frazei.

2. **subcategorizarea**: într-un arbore de constituenţi sintactici, aceştia sunt dominaţi de categorii sintactice abstracte cum ar fi grupul nominal (abreviat NP) sau grupul verbal (abreviat VP, vezi figura 4.1) care formează noduri în arborele sintactic. Poziţiile în care pot apărea aceste categorii sintactice abstracte în propoziţiile şi/sau frazele unei limbi, determină caracteristicile distribuţionale ale lor iar pe baza acestor caracterisitici, li se atribuie rolurile sintactice în fragmentul sintactic respectiv. În contrast, FDS nu admite formularea rolurilor sintactice

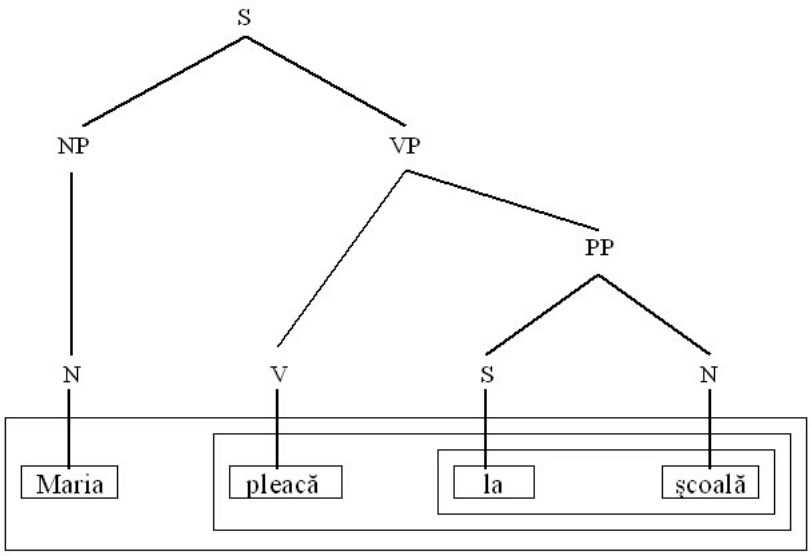

Figura 4.1: Un arbore de constituenţi

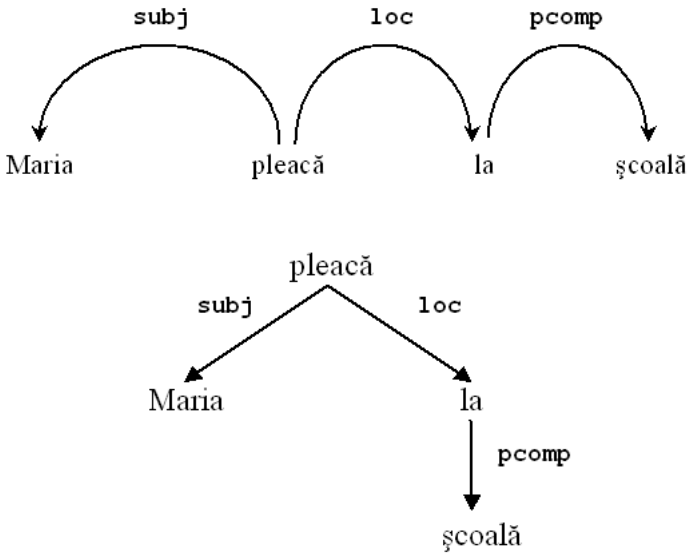

Figura 4.2: Un arbore de relaţii sintactice binare cu rădăcina în “pleacă”

pe baza distribuţiilor unităţilor sintactice şi nici nu permite prezenţa categoriilor sintactice abstracte în reprezentarea sintactică. FDS stipulează faptul că pentru fiecare limbă trebuie construit un inventar de relaţii sintactice şi că acest inventar este *necesar* analizei sintactice. În figura 4.2 avem exemplificate trei relaţii sintactice: `subj` care indică relaţia de subiect dintre "Maria" şi "pleacă", `loc` specifică complementul circumstanţial de loc pentru predicatul propoziţiei iar `pcomp` precizează complementul prepoziţiei "la". În această figură observăm de asemenea că nodurile arborelui de dependenţe sunt chiar cuvintele propoziţiei spre deosebire de arborele de constituenţi din figura 4.1 unde nodurile (interne) sunt date de categoriile sintactice abstracte.

3. **ordinea cuvintelor**: în arborele din figura 4.1 se observă că dacă îl parcurgem în inordine şi reţinem numai nodurile terminale (cuvintele propoziţiei) obţinem forma de suprafaţă a propoziţiei analizate. Acest lucru indică faptul că ordinea cuvintelor în propoziţie este codificată în structura sintactică[11]. Arborele de dependenţe nu codifică ordinea observabilă a cuvintelor propoziţiei pentru că:

   - variază de la o limbă la alta şi este un mijloc universal de codificare a informaţiei sintactice care nu poate fi exprimat formal fără a se ţine cont de limbă[12].
   - o secvenţă de cuvinte poate avea două interpretări sintactice diferite iar permutări ale aceleiaşi secvenţe de cuvinte pot avea aceeaşi interpretare sintactică. Nu există deci o funcţie bijectivă de la mulţimea secvenţelor de cuvinte la mulţimea interpretărilor sintactice (ne referim la cele bazate pe constituenţi, fără transformări.).

   FDS postulează faptul că orice trebuie reprezentat, trebuie reprezentat explicit folosind simbolurile care se impun.

FDS introduce structura sintacă de suprafaţă a unei propoziţii ca o pereche de două mulţimi:

[11] În gramaticile transformaţionale – o varietate a gramaticilor generative – arborele sintactic care codifică forma de suprafaţă se obţine aplicând o serie de transformări structurale asupra arborelui "de adâncime" al propoziţiei. În cazul acestui ultim tip de arbore, parcurgerea lui în inordine nu mai generează forma de suprafaţă a propoziţiei analizate dar existenţa transformărilor structurale implică existenţa unei ordini a cuvintelor în acest arbore.

[12] Aici trebuie adusă în discuţie teoria X–bară în sprijinul gramaticilor generative. Această teorie promovează un set de reguli parametrizate de generare a limbajului care devin dependente de limbă numai prin fixarea unor valori pentru parametrii specifici (pentru detalii vezi de exemplu [13]).

1. mulţimea M a formelor morfologice "de adâncime"[13] reduse ale cuvintelor din propoziţie. O formă morfologică de adâncime a unui cuvânt este dată de lema acestuia indexată de atributele morfosintactice[14] proprii formei sale ocurente şi a părţii de vorbire. Forma morfologică de adâncime şi redusă este forma morfologică de adâncime din care se elimină atributele morfosintactice care nu sunt purtătoare de înţeles[15]. Mai jos este redat un exemplu de formă morfologică de adâncime, formă morfologică redusă de adâncime şi lemă pentru substantivul "băiatului":

$$\begin{aligned} DMorphR(băiatului, subst.) &= băiat_{sg,masc,gen/dat,art} \\ RedDMorphR(băiatului, subst.) &= băiat_{sg,masc} \\ Lema(băiatului, subst.) &= băiat \end{aligned}$$

2. o relaţie binară R definită pe mulţimea M care acoperă toată mulţimea:

$$\forall w, v \in M, \langle w, v\rangle \in R \vee \langle v, w\rangle \in R$$

   În perechea $\langle w, v\rangle \in R$, cuvintele w şi v se numesc centru[16] respectiv dependent[17] iar relaţia care se stabileşte între ele se reprezintă grafic $w \rightarrow v$.

   Graful asociat relaţiei R este un arbore iar proprietăţile pe care R trebuie să le aibă astfel încât graful asociat să fie un arbore, sunt:

   - este **ireflexivă**:

     $$\forall w \in M, \langle w, w\rangle \notin R$$

     Cu alte cuvinte nu se poate trasa o relaţie sintactă între un cuvânt şi el însuşi.
   - este **asimetrică**[18]:

     $$\forall w_1, w_2 \in M, \langle w_1, w_2\rangle \in R \Rightarrow \langle w_2, w_1\rangle \notin R$$

     De exemplu, perechile

     $$\langle proprietar_{sg,masc}, apartament_{sg,masc}\rangle$$
     $$\langle apartament_{sg,masc}, proprietar_{sg,masc}\rangle$$

[13] "D(eep-)Morph(ological) R(erepsentation)" sau abreviat DMorphR.

[14] Atributele morfosintactice se mai numesc şi variabile morfosintactice. De exemplu, variabila morfosintactică *număr* poate lua valori în mulţimea $\{sg, pl\}$.

[15] De exemplu, în română, cazul şi articolul enclitic pentru substantive.

[16] În engleză, "governor".

[17] În engleză, "dependant".

[18] Asimetria implică ireflexivitatea.

corespunzătoare perechilor de forme ocurente

$$\langle proprietarul, apartamentului \rangle$$
$$\langle apartamentul, proprietarului \rangle$$

nu pot coexista în R deoarece au semnificaţii diferite şi ar trebui să aibă astfel şi structuri sintactice diferite.

- este **intranzitivă**[19]:

  $$\forall w_1, w_2, \ldots, w_k \in M, \langle w_1, w_2 \rangle \in R \wedge \cdots \wedge \langle w_{k-1}, w_k \rangle \in R \Rightarrow \langle w_1, w_k \rangle \notin R$$

  Proprietatea de intranzitivitate prezentată aici ar trebui numită "intrazitivitate totală" pentru că negarea definiţiei de tranzitivitate nu produce această definiţie ci următoarea definiţie (exprimăm $\langle a, b \rangle \in R$ ca aRb şi considerăm $k = 3$ pentru că derivarea este aceeaşi $\forall k, k \leq |M|$):

  $$\begin{aligned}
  \neg(\forall w_1, w_2, w_3 \in M, w_1Rw_2 \wedge w_2Rw_3 \Rightarrow w_1Rw_3) \Leftrightarrow \\
  \exists w_1, w_2, w_3 \in M, \neg(w_1Rw_2 \wedge w_2Rw_3 \Rightarrow w_1Rw_3) \Leftrightarrow \\
  \exists w_1, w_2, w_3 \in M, \neg(\neg(w_1Rw_2 \wedge w_2Rw_3) \vee w_1Rw_3) \Leftrightarrow \\
  \exists w_1, w_2, w_3 \in M, \neg(\neg w_1Rw_2 \vee \neg w_2Rw_3 \vee w_1Rw_3) \Leftrightarrow \\
  \exists w_1, w_2, w_3 \in M, w_1Rw_2 \wedge w_2Rw_3 \wedge \neg w_1Rw_3
  \end{aligned}$$

  Altfel spus, este suficient ca trei perechi cu proprietăţile de mai sus să nu respecte tranzitivitatea iar relaţia devine intranzitivă. Totuşi trebuie să impunem condiţia ca *toate* triplurile de perechi cu proprietăţile de mai sus să nu respecte caracteristica de tranzitivitate ori vom avea cazuri în care un cuvânt are două noduri părinte într-o structură care, evident, nu mai este un arbore. De exemplu, pentru figura 4.3, dacă adaugăm la R perechea $\langle lucra_{ind,prez}, maşină_{sg,fem} \rangle$ obţinem o relaţie intranzitivă în sensul negării definiţiei clasice dar care ar fi inacceptabilă pentru un arbore de dependenţe sintactice.

Pe lângă cele trei proprietăţi generale ale lui R mai trebuie impuse două pentru a transforma această relaţie într-una de dependenţă sintactică:

- existenţa unui nod unic în arborele de dependenţe care să constituie rădăcina arborelui:

  $$\exists! w \in M, ((\forall x \in M, \langle x, w \rangle \notin R)$$

[19]Intranzitivitatea aşa cum este definită aici implică asimetria.

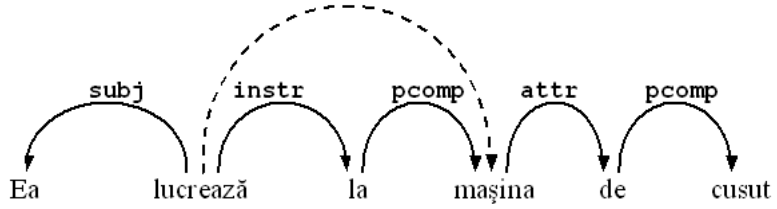

Figura 4.3: Relaţie intranzitivă care nu este relaţie de dependenţă sintactică

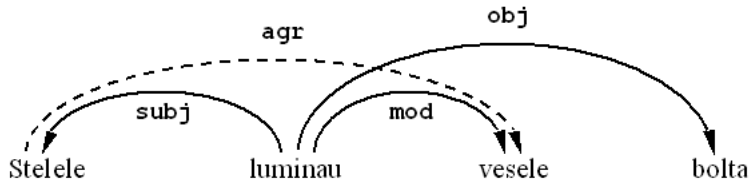

Figura 4.4: Exemplu în care condiţia de planaritate nu este îndeplinită

- inexistenţa unui nod cu doi părinţi diferiţi:

$$\forall x, y, w \in M, x \neq y, \neg(\langle x, w\rangle \in R \wedge \langle y, w\rangle \in R)$$

Alături de mulţimile menţionate mai sus avem nevoie de o funcţie r care să facă legătura între relaţia R şi multimea I a denumirilor relaţiilor sintactice din limba respectivă (sau inventarul de relaţii sintactice al limbii):

$$r : R \rightarrow I, r(x) \in I, \forall x \in R$$

Pentru un alt model matematic al structurii de dependenţă cât şi pentru un inventar al relaţiilor sintactice pentru limba română, cititorul poate consulta [35]. În [35] se consideră o nouă restricţie asupra relaţiei R şi anume aceea de planaritate (această proprietate a lui R este semnalată şi de [61]). O explicaţie intuitivă a proprietaţii de planaritate a relaţiei de dependenţă sintactică R este aceea că graful relaţiei desenat pe forma de suprafaţă a propoziţiei nu conţine arce care să se intersecteze (pentru o definiţie matematică a planarităţii, vezi [35]). Deşi majoritatea propoziţiilor din română au analize sintactice de dependenţe planare, există şi contraexemple (exemplu din [43], vezi figura 4.4). Relaţia de acord (notată `agr`) dintre substanti-

vul “Stelele” şi adjectivul “vesele” intersectează relaţia de complement direct (`obj`) dintre verbul “luminau” şi substantivul “bolta”. Aici observăm de asemenea şi o dublă dependenţă a adjectivului “vesele” astfel încât proprietatea de arbore a analizei sintactice este infirmată. Mel’čuk explică fenomenele de acest tip prin faptul că în orice limbă există cel puţin trei tipuri de relaţii sintagmatice între cuvintele unei propoziţii:

- relaţii morfologice care sunt în totalitate dependente de limbă;
- relaţii sintactice care sunt în parte dependente de limbă, în parte conceptuale;
- relaţii semantice care sunt conceptuale pe de-a-ntregul.

Astfel, relaţia trasată punctat în figura 4.4 este una morfologică şi nu intră în componenţa relaţiei de dependenţă sintactică R descrisă mai sus (pentru detalii vezi [61, pag. 25,106]).

4.1.2 Meaning Text Model

“Meaning Text Model” (abreviat MTM) reprezintă un cadru de lucru pentru descrierea şi studiul limbajelor naturale. Pe lângă componentele consacrate ale actului de vorbire, emiţătorul, receptorul şi canalul de comunicaţie, modelul include următoarele trei componente suplimentare:

- un *conţinut* cu o structură ierarhică care este comunicat de către emiţător receptorului şi care este inclus într-o mulţime numărabilă de semnificaţii sau **înţelesuri**[20].
- o *formă lingvistică* de exprimare a conţinutului care va fi denumită generic **text**[21] inclusă şi ea într-o mulţime numărabilă a textelor.
- o corespondenţă de $m : n$ $m \geq 1, n \geq 1$ $m, n \in \mathcal{N}$ între mulţimea înţelesurilor şi mulţimea textelor.

MTM reprezintă un sistem de reguli care descrie corespondenţa între mulţimea înţelesurilor şi mulţimea textelor sau, mai exact, între mulţimea

[20] Un înţeles este astfel o entitate de sine stătătoare dintr-o mulţime (se acceptă aici reprezentarea discretă a înţelesului). De asemenea, un “înţeles” se defineşte ca fiind un invariant al transformărilor sinonimice şi este deci ceea ce se extrage dintr-un cuvânt sau enunţ numai pe baza cunoştinţelor de natură lingvistică fără a se recurge la logică, pragmatică, cunoştinţe enciclopedice sau alte cunoştinţe extralingvistice.

[21] Un text este deci orice formă de comunicare lingvistică pornind de la cuvânt, secvenţă de cuvinte, propoziţii, fraze şi aşa mai departe.

reprezentărilor simbolice ale înţelesurilor Sem şi mulţimea reprezentărilor simbolice fonetice ale textelor $Phon$:

$$Sem = \{SemR_i \mid i \in \mathcal{N}\},$$
$$Phon = \{PhonR_j \mid j \in \mathcal{N}\},$$
$$Sem \overset{m:n}{\Longleftrightarrow} Phon$$

A descrie direct corespondenţa dintre Sem şi $Phon$ este un lucru imposibil pentru că aceasta este una foarte complexă. Un mod de simplificare a descrierii este acela de a introduce straturi intermediare de reprezentare care pentru orice limbaj natural se evidenţiază la cel puţin două nivele de analiză: nivelul formei ocurente a cuvântului şi cel al propoziţiei. Obţinem astfel reprezentările intermediare morfologice şi sintactice $MorphR$ respectiv $SyntR$[22] iar schema corespondenţei devine:

$$\{SemR_i\} \overset{m:n}{\Longleftrightarrow} \{SyntR_j\} \overset{m:n}{\Longleftrightarrow} \{MorphR_k\} \overset{m:n}{\Longleftrightarrow} \{PhonR_l\}, \forall i, j, k, l \in \mathcal{N}$$

În ce priveşte formalizarea structurilor $SemR, SyntR, MorphR, PhonR$, acestea se încadrează în reprezentarea cu grafuri. $SemR$ este un graf conex şi orientat, $SyntR$, aşa cum am văzut în secţiunea anterioară, este un arbore iar $MorphR$ şi $PhonR$ sunt la rândul lor arbori dar pentru care fiecare nod are un singur descendent (exceptând unica frunză). Fiecare nivel de reprezentare (mai puţin cel semantic $SemR$) este subîmpărţit în două: subnivel de suprafaţă (notat cu prefixul "S" ataşat denumirii nivelului, de exemplu $SSyntR$) adaptat la forma de suprafaţă a propoziţiei şi subnivel de adâncime (prefix "D") adaptat la reprezentarea semantică a ei. Se face în acest mod o trecere progresivă spre reprezentarea semantică evidenţiindu-se la fiecare subnivel de adâncime proprietăţile semantice care sunt observabile la acel nivel. De asemenea, fiecare nivel împreună cu subnivelele lui codifică *aceeaşi* informaţie lingvistică conţinută de propoziţie cu menţiunea că ambiguitatea *scade* pe măsură ce ne apropiem de $SemR$[23].

În cele ce urmează vom exemplifica cum se obţine o aceeaşi interpretare pentru două propoziţii diferite având reprezentări sintactice diferite (a doua se obţine din prima prin operaţia de pasivizare). Fie exemplele:

(4.1) Ion a spălat maşina.

(4.2) Maşina a fost spălată de Ion.

[22] Aceasta este relaţia R pe care am descris-o în secţiunea anterioară.

[23] Altfel spus, nivelele mai apropiate de $SemR$ conţin mai multă metainformaţie de reprezentare decat nivelele depărtate fiind din această cauză mai explicite decât acestea din urmă. O consecinţă firească a acestui fapt este că cu cât creşte nivelul de explicitare, cu atât scade nivelul de ambiguitate (vezi [61, pag. 49,50]).

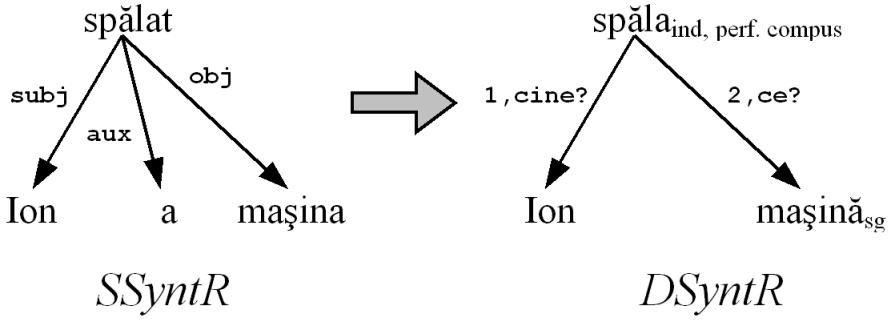

Figura 4.5: Exemplul 4.1: Translaţia de la $SSyntR$ la $DSyntR$

În acest caz, la subnivelul de adâncime al reprezentării sintactice $DSyntR$ predicatul propoziţiei "a spăla" va avea aceleaşi argumente: executantul operaţiei, "Ion" şi obiectul care suferă operaţia, "maşina". Ideea centrală a reprezentării uniforme se rezumă la precizarea că la nivelul $DSyntR$, fiecare formă morfologică redusă de adâncime predicativă are argumentele[24] identificate şi ordonate astfel încât pentru fiecare argument este posibilă precizarea tipului său cât şi a poziţiei pe care o ocupă[25] în structura predicativă a lexemului. Relaţia actant poate fi realizată la subnivelul de suprafaţă al reprezentării sintactice $SSyntR$ de o multitudine de construcţii sintactice stabilindu-se astfel o translaţie deterministă de la structurile sintactice de suprafaţă proprii unui lexem predicativ la relaţiile actant ale acestuia.

În exemplul de mai sus la propoziţia 4.1, la nivelul $SSyntR$, între "Ion" şi "spălat" se stabileşte o relaţie sintactică de subiect (`subj`) care se transferă la nivelul $DSyntR$ în relaţia actant de tip "cine?" aflată pe poziţia 1 în structura argumentală a verbului "a spăla". Similar, între "spălat" şi "maşina" există o relaţie de tip obiect (`obj`) care la rândul ei se transferă în relaţia actant de tip "ce?" aflată pe poziţia 2. Pentru propoziţia 4.2, relaţia sintactică de subiect se transferă la poziţia 2 a structurii argumentale a verbului pentru că acesta se află la diateza pasivă (vezi figurile 4.5 şi 4.6).

[24]Relaţia formă morfologică redusă de adâncime – argument proprie unui predicat mai este denumită la acest nivel şi actant (în engleză, "actant"). Structura argumentală a unui lexem predicativ reprezintă de fapt un cadru de valenţă generalizat al lexemului.

[25]În plus, tipurile şi poziţiile argumentelor sunt independente de limbă.

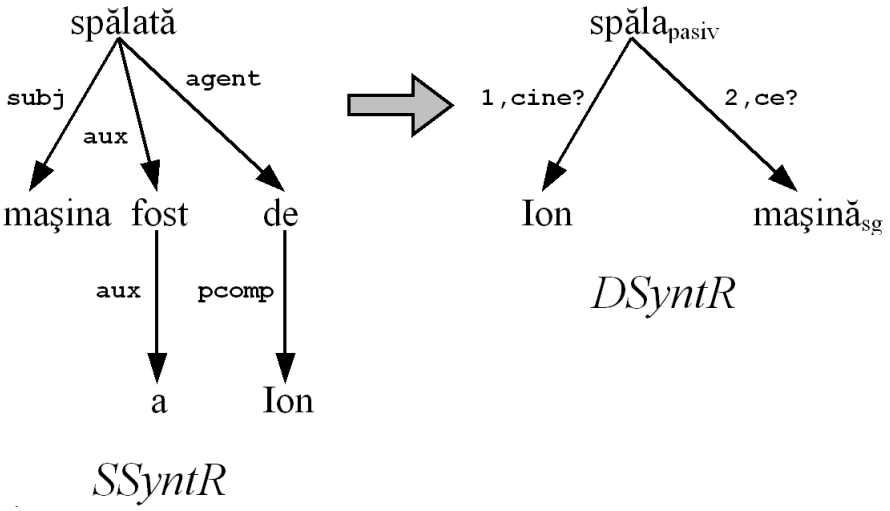

Figura 4.6: Exemplul 4.2: Translaţia de la $SSyntR$ la $DSyntR$

4.2 Modele de atracţie lexicală. Analizorul de legături LexPar

Generarea automată a unei structuri sintactice de dependenţe conform cu nivelul de analiză $SSyntR$ presupune existenţa unui algoritm care să producă structura sintactică având la intrare o gramatică de dependenţe şi fraza de analizat[26]. Dacă algoritmi de analiză sintactică cu dependenţe există (vezi de exemplu [77, 79, 99, 80, 30]), nu se poate afirma cu tărie acelaşi lucru despre gramaticile de dependenţe. Ce se poate afirma despre gramaticile de dependenţe (şi nu numai) în general cât şi despre cele existente la ora actuală ar fi:

- construcţia umană a unei gramatici cuprinzătoare pentru o limbă implică un timp îndelungat de dezvoltare;
- nu se poate garanta completitudinea unei gramatici în sensul că vor exista secvenţe de cuvinte formând expresii gramatical corecte pentru care gramatica nu conţine reguli de combinare;
- în timp ce pentru engleză există gramatici construite (vezi de exemplu [44, 45]) pentru alte limbi (printre care şi româna) nu sunt disponibile asemenea gramatici.

[26]Acest tip de analiză sintactică se numeşte în engleză "*grammar-driven parsing*".

O alternativă la modelul de analiză sintactică bazată pe gramatici o reprezintă analiza sintactică cu modele gramaticale induse automat[27]. O cerinţă firească[28] pentru construcţia acestor modele o reprezintă existenţa corpusurilor adnotate cu structuri sintactice cum ar fi cele din [32, 58] din care programe de învăţare automată[29] extrag informaţii statistice cu privire la modurile de combinare ale elementelor sintactice. Aceste informaţii sunt asamblate de obicei într-un "model gramatical" probabilistic care este folosit ulterior la analiza sintactică a frazelor ([78, 22]).

Indiferent de metoda de analiză sintactică, un analizor sintactic poate fi evaluat după următoarele trei măsuri (propuse în [78]):

- **robusteţe**: un analizor sintactic P este robust dacă şi numai dacă fiind dată o colecţie de fraze $T = \{x_1, x_2, \ldots, x_n\}$ în limba L, P atribuie *cel puţin* o analiză sintactică pentru orice x din T;

- **dezambiguizare**: un analizor sintactic P dezambiguizează[30] dacă şi numai dacă fiind dată o colecţie de fraze $T = \{x_1, x_2, \ldots, x_n\}$ în limba L, P atribuie *cel mult* o analiză sintactică pentru orice x din T;

- **acurateţe**: un analizor sintactic P este performant dacă şi numai dacă fiind dată o colecţie de fraze $T = \{x_1, x_2, \ldots, x_n\}$ în limba L, P găseşte analiza sintactică potrivită[31] pentru orice x din T;

În ceea ce urmează vom descrie modelele de atracţie lexicală (MAL) introduse de Deniz Yuret în [131] şi apoi, vom prezenta analizorul de legături **LexPar** [40] care extinde MAL cu reguli de combinare. De asemenea, vom evalua LexPar cu privire la robusteţe, dezambiguizare şi într-un mod aproximativ, cu privire la acurateţe.

4.2.1 Modele de atracţie lexicală

O versiune a modelelor de atracţie lexicală a fost descrisă de Deniz Yuret în [131] dar ideea de atracţie lexicală ca dependenţă între cuvinte aflate la distanţe arbitrare este introdusă în [5]. În cele ce urmează ne vom ocupa de

[27] În engleză, "*data-driven parsing*".

[28] Există şi metode care construiesc modele gramaticale din corpusuri care nu sunt adnotate la nivel sintactic (vezi [90]).

[29] În engleză, "*machine learning*".

[30] Această noţiune se referă de fapt la capacitatea sistemului de a alege o singură analiză din mai multe analize posibile pentru o frază dată.

[31] Această comparaţie se face între analiza dată de sistem şi analiza dată de om pentru o aceeaşi frază. De obicei se folosesc corpusurile adnotate la nivel sintactic şi se măsoară procentul de analize sintactice corecte furnizate de analizorul sintactic pe corpusul adnotat.

MAL ale lui Yuret pentru că ele se află la baza analizorului de legături Lex-Par. În plus, Yuret este cel care foloseşte MAL pentru descoperirea relaţiilor de dependenţă dintre cuvintele unui text *neadnotat*. În acest scop, dezvoltă de asemenea un algoritm care îmbină descoperirea relaţiilor cu trasarea lor, un principiu care se dovedeşte a fi esenţial pentru succesul ambelor procese. Algoritmul său emulează întrucâtva capacitatea umană de analiză sintactică.

Atracţia lexicală în accepţiunea lui Yuret, este o măsură a afinităţii de combinare a două cuvinte într-o frază. Yuret se bazează pe faptul că atât achiziţia cât şi învăţarea limbajului natural se fundamentează pe capacitatea umană de a construi în memorie o tabelă asociativă a conceptelor care apoi să fie folosită la determinarea corectă a înţelesului sintagmelor sau propoziţiilor. El ilustrează această ipoteză printr-un exemplu de achiziţie de limbaj în cazul unui copil care nu cunoaşte încă regulile sintactice de formare a propoziţiilor gramatical corecte dar cu toate acestea, este în stare să alcătuiască propoziţii simple corecte (vezi [131, pag. 10]). Acest lucru se întâmplă datorită modului în care conceptele se asociază. Alăturarea conceptelor într-o propoziţie sau frază devine posibilă prin afinitatea lor crescută dobândită prin învăţare iar în cazul copiilor învăţarea înseamnă în principal repetiţie. De aici decurge formalizarea afinităţii conceptelor – a "atracţiei lexicale" în consecinţă – ca o măsură (probabilistică).

Pentru cazul de faţă, un model probabilistic al unei limbi scrise este dat de un câmp de probabilitate $(\Omega, \mathcal{F}, P)$ în care Ω este spaţiul evenimentelor elementare, o mulţime nevidă care conţine simbolurile şi secvenţe de simboluri[32] ale unei limbi iar $\mathcal{F}$ este mulţimea de evenimente, $\mathcal{F} = 2^{\Omega}$[33]. P este o probabilitate definită pe $\mathcal{F}$ care distribuie masa de probabilitate peste toate evenimentele elementare din Ω. Simplificând, putem spune că un model probabilistic al unei limbi este dat de o funcţie probabilitate care asignează fiecărui cuvânt, expresie, propoziţie sau frază o probabilitate de apariţie. Singurul mod de a cuantifica aceste probabilităţi este estimarea lor din texte în limba respectivă, texte care vor trebui să aibă dimensiuni foarte mari pentru a putea obţine estimări robuste.

Probabilitatea P are o distribuţie necunoscută peste mulţimea de evenimente elementare dar probabilitatea unei propoziţii (sau fraze) S văzută ca o secvenţă de n cuvinte poate fi aproximată în funcţie de gradul de independenţă al unui cuvânt faţă de contextul său, astfel:

1. $P(S) = p(w_1) \cdot p(w_2) \cdot \ldots \cdot p(w_n) = \prod_{i=1}^{n} p(w_i)$ în care cuvintele se consideră că apar independent unele de altele. O astfel de aproximaţie

[32] Simbolurile sunt cuvinte, semne de punctuaţie, diverse alte simboluri ale limbii cum ar fi de exemplu '+', '-', etc. iar secvenţele expresii şi propoziţii sau fraze.

[33] 2^{Ω} este mulţimea părţilor lui Ω.

este în mod evident nepotrivită pentru limbajul natural pentru că orice permutare a secvenţei de cuvinte S produce aceeaşi probabilitate iar modelul ar trebui să poată atribui probabilităţi mai mari propoziţiilor şi frazelor gramatical corecte decât celor greşite.

2. $P(S) = p(w_1) \cdot p(w_2|w_1) \cdot \ldots \cdot p(w_n|w_{n-1}) = p(w_1) \cdot \prod_{i=1}^{n} p(w_i|w_{i-1})$ în care se foloseşte un model Markov de ordin 1 (vezi [57, pag. 192, 317]): fiecare cuvânt depinde ca apariţie doar de precedentul. O astfel de aproximaţie este mai bună decât independenţa totală dar are şi ea deficienţele ei:

   (4.3) Individul ieşi pe uşa din dos în grabă, vădit încurcat.

   În exemplul 4.3, expresia adverbială "*în_grabă*" depinde de predicatul propoziţiei "*ieşi*" în sensul că verbul "*ieşi*" favorizează apariţia acesteia în contextul său într-o mult mai mare măsură decât, să zicem, "*uşa*" iar adjectivul "*încurcat*" determină substantivul "*Individul*" (cele două se află la cele două capete ale propoziţiei). Bineînţeles că s-ar putea mări ordinul modelului Markov astfel încât astfel de dependenţe să fie capturate dar un asemenea model ar fi realizabil doar teoretic pentru că din punct de vedere practic numărul de parametri care ar trebui estimaţi ar fi imens (vezi tabelul 6.1 din [57, pag. 194]).

Dacă ar fi să calculăm entropia propoziţiei S în cele două aproximaţii de mai sus am avea:

$$H_1(S) = -\sum_{i=1}^{n} p(w_i) \log p(w_i) = \sum_{i=1}^{n} H_1(w_i)$$

şi pentru că în cazul 2 avem un model Markov de ordin 1 iar din [57, pag. 64, ecuaţia 2.30] avem

$$H(X_1, X_2, \ldots, X_n) = H(X_1) + H(X_2|X_1) + \ldots + H(X_n|X_1, \ldots, X_{n-1})$$

atunci

$$H_2(S) = H_2(w_1) + \sum_{i=2}^{n} H_2(w_i|w_{i-1})$$

Diferenţa între cele două entropii este

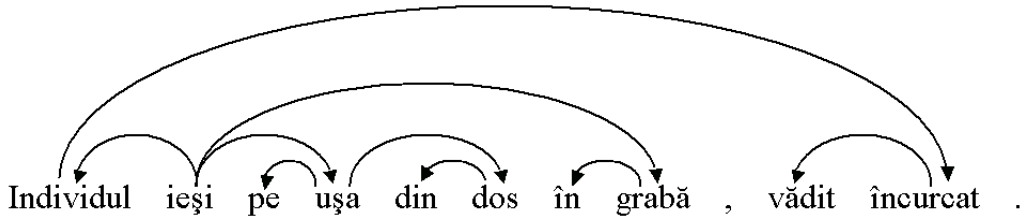

Figura 4.7: Dependenţe ale cuvintelor în context.

$$
\begin{aligned}
H_1(S) - H_2(S) &= H_1(w_1) + \sum_{i=2}^{n} H_1(w_i) - H_2(w_1) - \sum_{i=2}^{n} H_2(w_i|w_{i-1}) \\
&= \sum_{i=2}^{n} H_1(w_i) - \sum_{i=2}^{n} H_2(w_i|w_{i-1}) \\
&= \sum_{i=2}^{n} [H_1(w_i) - H_2(w_i|w_{i-1})] \\
&= \sum_{i=2}^{n} I(w_i; w_{i-1}) = \sum_{i=2}^{n} I(w_{i-1}; w_i)
\end{aligned}
$$

şi este pozitivă pentru că este suma informaţiilor mutuale ale cuvintelor adiacente din propoziţie (vezi [57, pag. 66,67 şi ecuaţia 2.36]). Asta înseamnă că entropia modelului Markov este mai mică decât entropia modelului bazat pe independenţa totală ceea ce duce la concluzia că modelul Markov atribuie o probabilitate mai mare propoziţiei S decât contracandidatul său[34].

Am făcut această minidemonstraţie pentru a servi ca suport introducerii modelelor de atracţie lexicală. În exemplul 4.3, am dori să existe o dependenţă între cuvintele *ieşi* şi *uşa* şi între *uşa* şi *dos* sau între *Individul* şi *încurcat*. În figura 4.7 sunt reprezentate dependenţele cuvintelor din exemplul 4.3 într-o structură de dependenţe care respectă relaţia de dependenţă sintactică din secţiunea 4.1. Yuret ([131, pag. 25]) arată că entropia unei propoziţii S a cărei probabilitate este dată de un model care se bazează pe o astfel de structură este mai mică decât entropia propoziţiei dată de modelul Markov.

Un model de atracţie lexicală (MAL) este aşadar un model de probabilitate a unei fraze în care fiecare cuvânt al frazei este probabilistic dependent

[34]Dacă S este gramatical corectă. De asemenea modelul Markov va atribui o probabilitate mai mică unei propoziţii gramatical greşite decât celălalt model.

numai de centrul său. Pe de altă parte, se presupune că informaţia mutuală între două cuvinte ale unei fraze este măsura relaţionării sintactice ale acestora[35]. Această măsură este simetrică însă şi nu poate fi o măsură a unei relaţii asimetrice dar Yuret demonstrează că probabilitatea unei fraze calculată cu un MAL este aceeaşi indiferent de cuvântul care este ales ca rădăcină a arborelui de dependenţe ([131, pag. 28,29]). Acest rezultat conduce la observaţia că orientarea arcelor poate fi eliminată fără ca modelul să calculeze altă probabilitate pentru o aceeaşi frază caz în care:

- un MAL devine un model de probabilitate a unei fraze în care fiecare cuvânt al său este probabilistic dependent numai de cuvintele cu care *se leagă*;
- informaţia mutuală poate fi o măsură a relaţionării sintactice a două cuvinte dintr-o frază.

Cu aceste precizări vedem că de fapt un MAL aproximează o relaţie sintactică de dependenţă din secţiunea 4.1 prin eliminarea orientării arcelor şi prin neconsiderarea inventarului de relaţii sintactice al limbii[36]. Vom numi această aproximaţie o *structură de legături* a unei fraze care este un graf ale cărui noduri sunt cuvintele frazei şi care este neorientat, conex, aciclic şi planar (vezi pagina 77). De asemenea, numim o *legătură* un arc al acestui graf.

Un MAL se poate construi[37] de exemplu dintr-un corpus care este adnotat cu structuri de dependenţă sintactică. Yuret descrie un algoritm care construieşte un MAL din texte simple, neadnotate şi care îmbină construcţia structurii de legături a unei fraze cu estimarea parametrilor modelului pe fraza respectivă. Fie C un corpus care conţine M fraze $S_i, i = \overline{1, M}$, liste de unităţi lexicale $w_j, j = \overline{1, n_i}$ (n_i este dimensiunea frazei S_i, $N = \sum_{i=1}^{M} n_i$). Fiecare frază S_i conţine de asemenea şi două marcaje de început, respectiv sfârşit de frază aflate pe poziţiile 0 şi $n_i + 1$. Algoritmul de construcţie/descoperire (vezi algoritmul 1) a structurii de legături are două componente:

- o memorie care înregistrează perechile de cuvinte $\langle w_a, w_b \rangle$ din fraza S_i, $0 \leq a < b \leq n_i + 1$ şi de asemenea şi perechile $\langle w_a, * \rangle$, $\langle *, w_b \rangle$ şi $\langle *, * \rangle$ (ultima este numărul K de perechi care au fost introduse în memorie) unde $*$ reprezintă orice cuvânt. Pentru fiecare pereche se contorizează numărul de apariţii a acesteia în corpus.

[35]În sensul relaţiei de dependenţă sintactică.

[36]Toate celelalte proprietăţi sunt păstrate însă, printre care şi proprietatea de planaritate.

[37]Ne referim la estimarea parametrilor săi.

- un analizor de legături (procesor) care are sarcina de a "desena" structura de legături astfel încât proprietăţile acesteia să fie respectate şi de a scrie în memorie perechile de cuvinte care sunt luate în calcul la legare.

Algoritmul 1 Analizorul de legături al lui Yuret.

```
1: procedure LINK-SENTENCE(S_k)
2:     for j ← 1 to length(S_k) − 1 do
3:         for i ← j − 1 downto 0 do
4:             last ← pop(rightlinks(i), Stack)
5:             for all l ∈ last do
6:                 MinLink[i] ← min(l, MinLink[rightindex(l)])
7:             end for
8:             if mi(⟨i, j⟩) > 0, mi(MinLink[i]), mi(Stack[s]), ∀s then
9:                 unlink(Stack[s]), ∀s
10:                reset(Stack)
11:                unlink(MinLink[i])
12:                MinLink[i] ← link(i, j)
13:            end if
14:            push(leftlinks(i), Stack)
15:        end for
16:    end for
17: end procedure
```

Fraza S_k pe care rulează algoritmul are $\texttt{length}(S_k) = n_k + 2$ cuvinte (cu tot cu marcajele de început şi sfârşit). Notaţia $S_k[i], 0 \leq i < n_k + 2$ desemnează cuvântul de pe poziţia i din frază (prima poziţie în acest vector are $i = 0$). O legătură l este o pereche de indecşi $\langle i, j \rangle, i < j$ care se formează cu funcţia `link` şi se distruge cu funcţia opusă `unlink`. *Stack* este o stivă care conţine legături (funcţiile `push` şi `pop` adaugă respectiv şterg legături din stivă în ordinea proprie acestei structuri de date) şi *MinLink* este un vector care la poziţia i conţine legătura de scor (= informaţie mutuală, $\texttt{mi}(\langle i, j \rangle) = \texttt{mi}(x = S_k[i], y = S_k[j]) = \texttt{mi}(x, y) = \log_2 \frac{f(x,y)K}{f(x,*)f(*,y)}$, unde $f(x, y)$ este frecvenţa cu care perechea $\langle x, y \rangle$ apare în corpus până la fraza curentă) minim între poziţiile i şi j (lucru valabil de la linia 8 în jos) şi care la final va conţine legăturile care alcătuiesc structura.

Algoritmul scanează fraza de la stânga la dreapta (linia 2) iar la poziţia curentă j încearcă să traseze una sau mai multe legături scanând de la dreapta la stânga (linia 3). La linia 8, un ciclu se detectează când o nouă legătură se încearcă între poziţiile i şi j iar poziţia i a vectorului *MinLink* conţine o

legătură cu un scor mai mic. O intersecţie este depistată atunci când stiva conţine în vârful ei una sau mai multe legături cu scor mai mic decât cea care se încearcă. În condiţia de la linia 8, virgula trebuie interpretată ca un operator *şi*:

$$\begin{aligned}\texttt{mi}(\langle i,j\rangle) > 0, \texttt{mi}(\mathit{MinLink}[i]), \texttt{mi}(\mathit{Stack}[s]), \forall s \Leftrightarrow \\ \texttt{mi}(\langle i,j\rangle) > 0 \wedge \\ \texttt{mi}(\langle i,j\rangle) > \texttt{mi}(\mathit{MinLink}[i]) \wedge \\ \texttt{mi}(\langle i,j\rangle) > \texttt{mi}(\mathit{Stack}[s]), \forall s\end{aligned}$$

Acest analizor nu este optimal (nu generează structura de legături cu cea mai mare informaţie mutuală) şi de asemenea nu este robust (pot exista cazuri în care să rămână cuvinte nelegate, [131, pag. 38]). Yuret oferă şi varianta optimală a analizorului dar la un cost al timpului de execuţie de $O(n^5)$ (faţă de $O(n^2)$). Pentru că algoritmul trebuie să ruleze pe texte foarte mari pentru a depista structurile de legături corecte[38], costul de timp de execuţie ridicat al analizorului optimal nu îi justifică folosirea în ce priveşte performanţele afişate.

Performanţele algoritmului suboptimal au fost evaluate pe 200 de fraze extrase din datele de antrenament (un text jurnalistic de 100 de milioane de cuvinte) şi adnotate cu legături de un expert. Testarea s-a făcut numai pe legăturile dintre cuvintele conţinut în număr de 1287 în cele 200 de fraze. Precizia analizorului a fost de aproximativ 61% iar recall-ul de aproximativ 56%, rezultate foarte bune în opinia noastră pentru un program care rulează pe texte simple, neadnotate în vreun fel cu excepţia segmentării la nivel de frază.

4.2.2 LexPar

LexPar ([40]) este un analizor de legături bazat pe reguli. Este o extensie firească a algoritmului 1 (pagina 87) care constrânge formarea de legături cu reguli sintactice specifice limbii textului procesat. În plus conţine şi un mecanism simplu de generalizare a proprietăţilor unei legături pentru a elimina inadaptabilitatea algoritmului iniţial faţă de cuvintele necunoscute.

Principalele diferenţe între algoritmul 1 şi LexPar sunt:

- LexPar rulează pe texte adnotate morfosintactic şi lematizate. Lematizarea oferă un prim nivel de generalizare pentru forma ocurentă a cuvântului contribuind la estimări mai bune ale parametrilor modelului.

[38] Algoritmul suboptimal a fost rulat pe un text jurnalistic de aproximativ 100 de milioane de cuvinte !

- LexPar calculează scorul unei legături considerând simultan lemele cuvintelor legate cât şi etichetele morfosintactice ale lor (vezi notaţiile de mai jos):

$$\texttt{mi}(\langle i,j\rangle) \stackrel{\text{def}}{=} \texttt{mi}(l_i,l_j) + \texttt{mi}(t_i,t_j)$$
$$\texttt{mi}(l_i,l_j) \stackrel{\text{def}}{=} 0,\ \textit{dacă}\ f(l_i,l_j) = 0$$
$$\texttt{mi}(t_i,t_j) \stackrel{\text{def}}{=} 0,\ \textit{dacă}\ f(t_i,t_j) = 0$$

  În cazul în care una din leme nu a fost întâlnită la antrenare, scorul legăturii este dat de perechea de etichete morfosintactice a cărei apariţie este mult mai probabilă decât cea a perechii de leme. Împreună cu lematizarea, luarea în calcul a etichetelor morfosintactice ale cuvintelor în formarea unei legături reprezintă principalul mecanism de generalizare al lui LexPar în calculul scorurilor legăturilor între cuvintele necunoscute.

- Ca şi în algoritmul 1, LexPar ia în calcul o legătură care nu produce un ciclu şi care nu încalcă proprietatea de planaritate dar în plus, LexPar nu consideră legătura care este rejectată de filtrul său sintactic[39] (vezi figura 4.8). Această filtrare are rolul de a grăbi convergenţa procesului de antrenament către MAL care aproximează structura de dependenţe a limbii date. În plus, perechile care nu pot fi relaţionate sintactic nu încarcă inutil memoria procesorului.

Algoritmul LexPar consideră o altă ordine de procesare a cuvintelor unei fraze decât scanarea de la stânga la dreapta. Principala presupunere pe care o face este aceea că cele mai multe legături se stabilesc între cuvinte adiacente iar apoi între grupuri adiacente de cuvinte legate. LexPar construieşte progresiv structura de legături a unei fraze, alcătuind grupuri de cuvinte legate de dimensiuni din ce în ce mai mari. Pentru o frază S_k (fără marcaje de început şi sfârşit), o listă de tripluri $\langle w_i, l_i, t_i\rangle$ de formă ocurentă (w), lemă (l) şi etichetă morfosintactică compatibilă MULTEXT-East (t), pseudocodul procesorului este algoritmul 2. Vom exemplifica funcţionarea acestui algoritm cât şi semnificaţia regulilor de combinare din figura 4.8 pe următorul exemplu:

(4.4) John/`John/Np` 's/`'s/St` watch/`watch/Ncns` fell/`fall/Vmis` on/`on/Sp` the/`the/Dd3` floor/`floor/Ncns` ./`./PERIOD`

[39]Prezenţa filtrului sintactic nu mai garantează o structură de graf conex a analizei de legături (vezi comentariile de la pagina 92).

Algoritmul 2 Analizorul de legături LexPar.

1: **function** LexPar(S_k)
2: $G \leftarrow \texttt{patterns}(S_k)$
3: **while true do**
4: $G_{new} \leftarrow \texttt{empty}()$
5: $i \leftarrow 0$
6: **while** $i < \texttt{sizeof}(G) - 2$ **do**
7: $lnk_{i,i+1} \leftarrow \texttt{bestlink}(G[i], G[i+1])$
8: **if** $i + 2 = \texttt{sizeof}(G)$ **then**
9: $\texttt{addgroup}(G_{new}, \texttt{makegroup}(G[i], G[i+1], lnk_{i,i+1}))$
10: **return** G_{new}
11: **end if**
12: $lnk_{i+1,i+2} \leftarrow \texttt{bestlink}(G[i+1], G[i+2])$
13: **if** $\texttt{nolink}(lnk_{i,i+1}) \wedge \texttt{nolink}(lnk_{i+1,i+2})$ **then**
14: $\texttt{addgroup}(G_{new}, G[i])$
15: $\texttt{addgroup}(G_{new}, G[i+1])$
16: $i \leftarrow i + 2$
17: **next loop**
18: **end if**
19: **if** $\texttt{mi}(lnk_{i,i+1}) > \texttt{mi}(lnk_{i+1,i+2})$ **then**
20: $\texttt{addgroup}(G_{new}, \texttt{makegroup}(G[i], G[i+1], lnk_{i,i+1}))$
21: $i \leftarrow i + 2$
22: **else**
23: $\texttt{addgroup}(G_{new}, G[i])$
24: $\texttt{addgroup}(G_{new}, \texttt{makegroup}(G[i+1], G[i+2], lnk_{i+1,i+2}))$
25: $i \leftarrow i + 3$
26: **end if**
27: **end while**
28: **if** $\texttt{sizeof}(G) = \texttt{sizeof}(G_{new})$ **then**
29: **return** G_{new}
30: **end if**
31: $G \leftarrow G_{new}$
32: **end while**
33: **end function**

```
deny left any right preposition(prep)
enforce left preposition(prep) right noun(*,*,*)
agree left determiner(*,*,*,*,*,*) right noun(*,*,*) at number{}
sequence noun(*,*,*);preposition(post);noun(*,*,*) link 1-2;1-3
```

Figura 4.8: Câteva reguli sintactice pentru engleză folosite de LexPar.

i		0	1	2	3	4	5	6
		`John`	`'s`	`watch`	`fall`	`on`	`the`	`floor`
0	`John`	×	30	45	19	5	6	10
1	`'s`	×	×	11	2	7	1	9
2	`watch`	×	×	×	29	13	10	15
3	`fall`	×	×	×	×	14	2	20
4	`on`	×	×	×	×	×	3	50
5	`the`	×	×	×	×	×	×	60
6	`floor`	×	×	×	×	×	×	×

Tabela 4.1: Memoria procesorului LexPar înainte de rularea acestuia pe exemplul 4.4.

Să considerăm de asemenea că scorul unei legături este extras direct din tabelul 4.1 (de exemplu $\texttt{mi}(\langle 0,1\rangle) = 30$) în care am dat scoruri mari legăturilor pe care algoritmul trebuie să le descopere[40].

La linia 2 vectorul iniţial de grupuri este vectorul calculat de funcţia `patterns` care foloseşte regulile ***`sequence`*** pentru a recunoaşte grupuri de cuvinte adiacente ale căror etichete morfosintactice corespund. În cazul nostru, în figura 4.8 avem o regulă ***`sequence`*** care recunoaşte secvenţa

```
John/noun(p,*,*);'s/preposition(t);watch/noun(c,n,s)
```

iar grupul se formează cu legăturile $[\langle 0,1\rangle, \langle 0,2\rangle]$. Vectorul G va conţine grupurile $\{[\langle 0,1\rangle, \langle 0,2\rangle], [3], [4], [5], [6]\}$.

Liniile 7 şi 12 sunt liniile în care funcţia `bestlink` caută cea mai bună legătură care să unească grupurile $G[i], G[i+1]$ sau $G[i+1], G[i+2]$. Căutarea se face respectând proprietăţile de planaritate şi aciclicitate ale noului grup care se poate forma şi constrângerile dictate de filtrul sintactic[41]. Depinzând de scorurile legăturilor $lnk_{i,i+1}$ şi $lnk_{i+1,i+2}$ (linia 19) se formează un grup care se adaugă la noul vector de grupuri G_{new}. Funcţia `makegroup` primeşte ca argument cele două grupuri care se vor uni şi legătura care le uneşte şi întoarce noul grup astfel format care cu funcţia `addgroup` se adaugă la noul vector de grupuri. Întorcându-ne la exemplul nostru, evoluţia vectorului G este:

[40] Vrem să vedem că algoritmul trasează legăturile corect dacă acestea au scorul maxim.

[41] Fiecare legătură care poate forma un nou grup este introdusă în memoria analizorului.

- G : $\{[\langle 0,1\rangle, \langle 0,2\rangle, \langle 2,3\rangle], [4], [\langle 5,6\rangle]\}$, legătura $\langle 3,4\rangle$ este respinsă de regula ***deny*** din figura 4.8 iar legătura $\langle 5,6\rangle$ este permisă de regula de acord gramatical ***agree***;

- G : $\{[\langle 0,1\rangle, \langle 0,2\rangle, \langle 2,3\rangle], [\langle 4,6\rangle, \langle 5,6\rangle]\}$, legătura $\langle 4,6\rangle$ a fost impusă de regula ***enforce*** ceea ce înseamnă că aceasta este considerată cea mai bună indiferent de scorul pe care îl are;

- G : $\{[\langle 0,1\rangle, \langle 0,2\rangle, \langle 2,3\rangle, \langle 3,6\rangle, \langle 4,6\rangle, \langle 5,6\rangle]\}$, vector care se obţine la linia 10 şi care conţine structura de legături a exemplului 4.4.

Algoritmul nu garantează o analiză robustă din cauză că filtrul sintactic poate conţine reguli care să interzică legarea a două grupuri într-un caz în care aceasta ar fi trebuit să fie permisă. Regulile au fost scrise în marea lor majoritate de către autor studiind comportamentul analizorului pe diferite texte. Există reguli atât pentru română cât şi pentru engleză care rejectează legături care sunt greşite în cele mai multe cazuri dar care rejectează de asemenea aceleaşi tipuri de legături care însă sunt corecte în alte cazuri (izolate)[42]. Linia 28 garantează terminarea algoritmului atunci când nu se mai adaugă nicio legătură la structura de legături (practic atunci când numărul de grupuri rămâne neschimbat).

LexPar a fost antrenat pe corpusurile NAACL, 1984, Republica, Ziare şi SemCor2.0 (1291736 de unităţi lexicale şi punctuaţie) pentru construcţia modelului de atracţie lexicală românesc şi pe corpusurile 1984 şi SemCor2.0[43] (în total 893650 de unităţi lexicale împreună cu punctuaţia) pentru modelul limbii engleze[44]. Pentru că nu dispunem de un corpus de referinţă adnotat cu legături, vom folosi analizorul sintactic de dependenţe MiniPar ([54]) pe partea de engleză a corpusului paralel SemCor2.0 şi vom compara analizele sintactice produse cu structurile de legături generate de LexPar pe acelaşi text. În tabelul 4.2 se află precizia, recall-ul şi f-measure pentru legăturile găsite de LexPar considerând adnotarea sintactică MiniPar ca referinţă.

Precizia analizorului sintactic MiniPar este de aproximativ 89% ([54]) şi pentru că LexPar găseşte 68% din legăturile între cuvinte conţinut ale lui MiniPar, nu greşim foarte mult dacă presupunem că *toate* aceste legături sunt corecte (au fost generate independent de două analizoare diferite)[45]. Cu

[42]Am preferat să păstrăm aceste reguli pentru că performanţa pe ansamblu era mai bună cu ele decât fără ele.

[43]Corpusul complet în engleză nu numai partea corespunzătoare traducerii în română.

[44]Au fost utilizate aceste corpusuri la antrenare pentru că adnotarea morfosintactică şi lematizarea au fost verificate manual eliminându-se astfel erorile de legare care s-ar fi datorat greşelior de adnotare morfosintactică sau lematizare.

[45]De asemenea presupunem că LexPar nu generează legături corecte pe care MiniPar nu le găseşte.

	Toate legăturile	Cuvinte conţinut
P(%)	53.692%	49.819%
R(%)	67.838%	68.209%
F(%)	59.941%	57.582%

Tabela 4.2: Gradul de acord între LexPar şi MiniPar pe SemCor2.0.

datele din tabelul 4.2 putem aprecia că recall-ul real al analizorului LexPar este de $0.68209 \times 0.89 = 60.706\%$ un recall mai bun decât cel al analizorului lui Yuret (56%). În plus, există o diferenţă semnificativă între dimensiunile textelor de antrenament: Lexpar, 10^6 de unităţi lexicale, Yuret, 10^8 de unităţi lexicale de unde se poate trage concluzia că filtrul sintactic măreşte considerabil viteza de convergenţă a analizoarelor de legături.

Algoritmul LexPar este de asemenea robust în proporţie de 97.830%, procent care evidenţiază gradul de conectare al unităţilor lexicale în întreg textul: dacă într-o frază întâlnim n unităţi lexicale din care sunt conectate numai x, frecvenţa de conectare creşte cu $\frac{x}{n}$. Pentru a obţine procentul de conectare, suma frecvenţelor de conectare se împarte la 8276 de fraze câte sunt în textul englezesc. Diferenţa de 2.17% reprezintă o aproximaţie a procentului mediu de unităţi lexicale neconectate dintr-o frază. Aceste unităţi lexicale rămân neconectate din cauza filtrului sintactic (vezi nota de subsol 42).

4.3 SynWSD

În secţiunea 4.1 am văzut că pentru fiecare structură sintactică de suprafaţă $SSyntR$ există o structură sintactică de adâncime $DSyntR$ din care poate fi obţinut apoi graful semantic $SemR$. În transformarea $SSyntR \leftrightarrow DSyntR \leftrightarrow SemR$[46] înţelesurile lexemelor sunt prezente în $SemR$. De exemplu, graful semantic $SemR$ pentru exemplul

(4.5) Dick asked Susan to visit him.

este redat în figura 4.10 împreună cu transformarea de la $DSyntR$ (arcele punctate desemnează coreferinţa elementelor şi nu fac parte din relaţia sintactico-semantică de la acest nivel). În $SemR$, fiecare lexem este indexat de înţelesul său (extras din dicţionarul de referinţă, PWN2.0 în cazul de faţă) iar graful reprezintă schematizarea functorului din figura 4.9.

[46]Vezi [61, pag. 73,81] pentru regulile de transformare.

$\text{ASK}_{ask(\mathbf{2})}$(
$\text{DICK}_{male(\mathbf{2})}$,
$\text{SUSAN}_{female(\mathbf{2})}$,
$\text{VISIT}_{visit(\mathbf{3})}$(
$\text{SUSAN}_{female(\mathbf{2})}$,
$\text{DICK}_{male(\mathbf{2})}$
)
)

Figura 4.9: Functor care exprimă înţelesul propoziţiei 4.5.

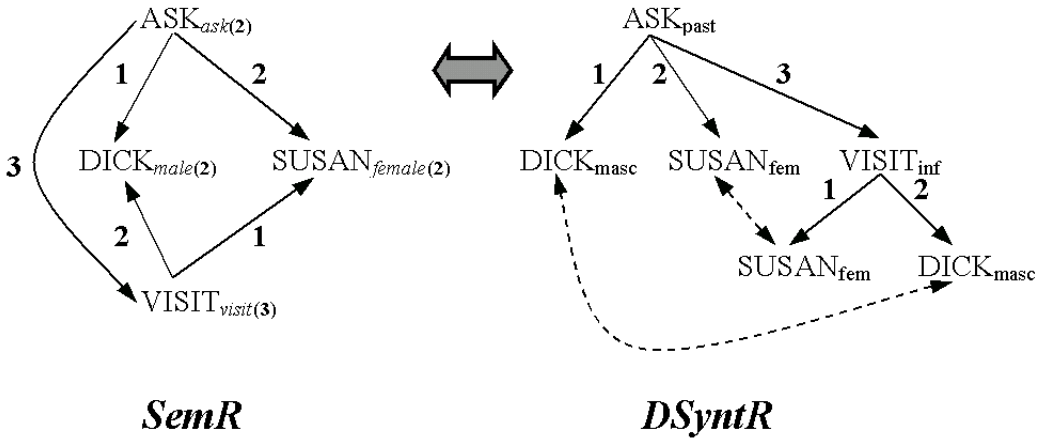

Figura 4.10: O corespondenţă între $SemR$ şi $DSyntR$.

Dacă s-ar pune problema generării automate a structurii $SemR$, atunci procesul de DSA ar trebui implementat pe transformarea $DSyntR \rightarrow SemR$ având structura $DSyntR$ la intrare. Cum nu dispunem de un algoritm de construcţie a acestei structuri şi pentru că dispunem doar de o aproximare a structurii $SSyntR$, vom încerca în cele ce urmează să construim procesul de DSA pe structura de legături a unei fraze. Această structură aproximează $SSyntR$ prin eliminarea orientării relaţiilor sintactice şi a denumirii acestora dar structura arborescentă se poate păstra prin alegerea primului cuvânt al frazei ca rădăcină a arborelui, considerarea tuturor cuvintelor legate de rădăcină ca dependenţi ai rădăcinii, ş.a.m.d.

În prezenţa structurii de legături a unei fraze, contextul de apariţie al unui cuvânt capătă un plus de explicitare faţă de definirea acestuia ca fereastră de cuvinte. Astfel, contextul unui cuvânt w este specificat de mulţimea de legături care-l conţine şi în consecinţă, înţelesul acestuia va fi determinat *direct* numai de cuvintele care sunt legate la el şi *indirect*, de celelalte cuvinte.

4.3.1 Descrierea algoritmului

SynWSD (un prototip al acestui algoritm a fost prezentat în [38]) este un algoritm de dezambiguizare semantică automată neasistată care utilizează structura de legături a unei fraze ca pe o specificare a dependenţelor înţelesurilor cuvintelor unele de altele. Dacă S este o frază şi w_i şi w_j două cuvinte din S, ipoteza pe care se bazează acest algoritm este următoarea: dacă w_i îl atrage lexical pe w_j atunci anumite înţelesuri ale lui w_i vor *atrage semantic* (într-un sens ce va fi definit ulterior) anumite înţelesuri ale lui w_j. Altfel spus, anumite înţelesuri vor avea o afinitate de combinare mai mare cu altele decât cu restul.

Algoritmul are două faze:

1. **antrenare**: estimarea valorilor atracţiei semantice folosind ca inventar de sens o reţea semantică lexicală (PWN2.0 sau ROWN2.0 în cazul experimentelor noastre) şi un corpus adnotat cu legături de LexPar;

2. **dezambiguizare**: pentru o frază S cu structura de legături L şi cu modelul M creat în pasul anterior, găseşte cea mai bună combinaţie de înţelesuri ale cuvintelor conţinut din S care să maximizeze atracţia semantică pe S.

Să considerăm fraza S ca un vector de tupluri $\langle w_i, t_i, l_i \rangle, 0 \leq i < |S|$ de formă ocurentă (w), etichetă morfosintactică (t) şi lemă (l) şi structura ei de legături L ca un vector de perechi de indecşi legaţi $\langle i, j \rangle, 0 \leq i < j < |S|$. De asemenea, să considerăm funcţiile $ili(l_i, t_i)$, $sumo(l_i, t_i)$ şi $dom(l_i, t_i)$ care

au ca valori mulţimea de ILI-uri, mulţimea de concepte SUMO şi respectiv mulţimea de domenii IRST corespunzătoare literalului l_i cu categoria gramaticală a etichetei t_i din reţelele semantice lexicale PWN2.0 şi ROWN2.0. Dacă l_i nu se află în vreun sinset din reţeaua semantică lexicală (dacă este lema unui cuvânt funcţional de exemplu), atunci valoarea oricărei funcţii de mai sus este chiar l_i. Dacă l_i se află într-un sinset care nu are un concept SUMO sau un domeniu IRST asociat, atunci valorile funcţiilor *sumo* şi *dom* sunt egale cu valoarea implicită `default`. De exemplu, valorile funcţiilor de mai sus calculate pe ROWN2.0 pentru substantivul "*floare*" sunt următoarele:

$$
\begin{aligned}
\mathit{ili}(\mathit{floare}, \mathrm{N}) &= \{\texttt{ENG20-10924345-n}, \texttt{ENG20-10792063-n}, \texttt{ENG20-10924920-n}\} \\
\mathit{sumo}(\mathit{floare}, \mathrm{N}) &= \{\texttt{Flower}, \texttt{Plant}, \texttt{FloweringPlant}\} \\
\mathit{dom}(\mathit{floare}, \mathrm{N}) &= \{\texttt{plants}\}
\end{aligned}
$$

Faza de antrenare

SynWSD se antrenează pe un corpus adnotat morfosintactic, lematizat şi analizat cu LexPar şi va construi câte un model de atracţie semantică pentru fiecare inventar de sensuri (ILI, SUMO sau IRST). Procesul de construcţie a modelului este acelaşi oricare ar fi inventarul de sensuri şi de aceea în continuare vom descrie numai modelul de atracţie semantică pentru ILI. Acest model se creează parcurgând pe rând frazele S din corpus iar pentru o astfel de frază, se efectuează următorii paşi:

1. pentru fiecare legătură $\langle i, j \rangle \in L$ calculează valorile $I_i = \mathit{ili}(l_i, t_i)$ şi $I_j = \mathit{ili}(l_j, t_j)$;

2. pentru fiecare ILI $c_a \in I_i$ generalizează valoarea acestuia şi depune rezultatul în mulţimea G_i (analog pentru fiecare ILI $c_b \in I_j$). Generalizarea se aplică doar în cazul inventarului de sens ILI şi este necesară pentru a reduce numărul de parametri ai modelului. În funcţie de categoria gramaticală, procedeul de generalizare urmează paşii:

   - substantive, verbe: alege primul ILI g_a din ierarhia de hipernimi care subsumează exact un înţeles al literalului l_i. Această condiţie se impune pentru a putea apoi extrage neambiguu înţelesul c_a al literalului l_i. De exemplu, în figura 4.11, conceptele care generalizează sensurile 1, 3, şi 6 ale substantivului "*floare*" sunt încercuite cu linie punctată (săgeţile verticale indică relaţia de hipernimie). Se observă că sensul 3 nu poate fi generalizat deoarece hipernimul său ar subsuma şi sensul numărul 6.

- adjective: dacă c_a se află într-o clasă de similaritate, alege ca generalizare înţelesul reprezentativ al clasei g_a; dacă nu, $g_a = c_a$;
- adverbe: $g_a = c_a$.

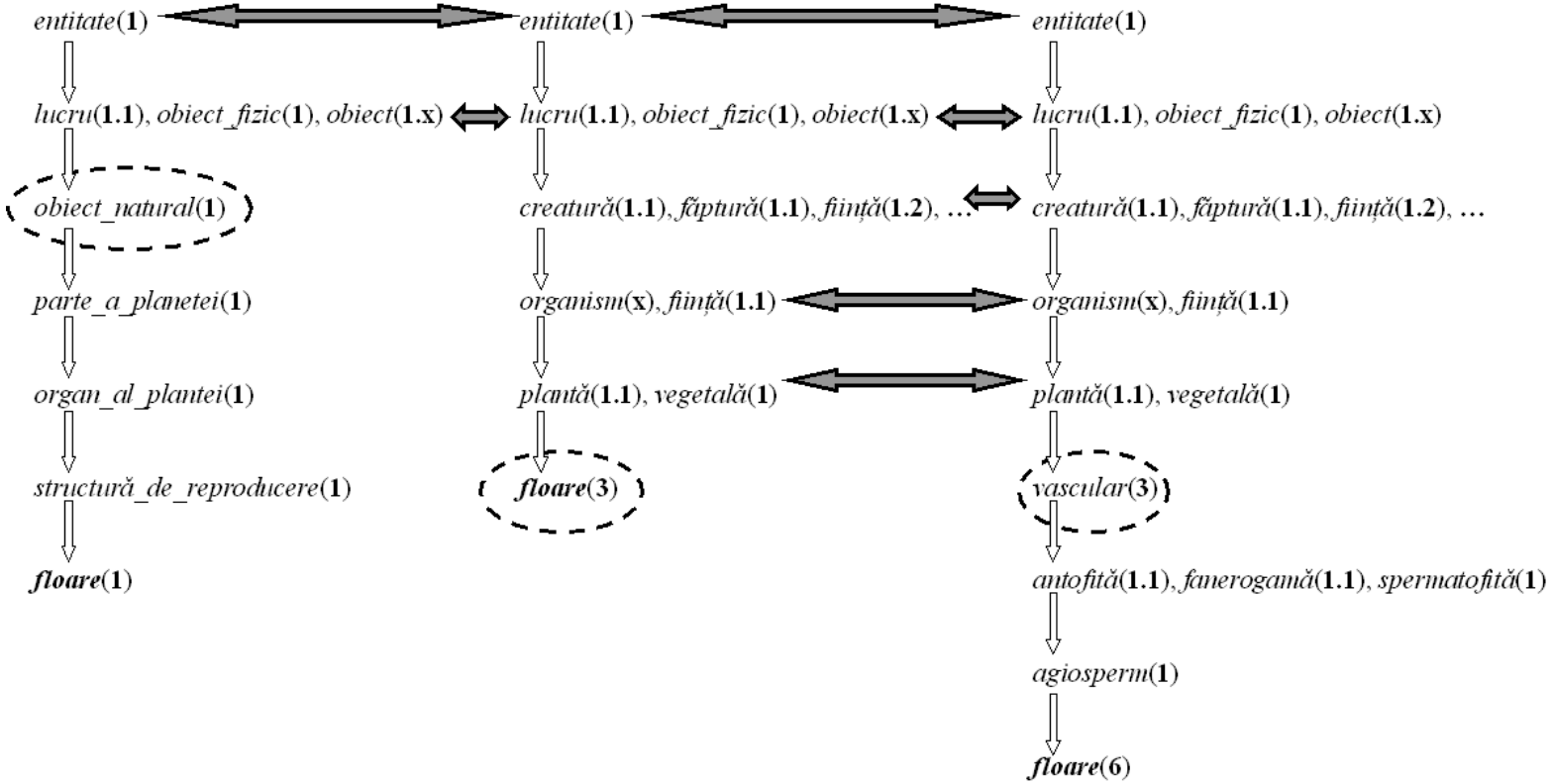

Figura 4.11: Exemplu de generalizare pentru substantivul "*floare*".

3. pentru fiecare pereche de ILI $\langle c_a, c_b \rangle$ (parametru al modelului) din produsul cartezian $G_i \otimes G_j$ incrementează frecvenţele $f(\langle c_a, c_b \rangle)$, $f(\langle c_a, * \rangle)$, $f(\langle *, c_b \rangle)$ şi $f(\langle *, * \rangle)$ în model. Notaţia $*$ semnifică "orice etichetă de sens" iar frecvenţa perechii $\langle *, * \rangle$ este egală cu numărul total de perechi de etichete semantice care au apărut în corpusul de antrenare până la pasul curent. De asemenea, $f(\langle c_a, * \rangle)$ este numărul de apariţii ale etichetei semantice c_a în stânga unei legături iar $f(\langle *, c_b \rangle)$ este numărul de apariţii ale lui c_b în dreapta unei legături (toate legăturile sunt perechi $\langle i, j \rangle$ pentru care $i < j$).

Modelul astfel format este o colecţie de frecvenţe ale perechilor de etichete semantice posibile ale cuvintelor care determină o legătură şi este identic din acest punct de vedere cu memoria analizorului de legături.

Faza de dezambiguizare

Este responsabilă de găsirea configuraţiei de înţelesuri care conferă un scor de atracţie semantică maxim pentru fraza S. În această privinţă, SynWSD diferă de algoritmii de DSA care atribuie înţelesuri cuvintelor unei fraze prin optimizarea locală, pentru fiecare cuvânt în parte, a parametrilor de clasificare.

Faza de dezambiguizare este de asemenea complet independentă de limbă în sensul că aceasta are nevoie de fraza S împreună cu structura ei de legături L şi de modelul de atracţie semantică M construit în faza anterioară. Pentru ca parametrii specifici frazei S să fie definiţi în momentul dezambiguizării, este *necesar* ca antrenarea să se fi produs şi pe fraza S. Această condiţie nu trebuie interpretată însă ca o favorizare a fazei de dezambiguizare (este vorba de evaluarea pe datele de test care nu poate fi obiectivă întrucât ar trebui testată abilitatea programului de învăţare automată de a generaliza pe date necunoscute) pentru că în cazul SynWSD, antrenarea nu se produce pe corpusuri adnotate cu etichete semantice (SynWSD fiind astfel un algoritm de DSA neasistată) iar din această cauză nu există premiza favorizării (procesul de antrenare nu "ştie" apriori care sunt etichetele semantice pe care faza de dezambiguizare ar trebui să le furnizeze).

Vom descrie algoritmul de dezambiguizare pentru inventarul de sensuri dat de ILI. În această fază, SynWSD parcurge următorii paşi:

1. pas identic cu pasul 1 (pagina 96) din faza de antrenament;

2. pas de asemenea identic cu pasul 2 din faza de antrenament;

3. consideră structura de legături L a frazei S ca pe un arbore cu rădăcina

în poziţia 0 a frazei[47]. Parcurge recursiv acest arbore în adâncime şi formează o listă V de indecşi prin inserarea nodurilor arborelui când acestea sunt vizitate.

4. pentru fiecare index $k \in V, 0 \leq k < |S|$, asociază acestuia mulţimea G_k de ILI calculată anterior. Prin $V[i], 0 \leq i < |V|$ vom înţelege mulţimea de ILI corespunzătoare poziţiei $k, 0 \leq k < |S|$ din fraza S;

5. pe secvenţa de stări $V[i]$, aplică algoritmul Viterbi ([57, pag. 332]) fără emisie de simboluri şi utilizând una din următoarele măsuri de atracţie semantică în locul probabilităţilor de tranziţie ($c_a \in V[i], c_b \in V[i+1]$):

   - coeficientul DICE (măsură simetrică):

$$\begin{aligned}
\texttt{dice}(c_a, c_b) &= \texttt{dice}(c_b, c_a) = \frac{2p(c_a, c_b)}{p(c_a) + p(c_b)} \\
&= \frac{2\frac{f(\langle c_a, c_b\rangle) + f(\langle c_b, c_a\rangle)}{f(\langle *, *\rangle)}}{\frac{f(\langle c_a, *\rangle) + f(\langle *, c_a\rangle)}{f(\langle *, *\rangle)} + \frac{f(\langle c_b, *\rangle) + f(\langle *, c_b\rangle)}{f(\langle *, *\rangle)}} = \\
&= \frac{2(f(\langle c_a, c_b\rangle) + f(\langle c_b, c_a\rangle))}{f(\langle c_a, *\rangle) + f(\langle *, c_a\rangle) + f(\langle c_b, *\rangle) + f(\langle *, c_b\rangle)}
\end{aligned}$$

   - probabilitatea $\texttt{prob}(c_b|c_a)$ (măsură asimetrică):

$$\begin{aligned}
\texttt{prob}(c_b|c_a) &= p(c_b|c_a) = \frac{p(c_a, c_b)}{p(c_a)} = \\
&= \frac{f(\langle c_a, c_b\rangle) + f(\langle c_b, c_a\rangle)}{f(\langle c_a, *\rangle) + f(\langle *, c_a\rangle)}
\end{aligned}$$

   - informaţia mutuală punct la punct[48] (măsură simetrică):

$$\begin{aligned}
\texttt{mi}(c_a, c_b) &= \texttt{mi}(c_b, c_a) = \log_2 \frac{p(c_a, c_b)}{p(c_a)p(c_b)} = \\
&= \log_2 \frac{f(\langle *, *\rangle)(f(\langle c_a, c_b\rangle) + f(\langle c_b, c_a\rangle))}{(f(\langle c_a, *\rangle) + f(\langle *, c_a\rangle))(f(\langle c_b, *\rangle) + f(\langle *, c_b\rangle))}
\end{aligned}$$

   - scorul Log-Likelihood $\texttt{ll}(c_a, c_b)$ ([71]) de asemenea o măsură simetrică.

[47] Toţi indecşii care apar în legături care conţin poziţia 0 devin dependenţi ai acesteia iar acest proces continuă recursiv până când se epuizează toţi indecşii frazei.

[48] "Pointwise mutual information" în engleză.

6. extragerea prin căutarea cu revenire[49] a *tuturor* configuraţiilor semantice ale lui S de scor maxim. Se obţine o listă D de mulţimi de etichete semantice astfel încât $D[i] \subseteq V[i], 0 \leq i < |V|$. Dacă $c_a \in D[i]$ este un ILI care nu conţine substantivul/verbul l_k, se caută sinsetul hiponim al lui c_a care-l conţine pe l_k (se aplică operaţia inversă generalizării) şi se înlocuieşte c_a cu acesta. Dacă l_k este adjectiv, se caută sinsetul similar cu c_a care-l conţine pe l_k şi se face de asemenea înlocuirea.

În pasul 6 al fazei de dezambiguizare există cazuri în care două sau mai multe configuraţii semantice au acelaşi scor de atracţie semantică. Acest lucru se traduce prin existenţa mai multor etichete semantice la poziţia $V[i]$ prin care calea optimă trece şi pentru că nu dispunem de un mecanism de a alege o singură etichetă, le vom considera pe toate ca rezultat al dezambiguizării. Acest lucru nu reprezintă neapărat o limitare a algoritmului pentru că, în cazul unui inventar de sensuri cu granularitate foarte mică cum este ILI, pentru un anume cuvânt pot exista mai multe etichete semantice aplicabile în context (diferenţierea sensurilor este dificilă chiar şi pentru experţi). De exemplu, în fraza

(4.6) The jury said it did find that many of Georgia's registration and election laws "are outmoded or inadequate and often ambiguous".

din SemCor2.0, SynWSD a atribuit sensurile 1 şi 2 literalului "*ambiguous*" din 3 posibile cu următoarele definiţii (din PWN2.0):

equivocal(1), *ambiguous*(**1**):	open to two or more interpretations; or of uncertain nature or significance; or (often) intended to mislead.
ambiguous(**2**):	having more than one possible meaning.
ambiguous(3):	having no intrinsic or objective meaning; not organized in conventional patterns.

Adnotarea de referinţă specifică numai sensul numărul 2 ca fiind corect.

4.3.2 Evaluări

SynWSD s-a antrenat pe aceleaşi date ca şi LexPar (vezi pagina 92) şi a fost evaluat pe SemCor2.0 ca şi WSDTool pentru a permite o comparaţie a performanţelor celor doi algoritmi.

Pe lângă măsurile de atracţie semantică cunoscute, definim în continuare un combinator care produce o nouă adnotare din adnotările obţinute cu `dice`, `prob`, `mi` şi `ll` (vezi pagina 95 şi pasul 6 al algoritmului pentru notaţii):

[49] "Backtracking" în engleză.

		Engleză				Română			
		P(%)	**R(%)**	**F(%)**	**S/C**	**P(%)**	**R(%)**	**F(%)**	**S/C**
ILI	dice	46.985	46.874	46.930	1.477	40.627	40.122	40.373	1.769
	prob	46.459	46.349	46.404	1.429	41.588	41.070	41.327	1.787
	mi	47.859	47.746	47.803	1.729	41.204	40.685	40.942	2.084
	ll	42.977	42.876	42.927	1.239	36.170	35.720	35.943	1.384
	int	**69.773**	26.638	38.556	1.163	**59.845**	22.214	32.401	1.373
	majv	43.952	43.848	43.900	1.285	37.212	36.747	36.978	1.493
	union	68.164	**68.001**	**68.082**	2.805	59.647	**58.896**	**59.269**	3.353
SUMO	dice	50.246	49.958	50.102	1.237	40.971	40.472	40.720	1.234
	prob	49.688	49.408	49.548	1.169	41.954	41.442	41.696	1.214
	mi	57.831	57.236	57.532	1.334	51.188	50.570	50.877	1.413
	ll	47.249	46.979	47.114	1.096	39.550	39.065	39.306	1.120
	int	**74.067**	32.503	45.180	1.009	**69.405**	25.609	37.413	1.008
	majv	48.135	47.915	48.025	1.087	39.566	39.084	39.323	1.092
	union	73.505	**73.165**	**73.335**	2.140	66.708	**65.901**	**66.302**	2.363
IRST	dice	78.042	77.658	77.849	1.090	77.461	76.516	76.986	1.089
	prob	76.351	75.974	76.162	1.018	76.685	75.749	76.214	1.032
	mi	75.437	74.983	75.210	1.274	65.235	64.440	64.835	1.276
	ll	75.735	75.359	75.546	1.010	76.140	75.210	75.672	1.004
	int	88.399	59.352	71.020	1.002	87.368	50.449	63.963	1.001
	majv	76.371	76.026	76.198	1.016	76.612	75.677	76.142	1.020
	union	**91.413**	**91.005**	**91.209**	1.621	**90.305**	**89.202**	**89.750**	1.719

Tabela 4.3: Rezultatele algoritmului SynWSD pe SemCor2.0.

- adnotarea prin *intersecţie* (`int`): dacă cuvântul $w_i, 0 \leq i < |S|$ a primit mulţimile de înţelesuri $D_{\texttt{dice}}[i], D_{\texttt{prob}}[i], D_{\texttt{mi}}[i]$ şi $D_{\texttt{ll}}[i]$ corespunzătoare execuţiei algoritmului cu măsurile de atracţie semantică `dice`, `prob`, `mi` şi `ll`, atunci $D_{\texttt{int}}[i] = \bigcap_{m \in \{\texttt{dice},\texttt{prob},\texttt{mi},\texttt{ll}\}} D_m[i]$;
- adnotarea prin *reuniune* (`union`): în condiţiile de la intersecţie, adnotarea lui w_i este $D_{\texttt{union}}[i] = \bigcup_{m \in \{\texttt{dice},\texttt{prob},\texttt{mi},\texttt{ll}\}} D_m[i]$;
- adnotarea prin *vot majoritar* (`majv`): fie $M = \bigodot_{m \in \{\texttt{dice},\texttt{prob},\texttt{mi},\texttt{ll}\}} D_m[i]$ lista obţinută din concatenarea mulţimilor $D_{\texttt{dice}}[i]$, $D_{\texttt{prob}}[i]$, $D_{\texttt{mi}}[i]$ şi $D_{\texttt{ll}}[i]$. Se păstrează etichetele de sens care au frecvenţa de apariţie maximă în M şi se depun în $D_{\texttt{majv}}[i]$.

În tabelul 4.3 sunt rezumate rezultatele algoritmului SynWSD pentru fiecare măsură a atracţiei semantice. Coloanele **P**, **R** şi **F** conţin măsurile procentuale medii[50] ale preciziei, recall-ului şi respectiv ale combinaţiei aces-

[50]Medierea s-a făcut pe rezultatele obţinute pe fiecare fişier al corpusului SemCor2.0 numai pentru precizie şi recall iar f-measure a fost recalculat cu aceste valori medii.

tora, f-measure, calculate cu relaţiile următoare:

$$P = \frac{N_{A,G}}{N_A}, R = \frac{N_{A,G}}{N_G}, R = \frac{2PR}{P+R}$$

unde

- A este lista de adnotări produsă de SynWSD pentru ocurenţele $i, 1 \leq i \leq N_A$ (N_A este numărul de ocurenţe adnotate de SynWSD) ale cuvintelor conţinut din SemCor2.0. La poziţia i în A se află lista de etichete semantice D_i^A atribuite de algoritm pentru ocurenţa respectivă;
- G este lista de adnotări de referinţă din SemCor2.0 pentru ocurenţele $j, 1 \leq j \leq N_G$ (N_G este numărul de ocurenţe adnotate de experţi în SemCor2.0) ale cuvintelor conţinut. La poziţia j în G se află lista de etichete semantice D_j^G;
- $N_{A,G}$ este numărul de ocurenţe în care SynWSD a atribuit eticheta sau etichetele semantice din adnotarea de referinţă SemCor2.0 (deci numarul de adnotări corecte). Acest număr se poate calcula în două feluri:
  - evaluarea *relaxată*: fie $C = D_i^A \cap D_j^G$. Dacă $C \neq \emptyset$, se adaugă 1 la $N_{A,G}$ (se consideră că SynWSD a găsit sensul corect chiar dacă $|D_i^A| > 1$);
  - evaluarea *strictă*: se adaugă $\frac{|C|}{|D_i^A|}$ la $N_{A,G}$. Altfel spus, se depunctează algoritmul dacă acesta atribuie mai multe etichete semantice: de exemplu, dacă SynWSD a găsit două etichete semantice din care numai una este în adnotarea de referinţă, $N_{A,G}$ creşte cu 0.5 în loc de 1.

Evaluarea relaxată favorizează precizia algoritmului iar cea strictă defavorizează recall-ul pentru că numărul de adnotări corecte ar trebui să crească cu 1 indiferent de dimensiunea mulţimii D_i^A [51]. În tabelul 4.3 am folosit evaluarea relaxată pentru inventarul de sensuri ILI şi evaluările stricte pentru SUMO şi IRST. Valorile subliniate reprezintă maximele atinse de algoritm rulând cu una din cele patru măsuri introduse în pasul 5 al său iar valorile îngroşate sunt maximele absolute pe inventarul de sensuri considerat (în anexa C se prezintă rezultatele detaliate ale algoritmului pe fiecare fişier al corpusului numai pentru valorile subliniate.). Coloana **S/C** indică numărul mediu de etichete semantice pe ocurenţă, număr care este egal cu $\frac{\sum_{i=1}^{N_A} |D_i^A|}{N_A}$.

[51] Evident, doar dacă $C \neq \emptyset$.

		Engleză		Română	
		P(%)	S/C	P(%)	S/C
ILI	**WSDTool**	70.217	1	53.478	1
	SynWSD	69.773	1.163	59.845	1.373
SUMO	**WSDTool**	76.788	1	65.095	1
	SynWSD	74.067	1.009	69.405	1.008
IRST	**WSDTool**	87.636	1.092	85.015	1.11
	SynWSD	88.399	1.002	87.368	1.001

Tabela 4.4: Comparaţia preciziilor algoritmilor WSDTool şi SynWSD (cu combinatorul `int`).

În ce priveşte performanţa algoritmului SynWSD comparativ cu cea a algoritmului WSDTool, apreciem că SynWSD (cu combinatorul `int`) atinge precizii comparabile (în unele cazuri chiar mai bune) cu cele ale lui WSDTool (luând în calcul şi numărul mediu de etichete semantice pe cuvânt) aşa cum se poate vedea în tabelul 4.4. Combinatorul `int` favorizează precizia algoritmului SynWSD în detrimentul recall-ului întrucât o ocurenţă va primi etichetele semantice găsite independent (prin rulările cu diferitele măsuri de atracţie semantică) care astfel sunt corecte cu o probabilitate mare.

Combinatorul `union` are un comportament complementar combinatorului `int` prin faptul că asigură creşterea recall-ului în defavoarea preciziei. Chiar dacă valorile recall-ului şi f-measure sunt comparabile cu cele obţinute de WSDTool (pentru ambele limbi, în unele cazuri chiar mai bune), trebuie să remarcăm puterea de decizie a lui WSDTool care este net superioară celei a lui SynWSD: în timp ce WSDTool are un număr mediu de sensuri pe cuvânt de 1, SynWSD ajunge şi la 3 pentru română în cazul inventarului de sens ILI.

Rezultatele obţinute de SynWSD pe SemCor2.0 confirmă şi ele faptul că cu cât inventarul de sensuri are o granularitate mai mare, cu atât sarcina algorimului devine mai uşoară. Pe de altă parte, observăm că cu cât inventarul de sensuri are o granularitate mai mică (şi deci o dimensiune mai mare), rezultatele algoritmului scad. SynWSD este dependent de performanţele analizorului de legături LexPar iar acestea la rândul lor de *dimensiunea textului de antrenament*. De asemenea, modelul de atracţie semantică pentru un inventar de sensuri bogat cere la rândul lui un text de antrenament de dimensiune mare pentru a-şi putea estima parametrii. În concluzie, credem că

	Inventar	Ocurenţe	Performanţe		
			P(%)	R(%)	F(%)
SensEval–1	Hector	8448	61.60	60.50	61.04
SensEval–2	WordNet 1.7	2473	69.00	69.00	69.00
SensEval–3	WordNet 1.7.1	2081	65.10	65.10	65.10
SynWsd (`int`)	WordNet 2.0	79595	69.77	26.63	38.55
WsdTool	WordNet 2.0	79595	70.21	66.88	68.50

Tabela 4.5: WSDTool şi SynWSD (cu combinatorul `int`) şi cei mai buni algoritmi de DSA din SensEval pentru limba engleză.

dimensiunea textului de antrenament are consecinţe directe (măsurabile ca atare prin indicatorii de performanţă) asupra comportamentului algoritmului SynWSD.

Încheiem această secţiune a evaluărilor precizând clasarea[52] performanţelor algoritmilor SynWSD şi WSDTool printre cele ale celor mai buni algoritmi prezenţi la concursurile de dezambiguizare semantică automată din prestigioasa serie SensEval[53]. Până în prezent s-au ţinut trei asemenea competiţii din care selectăm rezultatele corespunzătoare temei de concurs "English: All Words, fine-grained"[54] în care se cerea dezambiguizarea tuturor ocurenţelor cuvintelor conţinut din textele de test (SensEval–1 ([49])[55], SensEval–2[56] şi SensEval–3 ([96])). În tabelul 4.5 coloana **Inventar** precizează inventarul de sensuri folosit de algoritmii de dezambiguizare (pentru Hector vezi [49]) iar coloana **Ocurenţe** indică numărul de ocurenţe ale cuvintelor conţinut dezambiguizate. SynWSD are cea mai bună precizie dar în schimb cel mai slab recall[57] şi apreciem că WSDTool este superior oricărui algoritm din tabelul 4.5, demonstrând încă o dată că traducerea este o formă foarte eficientă de specificare a contextului de apariţie al unui cuvânt.

[52]O comparaţie directă nu este disponibilă datorită datelor de test şi a inventarelor de sens care sunt diferite de cele folosite în această lucrare.

[53]Vezi adresa de Internet `http://www.senseval.org/`.

[54]Pentru că am rezolvat aceeaşi problemă cu WSDTool şi SynWSD pe SemCor2.0.

[55]`www.itri.brighton.ac.uk/events/senseval/ARCHIVE/RESULTS/senseval.html`.

[56]`http://193.133.140.102/senseval2/Results/all_graphs.htm`.

[57]Cei mai buni algoritmi de DSA din SensEval sunt algoritmi care implementează metode de DSA asistată. SynWSD şi WSDTool sunt metode de DSA neasistată iar metodele de DSA asistată au de regulă performanţe mai bune (dar nu sunt scalabile).

Capitolul 5

Concluzii

Lucrarea de faţă şi-a propus să prezinte algoritmii cu care cititorul să poată construi un sistem de dezambiguizare semantică automată care să fie capabil să proceseze textele de la forma lor primară, aşa cum apar ele în publicaţiile electronice[1]. Pentru a se putea rula un algoritm de DSA pe un text, acesta are nevoie de preprocesare: segmentare la nivel de frază, cuvânt, adnotare morforsintactică şi lematizare. Dacă algoritmul rulează pe texte paralele, avem nevoie de asemenea şi de aliniere lexicală în vederea extragerii echivalenţilor de traducere.

Dezambiguizarea semantică automată este şi ea ca şi adnotarea morfosintactică sau analiza sintactică o procesare intermediară a textelor utilă unor aplicaţii de PLN cum ar fi traducerea automată, sistemele de întrebare-răspuns, sau aplicaţiile de înţelegere a limbajului natural. O observaţie importantă în ce priveşte utilitatea metodelor de DSA este aceea că pentru aplicaţii diferite, trebuie alese inventarele de sens care să asigure nivelul de granularitate dorit concomitent cu performanţe bune pe acest nivel de granularitate. De exemplu, pentru traducerea automată, un algoritm de DSA trebuie să aleagă echivalentul de traducere potrivit al cuvântului de tradus în contextul său de apariţie iar o metodă de DSA care rulează pe nivelul de granularitate ILI din WordNet poate fi forţată să aleagă (inutil) între concepte care se lexicalizează identic în limba ţintă. În acest caz "inventarul de sensuri" trebuie să fie un dicţionar bilingv din care algoritmul să selecteze un echivalent de traducere.

Una din dificultăţile de proiectare (principala dificultate în opinia autorului) a unei metode de DSA rezidă în formalizarea contextului de apariţie al cuvântului de dezambiguizat. Algoritmii de DSA prezentaţi aici utilizează

[1]Nu am prezentat totuşi o metodă de construţie a unui text paralel, anume un aliniator de fraze. Pentru a-şi forma o părere asupra acestui subiect, cititorul poate consulta de exemplu [10, 70].

definiţii diferite ale contextului. WSDTool cuantifică contextul de apariţie al cuvântului de dezambiguizat ca un vector de traduceri al acestuia în limbile corpusului paralel iar SynWSD defineşte acelaşi context ca pe mulţimea de legături la care cuvântul de dezambiguizat participă. Ambele metode de dezambiguizare folosesc fraza ca entitate ale cărei cuvinte constituie materialul din care se construieşte reprezentarea contextului. Elementele de noutate introduse de cei doi algoritmi în rezolvarea problemei de DSA sunt următoarele:

- WSDTool: folosirea reţelelor semantice lexicale aliniate la nivel de concept pentru asigurarea unei adnotări uniforme a ocurenţelor cuvintelor conţinut ale unui text paralel cât şi pentru determinarea *exactă*[2] a mulţimii de înţesuri comune unei perechi de traducere;

- SynWSD: atribuirea *celei mai bune interpretări semantice* pentru o frază, compusă din etichete semantice pentru fiecare cuvânt conţinut în contrast cu practicile uzuale care optimizează parametrii de clasificare pentru fiecare cuvânt în parte fără a se reveni asupra unei clasificări deja făcute.

WSDTool şi **SynWSD** sunt algoritmi de DSA *neasistată* independenţi de limbă. Evaluările celor doi algoritmi au fost făcute pe corpusul paralel englez-român SemCor2.0, un corpus de referinţă (partea engleză) în evaluările metodelor de DSA. Evaluările din această lucrare s-au făcut pe 79595 de ocurenţe adnotate în engleză şi pe 48392 de ocurenţe adnotate în română[3] cu inventarele de sensuri ILI, SUMO şi IRST. În ce priveşte inventarul de sensuri ILI, algoritmii au demonstrat următoarele rezultate:

- WSDTool a obţinut o precizie de 70.21% cu un recall de 66.88% pentru engleză şi o precizie de 53.47% cu un recall de 49.80% pentru română (vezi tabelele B.1 şi B.4 din anexa B pentru detalierea performanţelor pentru fiecare fişier al corpusului SemCor2.0);

- SynWSD cu combinatorul `int` a atins la rândul său o precizie de 69.77% cu un recall de 26.63% (1.16 etichete pe ocurenţă) pentru engleză şi 59.84% precizie, 22.21% recall (1.37 etichete pe ocurenţă) pentru limba română (vezi tabelele C.7 şi C.8 din anexa C pentru detalii).

[2]Exactictate dependentă de corectitudinea şi completitudinea reţelelor semantice şi de corectitudinea alinierii conceptuale.

[3]După ştiinţa autorului, cea mai cuprinzătoare evaluare ca număr de ocurenţe de test.

Rezultatele afişate de SynWSD pentru inventarul de sens ILI nu-l recomandă deocamdată unei aplicaţii de PLN care are nevoie de astfel de etichete[4]. Totuşi, să luăm în calcul şi faptul că antrenarea acestui algoritm s-a făcut pe texte foarte mici (aproximativ 10^6 de unităţi lexicale) în comparaţie cu dimensiunea inventarului de sensuri ILI şi din acest motiv multe perechi de etichete nu au obţinut frecvenţe care să reflecte realitatea (acest lucru influenţând negativ performanţele algoritmului).

În ce priveşte ciclul de dezvoltare al celor doi algoritmi, în viitorul apropriat planificăm următoarele activităţi:

- îmbunătăţirea recall-ului algoritmului SynWSD şi studiul performanţelor sale (şi ale analizorului de legături LexPar) pe texte de antrenament de dimensiuni mari: 10^7 şi 10^8 unităţi lexicale.

- studierea unei noi metode de reducere a parametrilor modelului de atracţie semantică bazat pe ILI: antrenarea algoritmului SynWSD pe texte dezambiguizate în prealabil cu categorii SUMO sau domenii IRST pentru că SynWSD oferă rezultate bune cu aceste inventare de sens;

- studiul performanţelor unei metode mixte de adnotare: WSDTool şi SynWSD. SynWSD poate asista decizia lui WSDTool de a alege un înţeles din mulţimea de înţelesuri comune perechii de traducere iar la rândul său WSDTool poate reduce numărul de stări pe care decodorul Viterbi trebuie să le parcurgă, *garantând* că eticheta corectă se află în mulţimea redusă.

5.1 Contribuţii proprii

Contribuţiile autorului la rezolvarea problemei de DSA pentru limbile engleză şi română sunt rezumate în cele ce urmează:

- asamblarea corpusului paralel englez-român **SemCor2.0** şi transferul adnotărilor semantice din engleză în română. SemCor este corpusul de referinţă în testarea algoritmilor de DSA pentru limba engleză;

- dezvoltarea modulului de preprocesat texte **TTL** care realizează operaţiile de recunoaştere a entităţilor denumite, segmentare la nivel de frază şi cuvânt, adnotare cu etichete morfosintactice, lematizare şi recunoaştere a grupurilor sintactice nominale şi prepoziţionale nerecursive cât şi a complecşilor verbali şi adjectivali. Segmentarea la nivel de

[4]În cazul în care aplicaţia are nevoie de un recall mare şi de asemenea de un număr mediu de etichete semantice pe ocurenţă egal cu 1.

cuvânt, adnotarea cu etichete morfosintactice şi lematizarea sunt necesare oricărui algoritm de DSA. TTL este în prezent adaptat ca serviciu web la adresa `http://nlp.racai.ro/`[5];

- dezvoltarea aliniatorului lexical **YAWA** necesar algoritmului WSDTool pentru depistarea echivalenţilor de traducere din textul paralel. YAWA este o componentă a sistemului de aliniere lexicală **COWAL** ([117, 107, 118]) care a câştigat primul loc la competiţia de aliniere lexicală din cadrul "The 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05) Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond", Ann Arbor, SUA. De asemenea, YAWA a avut câteva aplicaţii directe în probleme cum ar fi transferul adnotărilor sintactice într-un corpus paralel ([3, 4]) sau evaluarea dificultăţilor de traducere ([39]);

- dezvoltarea algoritmului de DSA neasistată **WSDTool** care operează pe texte paralele utilizând trei inventare de sensuri: conceptele ontologiilor lexicale PWN2.0 şi ROWN2.0, categoriile SUMO şi domeniile IRST. WSDTool a fost folosit de asemenea şi la verificarea corectitudinii alinierii conceptuale dintre PWN2.0 şi ROWN2.0 ([115, 112, 120]);

- dezvoltarea analizorului de legături **LexPar**, o implementare a unui model de atracţie lexicală cu reguli de constrângere a existenţei legăturilor. Structura de legături furnizată de LexPar este necesară[6] algoritmului SynWSD pentru a putea rula.

- dezvoltarea algoritmului de DSA neasistată **SynWSD** care utilizează ca şi WSDTool cele trei inventare de sensuri.

Pe perioada stagiului său doctoral, autorul a publicat **26** de lucrări din care 23 de articole în reviste de specialitate şi în volume ale conferinţelor sau întâlnirilor de lucru naţionale şi internaţionale. Cele mai importante dintre acestea sunt:

- The Association for Computational Linguistics (ACL)[7];

- The North American Chapter of the Association for Computational Linguistics (NAACL)[8];

[5]La momentul scrierii acestor rânduri TTL a prelucrat aproximativ 2 miliarde de cuvinte în limba română. De asemenea, a fost folosit la adnotarea corpusului paralel englez-român TimeBank1.2 ([27]) şi la adnotarea corpusului de întrebări din [86].

[6]LexPar a mai fost folosit de asemenea şi la depistarea grupurilor sintactice în [41].

[7]`http://www.aclweb.org/`

[8]`http://www.cs.cornell.edu/home/llee/naacl/`

- The International Conference on Computational Linguistics (COLING)[9];
- The European Chapter of the Association for Computational Linguistics (EACL)[10];
- The Language Resources And Evaluation Conference (LREC)[11];
- The International Florida Artificial Intelligence Research Society Conference (FLAIRS)[12];
- Revista "Language Resources and Evaluation", ISSN 1574-020X, Springer Netherlands;
- Revista "Romanian Journal on Information Science and Technology", ISSN 1453-8245, Editura Academiei Române.

[9] http://www.issco.unige.ch/coling2004/
[10] http://eacl.coli.uni-saarland.de/
[11] http://www.lrec-conf.org/
[12] http://www.flairs.com/

Anexa A

Tabelele de mai jos reprezintă setul de etichete morfosintactice pentru engleză şi română folosite de modulul TTL. Coloana **MSD** conţine etichetele compatibile cu specificaţiile MULTEXT-East ([21]), coloana **CTAG** dă etichetele derivate din etichetele MSD iar ultima coloană, **CAT** conţine la rândul ei metacategoriile folosite de YAWA.

Din motive de editare, ultima linie a primului tabel pentru română este reprodusă mai jos:

⟨ `Mmfpr-yy`, `M`, `1` ⟩, ⟨ `Mmfso-n`, `M`, `1` ⟩, ⟨ `Mmfso-ny`, `M`, `1` ⟩

Engleză

MSD	CTAG	CAT	MSD	CTAG	CAT	MSD	CTAG	CAT
Af	ADJE	1	Afc	ADJE	1	Afp	ADJE	1
Afs	ADJE	1	Cc	CCOMP	31	Cc-i	CCOO	31
Cc-n	CCOO	31	Cs	CSUB	31	Dd	DM	2
Dd3	DM	2	Dd3-p	DMP	2	Dd3-s	DMS	2
Di3	PI	22	Di3-p	PI	22	Di3-s	PI	22
Ds	PS	10	Ds----p	PSP	10	Ds----s	PSS	10
Ds1---p	PSP	10	Ds1---s	PSS	10	Ds2	PS	10
Ds3---p	PSP	10	Ds3---sf	PSS	10	Ds3---sm	PSS	10
Ds3---sn	PSS	10	Dw	RELQ	4	Dw--------q	RELQ	4
Dw--------r	RELQ	4	Dz3	PZ	22	Dz3-s	PZ	22
Dz3-p	PZ	22	I	UH	16	M	CD	1
Mc	CD	1	Mo	CD	1	Nc	NN	1
Nc----y	NNY	1	Nc-p	NNS	1	Nc-p--y	NNSY	1
Nc-s	NN	1	Nc-s--y	NNY	1	Ncf	NN	1
Ncf---y	NNY	1	Ncfp	NNS	1	Ncfp--y	NNSY	1
Ncfs	NN	1	Ncfs--y	NNY	1	Ncm	NN	1
Ncm---y	NNY	1	Ncmp	NNS	1	Ncmp--y	NNSY	1
Ncms	NN	1	Ncms--y	NNY	1	Ncn	NN	1
Ncn---y	NNY	1	Ncnp	NNS	1	Ncnp--y	NNSY	1
Ncns	NN	1	Ncns--y	NNY	1	Np	NNP	8
Np-p	NNPS	8	Np-s	NNP	8	Npfs	NNP	8
Npms	NNP	8	Npnp	NNPS	8	Npns	NNP	8
Pd3-p	DMP	2	Pd3-s	DMS	2	Pd3	DM	2
Pi3	PI	22	Pi3-p	PI	22	Pi3-s	PI	22
Pp	PPER	13	Pp--pn	PPER	13	Pp--sn	PPER	13
Pp1	PPER1	13	Pp1-sa	PPER1	13	Pp1-pa	PPER1	13
Pp1-pn	PPER1	13	Pp1-sn	PPER1	13	Pp2	PPER2	13
Pp2-p	PPER2	13	Pp3	PPER3	13	Pp3-pa	PPER3	13
Pp3-pn	PPER3	13	Pp3-sn	PPER3	13	Pp3fs	PPER3	13
Pp3fsa	PPER3	13	Pp3fsn	PPER3	13	Pp3ms	PPER3	13
Pp3msa	PPER3	13	Pp3msn	PPER3	13	Pp3ns	PPER3	13
Ps	PS	10	Ps----p	PSP	10	Ps----s	PSS	10
Ps1---p	PSP	10	Ps1---s	PSS	10	Ps2	PS	10
Ps3	PS	10	Ps3---p	PSP	10	Ps3---sf	PSS	10
Ps3---sm	PSS	10	Pt3	EX	0	Pw	RELQ	4
Pw---------------q	RELQ	4	Pw---------------r	RELQ	4	Pw---a----------q	RELQ	4
Pw---a----------r	RELQ	4	Pw3-------------q	RELQ	4	Pw3-------------r	RELQ	4
Pw3n	RELQ	4	Pw3-p	RELQ	4	Dw3-p	RELQ	4
Pw3-s	RELQ	4	Dw3-s	RELQ	4	Px	PREF	12
Px1-p	PREF	12	Px1-s	PREF	12	Px2-p	PREF	12
Px2-s	PREF	12	Px3-p	PREF	12	Px3-s	PREF	12
Px3fs	PREFFS	12	Px3ms	PREFMS	12	Px3ns	PREFNS	12
Pz3	PZ	22	Pz3-s	PZ	22	Pz3-p	PZ	22
Qn	TO	15	Qz	NOT	7	R	ADVE	14
Rm	ADVE	14	R-p---q	ADVE	14	Rmc	ADVE	14
Rmp	ADVE	14	Rmp---q	ADVE	14	Rmp---r	ADVE	14
Rms	ADVE	14	Rsc	ADVE	14	Rsp	ADVE	14
Rss	ADVE	14	S	PREP	5	Sp	PREP	5
St	POST	21	Ti-s	TS	21	Va	AUX	3
Vacs	AUX	3	Vaip	AUX	3	Vaip-p	AUXP	3

MSD	CTAG	CAT	MSD	CTAG	CAT	MSD	CTAG	CAT
Vaip1p	AUXP	3	Vaip1s	AUX1	3	Vaip2s	AUX2	3
Vaip3s	AUX3	3	Vais	AUX	3	Vais-p	AUXP	3
Vais1s	AUX1	3	Vais2s	AUX2	3	Vais3s	AUX3	3
Van	AUXB	3	Vapp	AUXPP	3	Vaps	AUXPS	3
Vm	VERB	1	Vmcs	PAST	1	Vmip	VERB	1
Vmip-p	VERB	1	Vmip1s	VERB1	1	Vmip2s	VERB2	1
Vmip3s	VERB3	1	Vmis	PAST	1	Vmis-p	PAST	1
Vmis1s	PAST1	1	Vmis2s	PAST2	1	Vmis3s	PAST3	1
Vmn	VINF	1	Vmnp	VERB	1	Vmpp	PPRE	1
Vmps	PPAS	1	Vo	VMOD	1	Voip	VMOD	1
Voip3s	VMOD	1	Vois	VMOD	1	Von	VMOD	1
Vops	VMOD	1	Vopp	VMOD	1	Y	Y	8
Yn	Y	8	X	X	100	Eno	CD	1
En	CD	1	Eni	CD	1	Enr	CD	1
Eti	CD	1	Etp	CD	1	Etd	NN	1
Eqy	NN	1	Eqyi	NN	1	Eqt	NN	1
Eqd	NN	1	Eqa	NN	1	Eqm	NN	1
Eqv	NN	1	Ed	NNP	8	Edp	NNP	8
Edpm	NNP	8	Edpf	NNP	8	Edl	NNP	8
Edlc	NNP	8	Edly	NNP	8			

Română

MSD	CTAG	CAT	MSD	CTAG	CAT	MSD	CTAG	CAT
Afcfp-n	APN	1	Afcfpoy	APOY	1	Afcfpry	APRY	1
Afcfson	ASON	1	Afcfsoy	ASOY	1	Afcfsrn	ASN	1
Afcfsry	ASRY	1	Afcmp-n	APN	1	Afcmpoy	APOY	1
Afcmpry	APRY	1	Afcms-n	ASN	1	Afp	A	1
Af	A	1	Afp-p-n	APN	1	Afp-p-ny	APN	1
Afp-poy	APOY	1	Afpf--n	AN	1	Afpf--ny	AN	1
Afpfp-n	APN	1	Afpfp-ny	APN	1	Afpfpon	APON	1
Afpfpoy	APOY	1	Afpfpoyy	APOY	1	Afpfpry	APRY	1
Afpfpryy	APRY	1	Afpfson	ASON	1	Afpfsony	ASON	1
Afpfsoy	ASOY	1	Afpfsoyy	ASOY	1	Afpfsrn	ASN	1
Afpfsrny	ASN	1	Afpfsry	ASRY	1	Afpfsryy	ASRY	1
Afpfsvn	ASVN	1	Afpfsvy	ASVY	1	Afpm--n	AN	1
Afpmp-n	APN	1	Afpmp-ny	APN	1	Afpmpoy	APOY	1
Afpmpoyy	APOY	1	Afpmpry	APRY	1	Afpmpryy	APRY	1
Afpms-n	ASN	1	Afpms-ny	ASN	1	Afpmsoy	ASOY	1
Afpmsoyy	ASOY	1	Afpmsry	ASRY	1	Afpmsryy	ASRY	1
Afpmsvn	ASVN	1	Afpmsvy	ASVY	1	Afs	A	1
Afsfp-n	APN	1	Afsfpoy	APOY	1	Afsfpry	APRY	1
Afsfson	ASON	1	Afsfsoy	ASOY	1	Afsfsrn	ASN	1
Afsfsry	ASRY	1	Afsmp-n	APN	1	Afsmpoy	APOY	1
Afsmpry	APRY	1	Afsms-n	ASN	1	Afsmsoy	ASOY	1
Afsmsry	ASRY	1	Afsmsvy	ASVY	1	Cccsp	C	31
Ccssp	C	31	Ccsspy	C	31	Crssp	CR	31
Cscsp	C	31	Csssp	C	31	Cssspy	C	31
Dd3-po---e	DMPO	2	Dd3-po---o	DMPO	2	Dd3fpo	DMPO	2
Dd3fpr	DMPR	2	Dd3fpr---e	DMPR	2	Dd3fpr---o	DMPR	2
Dd3fpr--y	DMPR	2	Dd3fso	DMSO	2	Dd3fso---e	DMSO	2
Dd3fso---o	DMSO	2	Dd3fsr	DMSR	2	Dd3fsr---e	DMSR	2
Dd3fsr---o	DMSR	2	Dd3fsr--ye	DMSR	2	Dd3fsr--yo	DMSR	2
Dd3mpo	DMPO	2	Dd3mpr	DMPR	2	Dd3mpr---e	DMPR	2
Dd3mpr---o	DMPR	2	Dd3mpr--y	DMPR	2	Dd3mpr--yo	DMPR	2
Dd3mso	DMSO	2	Dd3mso---e	DMSO	2	Dd3mso---o	DMSO	2
Dd3msr	DMSR	2	Dd3msr---e	DMSR	2	Dd3msr---o	DMSR	2
Dd3msr--y	DMSR	2	Dd3msr--yo	DMSR	2	Dh1fp	PSP	10
Dh1fs	PSS	10	Dh1fso	PSS	10	Dh1fsr	PSS	10
Dh1mp	PSP	10	Dh1ms	PSS	10	Dh2fp	PSP	10
Dh2fs	PSS	10	Dh2fso	PSS	10	Dh2fsr	PSS	10
Dh2mp	PSP	10	Dh2ms	PSS	10	Dh3fp	PSP	10
Dh3fs	PSS	10	Dh3fso	PSS	10	Dh3fsr	PSS	10
Dh3mp	PSP	10	Dh3ms	PSS	10	Di3	PI	22
Di3------e	PI	22	Di3-----y	PI	22	Di3--r	PI	22
Di3--r---e	PI	22	Di3-po	PI	22	Di3-po---e	PI	22
Di3-s----e	PI	22	Di3-sr	PI	22	Di3-sr---e	PI	22
Di3-sr--y	PI	22	Di3fp	PI	22	Di3fpr	PI	22
Di3fpr---e	PI	22	Di3fso	PI	22	Di3fso---e	PI	22
Di3fsr	PI	22	Di3fsr---e	PI	22	Di3mp	PI	22
Di3mpo	PI	22	Di3fpo	PI	22	Di3mpr	PI	22
Di3mpr---e	PI	22	Di3ms	PI	22	Di3mso---e	PI	22
Di3msr	PI	22	Di3msr---e	PI	22	Di3msr--y	PI	22
Ds1fp-p	PSP	10	Ds1fp-s	PSP	10	Ds1fsop	PSS	10
Ds1fsos	PSS	10	Ds1fsos-y	PSS	10	Ds1fsrp	PSS	10
Ds1fsrs	PSS	10	Ds1fsrs-y	PSS	10	Ds1mp-p	PSP	10
Ds1mp-s	PSP	10	Ds1ms-p	PSS	10	Ds1ms-s	PSS	10
Ds1msrs-y	PSS	10	Ds2fp-p	PSP	10	Ds2fp-s	PSP	10
Ds2fsop	PSS	10	Ds2fsos	PSS	10	Ds2fsos-y	PSS	10
Ds2fsrp	PSS	10	Ds2fsrs	PSS	10	Ds2fsrs-y	PSS	10
Ds2---s	PS	10	Ps2---s	PS	10	Ds2mp-p	PSP	10
Ds2mp-s	PSP	10	Ds2ms-p	PSS	10	Ds2ms-s	PSS	10
Ds2msrs-y	PSS	10	Ds3---s	PS	10	Ds3---p	PS	10
Ds3fp-s	PSP	10	Ds3fsos	PSS	10	Ds3fsos-y	PSS	10
Ds3fsrs	PSS	10	Ds3fsrs-y	PSS	10	Ds3mp-s	PSP	10
Ds3ms-s	PSS	10	Ds3msrs-y	PSS	10	Dw3--r---e	RELR	4
Dw3-po	RELO	4	Dw3-po---e	RELO	4	Dw3fpr	RELR	4
Dw3fso---e	RELO	4	Dw3fsr	RELR	4	Dw3mpr	RELR	4
Dw3mso---e	RELO	4	Dw3msr	RELR	4	Dz3-po---e	PI	22
Dz3fso---e	PI	22	Dz3fsr---e	PI	22	Dz3mpr---e	PI	22
Dz3mso---e	PI	22	Dz3msr---e	PI	22	I	I	16
Mc-p-d	M	1	Mc-p-l	M	1	Mc-p-r	M	1
Mc-s-d	M	1	Mc-s-r	M	1	Mcfp-l	M	1
Mcfp-ln	M	1	Mcfp-rn	M	1	Mcfpoly	M	1
Mcfprln	M	1	Mcfprly	M	1	Mcfsoln	M	1
Mcfsoly	M	1	Mcfsrln	M	1	Mcfsrly	M	1
Mcmp-l	M	1	Mcfs-l	M	1	Mcms-ln	M	1
Mcmsoly	M	1	Mcmsrl	M	1	Mcmsrly	M	1
Mffpoly	M	1	Mffprln	M	1	Mffprly	M	1
Mffsoln	M	1	Mffsoly	M	1	Mffsrln	M	1
Mffsrly	M	1	Ml-po	M	1	Ml-pr	M	1
Mlfpo	M	1	Mlfpr	M	1	Mlmpo	M	1
Mlmpr	M	1	Mmfp--n	M	1	Mmfp--ny	M	1
Mmfpo-y	M	1	Mmfpo-yy	M	1	Mmfpr-y	M	1

MSD	CTAG	CAT	MSD	CTAG	CAT	MSD	CTAG	CAT
Mmfso-y	M	1	Mmfso-yy	M	1	Mmfsr-n	M	1
Mmfsr-ny	M	1	Mmfsr-y	M	1	Mmfsr-yy	M	1
Mmmpo-y	M	1	Mmmpo-yy	M	1	Mmmpr-n	M	1
Mmmpr-ny	M	1	Mmmpr-y	M	1	Mmmpr-yy	M	1
Mmmso-y	M	1	Mmmso-yy	M	1	Mmmsr-n	M	1
Mmmsr-ny	M	1	Mmmsr-y	M	1	Mmmsr-yy	M	1
Mo---l	M	1	Mo---ln	M	1	Mo---lny	M	1
Mo-s-r	M	1	Mofp-ln	M	1	Mofpoly	M	1
Mofpolyy	M	1	Mofprly	M	1	Mofprlyy	M	1
Mofs-l	M	1	Mofsoln	M	1	Mofsoly	M	1
Mofsolyy	M	1	Mofsrln	M	1	Mofsrly	M	1
Mofsrlyy	M	1	Momp-ln	M	1	Mompoly	M	1
Mompolyy	M	1	Momprly	M	1	Momprlyy	M	1
Moms-l	M	1	Moms-ln	M	1	Momsoly	M	1
Momsolyy	M	1	Momsrly	M	1	Momsrlyy	M	1
Nc	NN	1	Ncm	NN	1	Nc---n	NN	1
Nc-s-ny	NSN	1	Ncf--n	NN	1	Ncf--ny	NN	1
Ncfp-n	NPN	1	Ncfp-ny	NPN	1	Ncfpoy	NPOY	1
Ncfpoyy	NPOY	1	Ncfpry	NPRY	1	Ncfpryy	NPRY	1
Ncfpvy	NPVY	1	Ncfs-n	NSN	1	Ncfson	NSON	1
Ncfsony	NSON	1	Ncfsoy	NSOY	1	Ncfsoyy	NSOY	1
Ncfsrn	NSRN	1	Ncfsrny	NSRN	1	Ncfsry	NSRY	1
Ncfsryy	NSRY	1	Ncfsvy	NSVY	1	Ncm--n	NN	1
Ncmp-n	NPN	1	Ncmp-ny	NPN	1	Ncmpoy	NPOY	1
Ncmpoyy	NPOY	1	Ncmpry	NPRY	1	Ncmpryy	NPRY	1
Ncmpvy	NPVY	1	Ncms-n	NSN	1	Ncms-ny	NSN	1
Ncms-y	NSY	1	Ncmsoy	NSOY	1	Ncmsoyy	NSOY	1
Ncmsrny	NSRN	1	Ncmsrn	NSRN	1	Ncmsry	NSRY	1
Ncmsryy	NSRY	1	Ncmsvn	NSVN	1	Ncmsvny	NSVN	1
Ncmsvy	NSVY	1	Np	NP	8	Npfp-n	NP	8
Npfpoy	NP	8	Npfpry	NP	8	Npfs-n	NP	8
Npfson	NP	8	Npfsoy	NP	8	Npfsrn	NP	8
Npfsry	NP	8	Npfsvy	NP	8	Npmp-n	NP	8
Npmpoy	NP	8	Npmpry	NP	8	Npms-n	NP	8
Npms-y	NP	8	Npmsoy	NP	8	Npmsry	NP	8
Npmsvn	NP	8	Npmsvy	NP	8	Pd3-po	DMPO	2
Pd3fpo	DMPO	2	Pd3fpr	DMPR	2	Pd3fpr--y	DMPR	2
Pd3fso	DMSO	2	Pd3fsr	DMSR	2	Pd3fsr--y	DMSR	2
Pd3mpo	DMPO	2	Pd3mpr	DMPR	2	Pd3mpr--y	DMPR	2
Pd3mso	DMSO	2	Pd3msr	DMSR	2	Pd3msr--y	DMSR	2
Pi3	PI	22	Pi3-pr	PI	22	Pi3--r	PI	22
Pi3-po	PI	22	Pi3-so	PI	22	Pi3-sr	PI	22
Pi3fpr	PI	22	Pi3fso	PI	22	Pi3fsr	PI	22
Pi3mpr	PI	22	Pi3mpo	PI	22	Pi3fpo	PI	22
Pi3mso	PI	22	Pi3msr	PI	22	Pi3msr--y	PI	22
Pp1-pa--------w	PPPA	13	Pp1-pa--y-----w	PPPA	13	Pp1-pd--------s	PPPD	13
Pp1-pd--------w	PPPD	13	Pp1-pd--y-----w	PPPD	13	Pp1-pr--------s	PPPR	13
Pp1-sa--------s	PPSA	13	Pp1-sa--------w	PPSA	13	Pp1-sa--y-----w	PPSA	13
Pp1-sd--------s	PPSD	13	Pp1-sd--------w	PPSD	13	Pp1-sd--y-----w	PPSD	13
Pp1-sn--------s	PPSN	13	Pp1-sr--------s	PPSR	13	Pp2----------s	PP	13
Pp2-pa--------w	PPPA	13	Pp2-pa--y-----w	PPPA	13	Pp2-pd--------s	PPPD	13
Pp2-pd--------w	PPPD	13	Pp2-pd--y-----w	PPPD	13	Pp2-po--------s	PPPO	13
Pp2-pr--------s	PPPR	13	Pp2-s---------s	PP	13	Pp2-sa--------s	PPSA	13
Pp2-sa--------w	PPSA	13	Pp2-sa--y-----w	PPSA	13	Pp2-sd--------s	PPSD	13
Pp2-sd--------w	PPSD	13	Pp2-sd--y-----w	PPSD	13	Pp2-sn--------s	PPSN	13
Pp2-so--------s	PPSO	13	Pp2-sr--------s	PPSR	13	Pp3-p---------s	PPP	13
Pp3-pd--------w	PPPD	13	Pp3-pd--y-----w	PPPD	13	Pp3-po--------s	PPPO	13
Pp3-pr--------s	PPPR	13	Pp3-sd--------w	PPSD	13	Pp3-sd--y-----w	PPSD	13
Pp3-so--------s	PPSO	13	Pp3-sr--------s	PPSR	13	Pp3fpa--------w	PPPA	13
Pp3fpa--y-----w	PPPA	13	Pp3fpo--------s	PPPO	13	Pp3fpr--------s	PPPR	13
Pp3fpr--y-----s	PPPR	13	Pp3fs---------s	PPS	13	Pp3fsa--------w	PPSA	13
Pp3fsa--y-----w	PPSA	13	Pp3fso--------s	PPSO	13	Pp3fsr--------s	PPSR	13
Pp3fsr--y-----s	PPSR	13	Pp3mpa--------w	PPPA	13	Pp3mpa--y-----w	PPPA	13
Pp3mpo--------s	PPPO	13	Pp3mpr--------s	PPPR	13	Pp3mpr--y-----s	PPPR	13
Pp3ms---------s	PPS	13	Pp3msa--------w	PPSA	13	Pp3msa--y-----w	PPSA	13
Pp3mso--------s	PPSO	13	Pp3msr--------s	PPSR	13	Pp3msr--y-----s	PPSR	13
Ps1fp-p	PSP	10	Ps1fp-s	PSP	10	Ps1fsrp	PSS	10
Ps1fsrs	PSS	10	Ps1mp-p	PSP	10	Ps1mp-s	PSP	10
Ps1mprp	PSP	10	Ps1mprs	PSP	10	Ps1ms-p	PSS	10
Ps1ms-s	PSS	10	Ps2fp-p	PSP	10	Ps2fp-s	PSP	10
Ps2fsrp	PSS	10	Ps2fsrs	PSS	10	Ps2mp-p	PSP	10
Ps2mp-s	PSP	10	Ps2mprp	PSP	10	Ps2mprs	PSP	10
Ps2ms-p	PSS	10	Ps2ms-s	PSS	10	Ps2msrs-y	PSS	10
Ps3fp-s	PSP	10	Ps3fsrs	PSS	10	Ps3mp-s	PSP	10
Ps3mprs	PSP	10	Ps3ms-s	PSS	10	Ps3---s	PS	10
Ps3---p	PS	10	Pw3--r	RELR	4	Pw3-po	RELO	4
Pw3-so	RELO	4	Pw3fpr	RELR	4	Pw3fso	RELO	4
Pw3fsr	RELR	4	Pw3mpr	RELR	4	Pw3mso	RELO	4
Pw3msr	RELR	4	Px3--a---------s	PXA	12	Px3--a--------w	PXA	12
Px3--a--y-----w	PXA	12	Px3--d---------s	PXD	12	Px3--d--------w	PXD	12
Px3--d--y-----w	PXD	12	Pz3-po	PI	22	Pz3-so	PI	22

MSD	CTAG	CAT	MSD	CTAG	CAT	MSD	CTAG	CAT
Pz3-sr	PI	22	Pz3fpr	PI	22	Pz3fso	PI	22
Pz3fsr	PI	22	Pz3mpr	PI	22	Pz3mso	PI	22
Pz3msr	PI	22	Qf	QF	3	Qn	QN	15
Qn-y	QN	15	Qs	QS	15	Qz	QZ	7
Qz-y	QZ	7	Rc	RC	14	Rgc	R	14
Rgp	R	14	Rgpy	R	14	Rgs	R	14
Rp	R	14	Rp-y	R	14	Rw	R	14
Rw-y	R	14	Rz	R	14	Spca	S	5
Spcg	S	5	Spsa	S	5	Spsay	S	5
Spsd	S	5	Spsg	S	5	Spsgy	S	5
Sp	S	5	Td-po	TPO	21	Tdfpr	TPR	21
Tdfso	TSO	21	Tdfsr	TSR	21	Tdmpr	TPR	21
Tdmso	TSO	21	Tdmsr	TSR	21	Tf-so	TSO	21
Tffpoy	TPO	21	Tffpry	TPR	21	Tffs-y	TS	21
Tf-s-y	TS	21	Tffsoy	TSO	21	Tfmpoy	TPO	21
Tfmpry	TPR	21	Tfms-y	TS	21	Tfmsoy	TSO	21
Tfmsry	TSR	21	Ti-po	TPO	21	Tifp-y	TP	21
Tifso	TSO	21	Tifsoy	TSO	21	Tifsr	TSR	21
Tifsry	TSR	21	Timp-y	TP	21	Timso	TSO	21
Timsr	TSR	21	Timsry	TSR	21	Ts-po	TPO	21
Tsfp	TP	21	Tsfs	TS	21	Tsmp	TP	21
Tsms	TS	21	Va	VA	3	Va--1	VA1	3
Va--1-----y	VA1	3	Va--1p	VA1P	3	Va--1s	VA1S	3
Va--1s----y	VA1S	3	Va--2p	VA2P	3	Va--2p----y	VA2P	3
Va--2s	VA2S	3	Va--2s----y	VA2S	3	Va--3	VA3	3
Va--3-----y	VA3	3	Va--3p	VA3P	3	Va--3p----y	VA3P	3
Va--3s	VA3S	3	Va--3s----y	VA3S	3	Vaip3s----y	VA3S	3
Vaip1s----y	VA1S	3	Vaip3p----y	VA3P	3	Vaip1s	VA1S	3
Vais1s	VA1S	3	Vail2s	VA2S	3	Vail3s	VA3S	3
Vaip3s	VA3S	3	Vaii3s	VA3S	3	Vais3s	VA3S	3
Vais2s	VA2S	3	Vaip2s	VA2S	3	Vaii2s	VA2S	3
Vaip1p	VA1P	3	Vais1p	VA1P	3	Vail1p	VA1P	3
Vail2p	VA2P	3	Vail3p	VA3P	3	Vaip3p	VA3P	3
Vaii3p	VA3P	3	Vais3p	VA3P	3	Vais2p	VA2P	3
Vaip2p	VA2P	3	Vaii2p	VA2P	3	Vaii1	VA1	3
Vail1s	VA1S	3	Vasp3	VA3	3	Vam-2s	VA2S	3
Vam-2p	VA2P	3	Vasp1p	VA1P	3	Vasp2p	VA2P	3
Vasp1s	VA1S	3	Vasp2s	VA2S	3	Vag	VA	3
Vag-------y	VA	3	Vap--sm---y	VA	3	Vap--sm	VA	3
Vanp	VA	3	Vmg	VG	1	Vmg-------y	VG	1
Vmii1	V1	1	Vmii1-----y	V1	1	Vmii1p	V1	1
Vmii1s	V1	1	Vmii2p	V2	1	Vmii2p----y	V2	1
Vmii2s	V2	1	Vmii2s----y	V2	1	Vmii3p	V3	1
Vmii3p----y	V3	1	Vmii3s	V3	1	Vmii3s----y	V3	1
Vmil1p	V1	1	Vmil1p----y	V1	1	Vmil1s	V1	1
Vmil1s----y	V1	1	Vmil2p	V2	1	Vmil2p----y	V2	1
Vmil2s	V2	1	Vmil2s----y	V2	1	Vmil3p	V3	1
Vmil3p----y	V3	1	Vmil3s	V3	1	Vmil3s----y	V3	1
Vmip1p	V1	1	Vmip1p----y	V1	1	Vmip1s	V1	1
Vmip1s----y	V1	1	Vmip2p	V2	1	Vmip2p----y	V2	1
Vmip2s	V2	1	Vmip2s----y	V2	1	Vmip3	V3	1
Vmip3-----y	V3	1	Vmip3p	V3	1	Vmip3p----y	V3	1
Vmip3s	V3	1	Vmip3s----y	V3	1	Vmis1p	V1	1
Vmis1p----y	V1	1	Vmis1s	V1	1	Vmis1s----y	V1	1
Vmis2p	V2	1	Vmis2p----y	V2	1	Vmis2s	V2	1
Vmis2s----y	V2	1	Vmis3p	V3	1	Vmis3p----y	V3	1
Vmis3s	V3	1	Vmis3s----y	V3	1	Vmm-2p	V2	1
Vmm-2p----y	V2	1	Vmm-2s	V2	1	Vmm-2s----y	V2	1
Vmnp	VN	1	Vmnp------y	VN	1	Vmp--pf	VPPF	1
Vmp--pf---y	VPPF	1	Vmp--pm	VPPM	1	Vmp--pm---y	VPPM	1
Vmp--sf	VPSF	1	Vmp--sf---y	VPSF	1	Vmp--sm	VPSM	1
Vmp--sm---y	VPSM	1	Vmsp1p	V1	1	Vmsp1s	V1	1
Vmsp2p	V2	1	Vmsp2s	V2	1	Vmsp3	V3	1
Vmsp3-----y	V3	1	Vmsp3s	V3	1	Vmsp3s----y	V3	1
X	X	100	Y	Y	8	Ya	Y	8
Yn	Y	8	Ynfpvy	Y	8	Ynfsoy	Y	8
Ynfsry	Y	8	Ynmpoy	Y	8	Ynmpry	Y	8
Ynmpvy	Y	8	Ynmsoy	Y	8	Ynmsry	Y	8
Ynmsvy	Y	8	Yp	Y	8	Yp-p	Y	8
Yp-so	Y	8	Yp-sr	Y	8	Ypfpr	Y	8
Ypfs	Y	8	Ypfso	Y	8	Ypfsr	Y	8
Ypmpr	Y	8	Ypms	Y	8	Ypmso	Y	8
Ypmsr	Y	8	Yr	Y	8	Yv	Y	8
Eni	M	1	Enr	M	1	Etp	M	1
Etd	NN	1	Eqy	NN	1	Eqt	NN	1
Eqd	NN	1	Eqa	NN	1	Eqm	NN	1
Eqv	NN	1	Ed	NP	8	Edp	NP	8
Edpm	NP	8	Edpf	NP	8	Edl	NP	8
Edlc	NP	8	Edly	NP	8			

Anexa B

Tabelele de mai jos reprezintă performaţa algoritmului WSDTool pe corpusul paralel englez-român SemCor2.0. Coloana **Fişier** precizează fişierul corpusului pe care s-a rulat algoritmul, după care urmează valorile procentuale ale preciziei (**P(%)**), recall-ului (**R(%)**) şi ale f-measure (**F(%)**) iar coloana **S/C** indică numărul mediu de etichete semantice per cuvânt dezambiguizat (= numărul total de etichete semantice date de algoritm împărţit la numărul de ocurenţe ale tuturor cuvintelor dezambiguizate de algoritm). Tabelul este împărţit în două coloane, a doua în continuarea primeia.

Fișier	P(%)	R(%)	F(%)	S/C	Fișier	P(%)	R(%)	F(%)	S/C
br-a01	69.744	67.768	68.741	1	br-j23	68.351	65.44	66.863	1
br-a02	73.36	71.354	72.343	1	br-j37	71.648	68.776	70.182	1
br-a11	66.666	64.938	65.79	1	br-j52	63.766	58.452	60.993	1
br-a12	66.834	65.116	65.963	1	br-j53	70.167	67.309	68.708	1
br-a13	65.731	62.412	64.028	1	br-j54	68.488	66.012	67.227	1
br-a14	68.732	66.356	67.523	1	br-j55	72.296	70.247	71.256	1
br-a15	66.481	63.479	64.945	1	br-j56	71.383	67.359	69.312	1
br-b13	69.444	65.445	67.385	1	br-j57	68.625	63.964	66.212	1
br-b20	68.354	64.056	66.135	1	br-j58	67.107	64.548	65.802	1
br-c01	64.105	62.093	63.082	1	br-j59	70.104	68.828	69.46	1
br-c02	69.265	66.953	68.089	1	br-j60	70.022	65.732	67.809	1
br-c04	69.019	64.077	66.456	1	br-j70	73.874	70.56	72.178	1
br-d01	66.293	61.832	63.984	1	br-k01	72.235	68.901	70.528	1
br-d02	67.639	62.612	65.028	1	br-k02	69.52	67.144	68.311	1
br-d03	73.333	69.498	71.364	1	br-k03	66.978	62.081	64.436	1
br-d04	58.638	55.225	56.88	1	br-k04	71.41	66.161	68.685	1
br-e01	66.47	63.554	64.979	1	br-k05	67.549	61.893	64.597	1
br-e02	71.049	67.118	69.027	1	br-k06	69.75	66.067	67.858	1
br-e04	65.149	61.789	63.424	1	br-k07	69.292	66.114	67.665	1
br-e21	69.556	65.436	67.433	1	br-k08	66.84	63.456	65.104	1
br-e24	69.565	66.464	67.979	1	br-k09	71.784	68.204	69.948	1
br-e29	72.328	69.588	70.931	1	br-k10	68.795	66.087	67.413	1
br-f03	69.589	66.844	68.188	1	br-k11	71.462	69.061	70.24	1
br-f10	73.764	71.169	72.443	1	br-k12	70.349	68.271	69.294	1
br-f19	67.888	65.347	66.593	1	br-k13	74.584	69.523	71.964	1
br-f43	66.219	62.978	64.557	1	br-k14	71.062	67.911	69.45	1
br-g01	69.368	64.238	66.704	1	br-k15	65.832	64.146	64.978	1
br-g11	70.097	67.185	68.61	1	br-k16	72.871	69.934	71.372	1
br-g15	67.968	64.994	66.447	1	br-k17	68.219	64.341	66.223	1
br-h01	73.201	70.66	71.908	1	br-k18	67.805	63.08	65.357	1
br-j01	77.696	73.762	75.677	1	br-k19	69.318	65.435	67.32	1
br-j02	71.92	70.594	71.25	1	**Media**	**70.217**	**66.882**	68.501	**1.000**
br-j03	72.604	66.93	69.651	1	**F(%)**			**68.509**	
br-j04	72.172	66.158	69.034	1	**MIN**	58.638	55.225	56.880	1.000
br-j05	76.115	73.983	75.033	1	**MAX**	80.174	76.290	78.050	1.000
br-j06	69.389	67.982	68.678	1					
br-j07	71.229	68.028	69.591	1					
br-j08	74.652	70.221	72.368	1					
br-j09	76.489	73.465	74.946	1					
br-j10	74.778	70.711	72.687	1					
br-j11	73.239	69.565	71.354	1					
br-j12	79.647	76.29	77.932	1					
br-j13	71.851	70.918	71.381	1					
br-j14	74.829	72.21	73.496	1					
br-j15	80.174	76.036	78.05	1					
br-j16	75.601	73.908	74.744	1					
br-j17	74.066	73.018	73.538	1					
br-j18	67.289	57.067	61.757	1					
br-j19	71.019	67.414	69.169	1					
br-j20	67.787	62.532	65.053	1					
br-j22	67.948	67.948	67.948	1					

Tabela B.1: Rezultatele algoritmului WSDTool pe corpusul SemCor2.0 în limba engleză. Inventarul de sensuri este dat de ILI iar evaluarea este strictă.

Fișier	P(%)	R(%)	F(%)	S/C	Fișier	P(%)	R(%)	F(%)	S/C
br-a01	74.483	72.373	73.412	1	br-j23	74.216	71.055	72.601	1
br-a02	78.848	76.692	77.755	1	br-j37	78.351	75.21	76.748	1
br-a11	71.568	69.713	70.628	1	br-j52	71.818	65.833	68.695	1
br-a12	71.105	69.277	70.179	1	br-j53	75.083	72.025	73.522	1
br-a13	71.048	67.461	69.208	1	br-j54	76.205	73.45	74.802	1
br-a14	73.966	71.409	72.665	1	br-j55	78.007	75.796	76.885	1
br-a15	70.514	67.33	68.885	1	br-j56	78.459	74.035	76.182	1
br-b13	75.222	70.89	72.991	1	br-j57	76.263	71.084	73.582	1
br-b20	74.556	69.869	72.136	1.003	br-j58	75.957	73.062	74.481	1
br-c01	72.028	69.767	70.879	1	br-j59	76.477	75.085	75.774	1
br-c02	76.503	73.95	75.204	1	br-j60	76.438	71.754	74.021	1
br-c04	77.254	71.723	74.385	1.001	br-j70	78.277	74.766	76.481	1
br-d01	75.746	70.649	73.108	1	br-k01	78.456	74.835	76.602	1
br-d02	73.722	68.243	70.876	1	br-k02	77.511	74.862	76.163	1
br-d03	79.425	75.272	77.292	1	br-k03	74.732	69.268	71.896	1
br-d04	68.6	64.608	66.544	1	br-k04	76.196	70.595	73.288	1
br-e01	73.176	69.966	71.535	1	br-k05	74.701	68.446	71.436	1
br-e02	79.447	75.052	77.186	1	br-k06	77.817	73.707	75.706	1
br-e04	72.807	69.052	70.879	1	br-k07	75.834	72.356	74.054	1
br-e21	74.716	70.291	72.435	1	br-k08	75.422	71.604	73.463	1
br-e24	76.776	73.353	75.025	1	br-k09	78.083	74.189	76.086	1
br-e29	78.74	75.757	77.219	1	br-k10	75.542	72.569	74.025	1
br-f03	76.359	73.347	74.822	1	br-k11	78.657	76.013	77.312	1
br-f10	78.823	76.049	77.411	1	br-k12	77.001	74.726	75.846	1
br-f19	76.555	73.689	75.094	1	br-k13	81.609	76.071	78.742	1
br-f43	75.615	71.914	73.718	1	br-k14	76.678	73.278	74.939	1
br-g01	76.162	70.529	73.237	1	br-k15	73.967	72.073	73.007	1
br-g11	78.223	74.974	76.564	1	br-k16	79.341	76.143	77.709	1.002
br-g15	75.892	72.572	74.194	1	br-k17	75.479	71.188	73.27	1.001
br-h01	78.654	75.923	77.264	1	br-k18	75.558	70.293	72.83	1
br-j01	83.636	79.401	81.463	1	br-k19	77.651	73.301	75.413	1
br-j02	77.244	75.819	76.524	1	**Media**	**76.788**	**73.144**	74.912	**1.000**
br-j03	76.289	70.328	73.187	1	**F(%)**			**74.921**	
br-j04	76.385	70.02	73.064	1	**MIN**	68.600	59.973	64.902	1.000
br-j05	81.784	79.494	80.622	1	**MAX**	85.131	80.769	82.875	1.003
br-j06	76.143	74.599	75.363	1.001					
br-j07	75.508	72.114	73.771	1.002					
br-j08	80.748	75.955	78.278	1					
br-j09	81.798	78.563	80.147	1					
br-j10	80.752	76.359	78.494	1					
br-j11	79.929	75.919	77.872	1					
br-j12	82.728	79.241	80.946	1					
br-j13	81.599	80.54	81.066	1					
br-j14	79.705	76.914	78.284	1					
br-j15	85.131	80.737	82.875	1					
br-j16	81.557	79.731	80.633	1					
br-j17	78.086	76.981	77.529	1					
br-j18	70.716	59.973	64.902	1					
br-j19	77.736	73.789	75.711	1					
br-j20	76.05	70.155	72.983	1					
br-j22	80.769	80.769	80.769	1					

Tabela B.2: Rezultatele algoritmului WSDTool pe corpusul SemCor2.0 în limba engleză. Inventarul de sensuri este dat de categoriile SUMO iar evaluarea este strictă.

Fișier	P(%)	R(%)	F(%)	S/C	Fișier	P(%)	R(%)	F(%)	S/C
br-a01	88.092	85.596	86.826	1.147	br-j23	85.945	82.284	84.074	1.045
br-a02	89.022	86.588	87.788	1.133	br-j37	88.461	84.915	86.651	1.134
br-a11	79.971	77.899	78.921	1.091	br-j52	85.584	78.452	81.862	1.044
br-a12	84.296	82.129	83.198	1.111	br-j53	90.614	86.923	88.73	1.045
br-a13	79.468	75.455	77.409	1.097	br-j54	87.352	84.194	85.743	1.081
br-a14	85.674	82.712	84.166	1.119	br-j55	89.671	87.131	88.382	1.14
br-a15	85.257	81.407	83.287	1.08	br-j56	87.893	82.937	85.343	1.144
br-b13	84.666	79.79	82.155	1.078	br-j57	85.663	79.846	82.652	1.092
br-b20	87.594	82.087	84.751	1.1	br-j58	85.601	82.337	83.937	1.088
br-c01	84.393	81.744	83.047	1.129	br-j59	91.193	89.533	90.355	1.032
br-c02	86.414	83.53	84.947	1.096	br-j60	88.495	83.073	85.698	1.183
br-c04	86.013	79.854	82.819	1.15	br-j70	88.258	84.299	86.233	1.096
br-d01	87.562	81.67	84.513	1.052	br-k01	89.17	85.054	87.063	1.079
br-d02	88.564	81.981	85.145	1.043	br-k02	86.757	83.792	85.248	1.067
br-d03	90.574	85.838	88.142	1.103	br-k03	86.497	80.173	83.215	1.098
br-d04	84.741	79.809	82.201	1.035	br-k04	88.539	82.03	85.16	1.112
br-e01	86.588	82.789	84.645	1.072	br-k05	87.682	80.339	83.85	1.082
br-e02	88.287	83.402	85.775	1.161	br-k06	88.612	83.932	86.208	1.102
br-e04	86.681	82.21	84.386	1.058	br-k07	90.387	86.242	88.265	1.042
br-e21	86.996	81.844	84.341	1.1	br-k08	88.036	83.58	85.75	1.084
br-e24	87.592	83.687	85.594	1.075	br-k09	91.207	86.658	88.874	1.115
br-e29	89.426	86.038	87.699	1.077	br-k10	88.915	85.416	87.13	1.086
br-f03	89.123	85.607	87.329	1.061	br-k11	89.088	86.095	87.565	1.119
br-f10	88.588	85.471	87.001	1.081	br-k12	89.177	86.542	87.839	1.098
br-f19	87	83.743	85.34	1.084	br-k13	90.166	84.047	86.999	1.111
br-f43	84.563	80.425	82.442	1.228	br-k14	91.208	87.164	89.14	1.092
br-g01	89.868	83.222	86.417	1.088	br-k15	87.984	85.731	86.842	1.098
br-g11	88.299	84.631	86.426	1.059	br-k16	89.557	85.947	87.714	1.135
br-g15	86.941	83.137	84.996	1.046	br-k17	87.808	82.816	85.238	1.097
br-h01	87.122	84.098	85.583	1.063	br-k18	88.83	82.64	85.623	1.086
br-j01	87.878	83.429	85.595	1.047	br-k19	90.656	85.578	88.043	1.075
br-j02	83.194	81.659	82.419	1.147	**Media**	**87.636**	**83.463**	85.488	**1.092**
br-j03	87.469	80.634	83.912	1.052	**F(%)**			**85.498**	
br-j04	86.585	79.369	82.82	1.087	**MIN**	79.468	71.598	77.409	1.032
br-j05	91.263	88.708	89.967	1.075	**MAX**	92.307	92.307	92.307	1.228
br-j06	82.679	81.003	81.832	1.09					
br-j07	84.171	80.388	82.236	1.077					
br-j08	89.625	84.305	86.883	1.069					
br-j09	91.115	87.513	89.277	1.075					
br-j10	89.048	84.205	86.558	1.139					
br-j11	88.497	84.057	86.219	1.153					
br-j12	91.749	87.881	89.773	1.068					
br-j13	88.061	86.918	87.485	1.066					
br-j14	86.054	83.041	84.52	1.111					
br-j15	89.99	85.345	87.605	1.089					
br-j16	84.994	83.09	84.031	1.116					
br-j17	90.047	88.773	89.405	1.103					
br-j18	84.423	71.598	77.483	1.062					
br-j19	88.184	83.707	85.887	1.079					
br-j20	84.453	77.906	81.047	1.084					
br-j22	92.307	92.307	92.307	1.064					

Tabela B.3: Rezultatele algoritmului WSDTool pe corpusul SemCor2.0 în limba engleză. Inventarul de sensuri este dat de domeniile IRST iar evaluarea este strictă.

Fişier	P(%)	R(%)	F(%)	S/C	Fişier	P(%)	R(%)	F(%)	S/C
br-a01	48.893	47.184	48.023	1	br-j23	53.375	49.705	51.474	1
br-a02	55.205	52.9	54.027	1	br-j37	46.617	44.58	45.575	1
br-a11	52.647	50.297	51.445	1	br-j52	42.129	37.067	39.436	1
br-a12	52.784	50.934	51.842	1	br-j53	56.374	53.295	54.791	1
br-a13	52.188	47.546	49.758	1	br-j54	54.639	51.356	52.946	1
br-a14	52	49.367	50.649	1	br-j55	57.228	55.232	56.212	1
br-a15	52.089	48.071	49.999	1	br-j56	56.315	51.69	53.903	1
br-b13	59.71	54.528	57.001	1	br-j57	50.947	44.605	47.565	1
br-b20	54.385	48.329	51.178	1	br-j58	45.274	43.096	44.158	1
br-c01	57.907	54.587	56.198	1	br-j59	48.268	46.75	47.496	1
br-c02	53.862	50.912	52.345	1	br-j60	56.399	50.902	53.509	1
br-c04	58.914	51.818	55.138	1	br-j70	55.876	53.703	54.767	1
br-d01	48.837	44.168	46.385	1	br-k01	58.333	54.804	56.513	1
br-d02	48.704	42.533	45.409	1	br-k02	50.462	48.063	49.233	1
br-d03	59.86	54.315	56.952	1	br-k03	53.033	47.773	50.265	1
br-d04	51.868	46.918	49.268	1	br-k04	55.803	49.9	52.686	1
br-e01	53.004	49.698	51.297	1	br-k05	53.763	47.984	50.709	1
br-e02	51.076	46.942	48.921	1	br-k06	61.322	56.771	58.958	1
br-e04	51.294	46.382	48.714	1	br-k07	52.594	49.01	50.738	1
br-e21	46.63	42.105	44.252	1	br-k08	52.845	48.598	50.632	1
br-e24	49.019	45.833	47.372	1	br-k09	56.631	52.436	54.452	1
br-e29	50.205	47.104	48.605	1	br-k10	56.179	53.38	54.743	1
br-f03	52.173	48.618	50.332	1	br-k11	60.998	58.201	59.566	1
br-f10	54.716	51.683	53.156	1	br-k12	55.619	53.284	54.426	1
br-f19	52.834	49.525	51.126	1	br-k13	56.954	51.794	54.251	1
br-f43	52.93	49.122	50.954	1	br-k14	53.51	50.81	52.125	1
br-g01	54.567	47.89	51.01	1	br-k15	54.832	52.851	53.823	1
br-g11	54.809	51.89	53.309	1	br-k16	58.958	55.599	57.229	1
br-g15	49.892	46.015	47.875	1	br-k17	52.608	48.692	50.574	1
br-h01	52.601	49.908	51.219	1	br-k18	50.526	45.54	47.903	1
br-j01	63.461	59.045	61.173	1	br-k19	52.62	48.643	50.553	1
br-j02	66.37	64.869	65.61	1	**Media**	**53.478**	**49.805**	51.561	**1.000**
br-j03	49.896	44.731	47.172	1	**F(%)**			**51.576**	
br-j04	57.637	52.996	55.219	1	**MIN**	30.810	27.078	28.823	1.000
br-j05	53.891	50.919	52.362	1	**MAX**	71.261	68.848	70.033	1.000
br-j06	47.954	46.475	47.202	1					
br-j07	45.882	43.173	44.486	1					
br-j08	50.185	46.404	48.22	1					
br-j09	59.215	58.076	58.639	1					
br-j10	60.035	55.833	57.857	1					
br-j11	54.088	50.588	52.279	1					
br-j12	54.634	50.111	52.274	1					
br-j13	46.897	46.223	46.557	1					
br-j14	56.532	53.483	54.965	1					
br-j15	59.803	55.555	57.6	1					
br-j16	71.261	68.848	70.033	1					
br-j17	51.976	50.674	51.316	1					
br-j18	46.341	36.45	40.804	1					
br-j19	39.189	36.708	37.907	1					
br-j20	30.81	27.078	28.823	1					
br-j22	62.5	62.5	62.5	1					

Tabela B.4: Rezultatele algoritmului WSDTool pe corpusul SemCor2.0 în limba română. Inventarul de sensuri este dat de ILI iar evaluarea este strictă.

Fișier	P(%)	R(%)	F(%)	S/C	Fișier	P(%)	R(%)	F(%)	S/C
br-a01	57.746	55.728	56.719	1	br-j23	61.181	56.974	59.002	1
br-a02	66.585	63.805	65.165	1	br-j37	56.124	53.671	54.87	1
br-a11	68.847	65.773	67.274	1	br-j52	58.101	51.12	54.387	1
br-a12	67.312	64.953	66.111	1	br-j53	66.334	62.711	64.471	1
br-a13	67.34	61.349	64.205	1	br-j54	65.36	61.434	63.336	1
br-a14	67.733	64.303	65.973	1	br-j55	67.269	64.922	66.074	1
br-a15	66.573	61.439	63.903	1	br-j56	68.421	62.801	65.49	1
br-b13	68.181	62.264	65.088	1	br-j57	63.507	55.601	59.291	1
br-b20	67.167	59.688	63.207	1.002	br-j58	58.901	56.066	57.448	1
br-c01	71.776	67.66	69.657	1	br-j59	66.666	64.57	65.601	1
br-c02	66.523	62.88	64.65	1	br-j60	68	61.371	64.515	1
br-c04	71.576	62.954	66.988	1	br-j70	68.208	65.555	66.855	1
br-d01	59.196	53.537	56.224	1	br-k01	69.886	65.658	67.706	1
br-d02	58.808	51.357	54.83	1	br-k02	63.77	60.739	62.217	1
br-d03	70.301	63.789	66.886	1	br-k03	63.82	57.489	60.489	1
br-d04	70.769	64.015	67.222	1	br-k04	68.973	61.676	65.12	1
br-e01	67.167	62.977	65.004	1	br-k05	68.602	61.228	64.705	1
br-e02	62.818	57.733	60.168	1	br-k06	72.945	67.532	70.134	1
br-e04	65.176	58.936	61.899	1	br-k07	66.745	62.197	64.39	1
br-e21	57.741	52.138	54.796	1	br-k08	67.479	62.056	64.653	1
br-e24	62.566	58.5	60.464	1	br-k09	69.473	64.327	66.801	1
br-e29	62.345	58.494	60.358	1	br-k10	66.292	62.989	64.598	1
br-f03	61.66	57.458	59.484	1	br-k11	71.534	68.253	69.854	1
br-f10	65.408	61.782	63.543	1	br-k12	67.238	64.416	65.796	1
br-f19	63.157	59.203	61.116	1	br-k13	71.428	64.957	68.038	1
br-f43	65.406	60.701	62.965	1	br-k14	62.618	59.459	60.997	1
br-g01	64.903	56.962	60.673	1	br-k15	67.258	64.828	66.02	1
br-g11	67.876	64.261	66.019	1	br-k16	70.833	66.797	68.755	1
br-g15	58.963	54.382	56.579	1	br-k17	63.478	58.752	61.023	1.002
br-h01	63.583	60.329	61.913	1	br-k18	62.105	55.977	58.881	1.002
br-j01	70.299	65.407	67.764	1	br-k19	63.102	58.333	60.623	1
br-j02	71.53	69.913	70.712	1	**Media**	**65.059**	**60.572**	62.717	**1.000**
br-j03	58.762	52.68	55.555	1	**F(%)**			**62.735**	
br-j04	67.006	61.61	64.194	1	**MIN**	42.432	37.292	39.696	1.000
br-j05	64.202	60.661	62.381	1	**MAX**	80.000	80.000	80.000	1.002
br-j06	54.318	52.643	53.467	1					
br-j07	53.137	50	51.52	1					
br-j08	62.037	57.363	59.608	1					
br-j09	61.764	60.576	61.164	1					
br-j10	69.354	64.5	66.838	1					
br-j11	66.876	62.549	64.64	1					
br-j12	61.219	56.152	58.576	1					
br-j13	60.583	59.712	60.144	1					
br-j14	63.895	60.449	62.124	1					
br-j15	69.607	64.663	67.043	1					
br-j16	75.934	73.363	74.626	1					
br-j17	63.241	61.657	62.438	1					
br-j18	56.402	44.364	49.663	1					
br-j19	61.389	57.504	59.383	1					
br-j20	42.432	37.292	39.696	1					
br-j22	80	80	80	1					

Tabela B.5: Rezultatele algoritmului WSDTool pe corpusul SemCor2.0 în limba română. Inventarul de sensuri este dat de categoriile SUMO iar evaluarea este strictă.

Fișier	P(%)	R(%)	F(%)	S/C	Fișier	P(%)	R(%)	F(%)	S/C
br-a01	86.317	83.3	84.781	1.175	br-j23	85.864	79.96	82.806	1.035
br-a02	84.019	80.51	82.227	1.128	br-j37	87.751	83.916	85.79	1.191
br-a11	86.604	82.738	84.626	1.093	br-j52	91.203	80.244	85.373	1.053
br-a12	85.23	82.242	83.709	1.089	br-j53	90.637	85.687	88.092	1.059
br-a13	83.164	75.766	79.292	1.09	br-j54	83.917	78.875	81.317	1.065
br-a14	81.6	77.468	79.48	1.128	br-j55	89.759	86.627	88.165	1.154
br-a15	84.401	77.892	81.015	1.069	br-j56	86.052	78.985	82.367	1.239
br-b13	81.611	74.528	77.908	1.082	br-j57	78.909	69.087	73.672	1.111
br-b20	87.218	77.505	82.075	1.127	br-j58	85.494	81.38	83.386	1.116
br-c01	85.888	80.963	83.352	1.177	br-j59	88.528	85.744	87.113	1.047
br-c02	84.334	79.716	81.96	1.111	br-j60	86.8	78.339	82.352	1.226
br-c04	86.304	75.909	80.773	1.147	br-j70	88.439	85	86.685	1.078
br-d01	76.321	69.024	72.489	1.082	br-k01	87.31	82.028	84.586	1.138
br-d02	89.637	78.28	83.574	1.113	br-k02	87.615	83.45	85.481	1.12
br-d03	90.255	81.894	85.871	1.206	br-k03	85.617	77.125	81.149	1.132
br-d04	89.01	80.516	84.55	1.061	br-k04	87.053	77.844	82.191	1.223
br-e01	84.12	78.873	81.412	1.04	br-k05	89.892	80.23	84.786	1.105
br-e02	84.931	78.057	81.349	1.111	br-k06	90.581	83.858	87.089	1.186
br-e04	86.588	78.297	82.234	1.07	br-k07	88.679	82.637	85.551	1.068
br-e21	76.32	68.914	72.428	1.125	br-k08	87.804	80.747	84.127	1.113
br-e24	86.987	81.333	84.065	1.122	br-k09	88.842	82.261	85.424	1.141
br-e29	80.041	75.096	77.489	1.078	br-k10	88.576	84.163	86.313	1.155
br-f03	82.411	76.795	79.503	1.077	br-k11	90.388	86.243	88.266	1.162
br-f10	83.438	78.811	81.058	1.067	br-k12	87.809	84.124	85.927	1.129
br-f19	86.437	81.024	83.643	1.113	br-k13	90.037	81.88	85.764	1.118
br-f43	83.931	77.894	80.799	1.224	br-k14	84.06	79.819	81.884	1.127
br-g01	82.692	72.573	77.302	1.069	br-k15	86.982	83.84	85.382	1.136
br-g11	89.11	84.364	86.672	1.059	br-k16	87.916	82.907	85.338	1.183
br-g15	80.777	74.501	77.512	1.086	br-k17	85.652	79.275	82.34	1.106
br-h01	82.466	78.244	80.299	1.059	br-k18	86.315	77.798	81.835	1.138
br-j01	83.333	77.534	80.328	1.049	br-k19	86.163	79.651	82.779	1.079
br-j02	87.366	85.391	86.367	1.193	**Media**	**85.015**	**79.124**	81.940	**1.110**
br-j03	76.288	68.391	72.123	1.018	**F(%)**			**81.964**	
br-j04	88.798	81.647	85.072	1.099	**MIN**	62.432	54.869	58.406	1.018
br-j05	86.575	81.801	84.12	1.044	**MAX**	91.203	87.500	88.266	1.239
br-j06	70.681	68.502	69.574	1.056					
br-j07	73.333	69.003	71.102	1.047					
br-j08	82.592	76.369	79.358	1.153					
br-j09	86.862	85.192	86.018	1.08					
br-j10	87.455	81.333	84.282	1.137					
br-j11	81.97	76.666	79.229	1.205					
br-j12	89.024	81.655	85.18	1.158					
br-j13	76.642	75.539	76.086	1.065					
br-j14	84.085	79.55	81.754	1.123					
br-j15	87.647	81.42	84.418	1.129					
br-j16	87.383	84.424	85.878	1.051					
br-j17	86.363	84.2	85.267	1.067					
br-j18	76.219	59.952	67.113	1.085					
br-j19	85.907	80.47	83.099	1.065					
br-j20	62.432	54.869	58.406	1.043					
br-j22	87.5	87.5	87.5	1.05					

Tabela B.6: Rezultatele algoritmului WSDTool pe corpusul SemCor2.0 în limba română. Inventarul de sensuri este dat de domeniile IRST iar evaluarea este strictă.

Anexa C

Tabelele de mai jos reprezintă performaţa algoritmului SynWSD pe corpusul paralel englez-român SemCor2.0. Coloana **Fişier** precizează fişierul corpusului pe care s-a rulat algoritmul, după care urmează valorile procentuale ale preciziei (**P(%)**), recall-ului (**R(%)**) şi ale f-measure (**F(%)**) iar coloana **S/C** indică numărul mediu de etichete semantice per cuvânt dezambiguizat (= numărul total de etichete semantice date de algoritm împărţit la numărul de ocurenţe ale tuturor cuvintelor dezambiguizate de algoritm). Tabelul este împărţit în două coloane, a doua în continuarea primeia.

Fişier	P(%)	R(%)	F(%)	S/C	Fişier	P(%)	R(%)	F(%)	S/C
br-a01	50.295	50.295	50.295	1.714	br-j23	47.669	47.531	47.599	1.489
br-a02	47.916	47.916	47.916	1.593	br-j37	45.78	45.78	45.779	1.658
br-a11	47.54	47.476	47.507	1.786	br-j52	36.2	36.071	36.135	2.124
br-a12	43.451	43.451	43.451	2.03	br-j53	46.888	46.838	46.862	1.674
br-a13	46.348	46.283	46.315	1.869	br-j54	46.59	46.59	46.59	1.696
br-a14	45.527	45.345	45.435	1.807	br-j55	45.744	45.69	45.716	1.897
br-a15	44.281	44.223	44.251	1.942	br-j56	49.925	49.851	49.887	1.774
br-b13	43.441	43.35	43.395	1.648	br-j57	48.627	48.521	48.573	1.726
br-b20	48.571	48.398	48.484	1.79	br-j58	47.898	47.776	47.836	1.765
br-c01	50.582	50.465	50.523	1.724	br-j59	46.971	46.757	46.863	1.714
br-c02	47.306	47.255	47.28	1.589	br-j60	52.336	52.336	52.336	1.715
br-c04	44.634	44.417	44.525	1.77	br-j70	55.7	55.7	55.7	1.423
br-d01	41.995	41.995	41.995	1.879	br-k01	45.474	45.274	45.373	1.789
br-d02	49.492	49.436	49.463	1.801	br-k02	44.938	44.542	44.739	1.87
br-d03	55.251	55.01	55.13	1.658	br-k03	48.074	47.955	48.014	1.859
br-d04	45.768	45.605	45.686	1.883	br-k04	46.658	46.441	46.549	1.719
br-e01	44.92	44.769	44.844	1.851	br-k05	42.874	42.718	42.795	1.971
br-e02	46.645	46.45	46.547	1.72	br-k06	50.561	50.561	50.561	1.686
br-e04	43.505	43.368	43.436	1.614	br-k07	39.197	38.598	38.895	1.864
br-e21	43.537	43.495	43.515	1.66	br-k08	44.114	43.95	44.031	1.939
br-e24	42.624	42.451	42.537	1.885	br-k09	45.739	45.511	45.624	2.126
br-e29	53.304	53.246	53.274	1.611	br-k10	47.031	46.759	46.894	1.827
br-f03	45.842	45.842	45.841	1.733	br-k11	45.804	45.538	45.67	1.906
br-f10	51.657	51.305	51.48	1.651	br-k12	47.859	47.702	47.78	1.744
br-f19	51.609	51.443	51.525	1.717	br-k13	48.748	48.69	48.718	1.887
br-f43	44.882	44.787	44.834	1.759	br-k14	43.728	43.523	43.625	1.944
br-g01	48.39	48.123	48.256	1.754	br-k15	45.165	45	45.082	2.003
br-g11	46.305	46.209	46.256	1.649	br-k16	53.703	53.703	53.703	1.728
br-g15	49.946	49.733	49.839	1.705	br-k17	46.761	46.64	46.7	1.748
br-h01	46.248	46.248	46.247	1.53	br-k18	43.558	43.398	43.477	1.991
br-j01	51.152	51.093	51.122	1.554	br-k19	43.675	43.623	43.648	1.889
br-j02	41.29	41.29	41.29	1.463	**Media**	**47.859**	**47.746**	47.802	**1.729**
br-j03	42.403	42.355	42.378	1.667	**F(%)**			**47.803**	
br-j04	48.272	48.272	48.272	1.581	**MIN**	36.200	36.071	36.135	1.395
br-j05	51.175	51.129	51.151	1.49	**MAX**	60.358	60.358	60.358	2.126
br-j06	49.732	49.626	49.678	1.555					
br-j07	52.351	52.298	52.324	1.553					
br-j08	51.362	51.207	51.284	1.687					
br-j09	55.057	54.942	54.999	1.395					
br-j10	53.676	53.451	53.563	1.678					
br-j11	49.888	49.721	49.804	1.645					
br-j12	52.687	52.687	52.687	1.518					
br-j13	41.558	41.513	41.535	1.636					
br-j14	60.065	60.065	60.065	1.498					
br-j15	59.057	58.894	58.975	1.496					
br-j16	60.358	60.358	60.358	1.477					
br-j17	55.807	55.754	55.78	1.488					
br-j18	49.537	49.537	49.537	1.837					
br-j19	46.335	46.28	46.307	1.721					
br-j20	46.891	46.77	46.83	1.965					
br-j22	50	50	50	1.435					

Tabela C.1: Rezultatele algoritmului SynWSD pe corpusul SemCor2.0 în limba engleză (măsura de atracţie semantică este `mi`). Inventarul de sensuri este dat de ILI iar evaluarea este relaxată.

Fişier	P(%)	R(%)	F(%)	S/C	Fişier	P(%)	R(%)	F(%)	S/C
br-a01	55.595	55.135	55.364	1.327	br-j23	55.145	54.985	55.064	1.316
br-a02	56.578	55.989	56.281	1.276	br-j37	59.196	59.071	59.133	1.331
br-a11	55.845	55.388	55.615	1.295	br-j52	51.015	50.833	50.923	1.464
br-a12	53.152	52.631	52.89	1.354	br-j53	60.925	60.664	60.794	1.4
br-a13	54.943	54.558	54.749	1.336	br-j54	53.381	52.995	53.187	1.3
br-a14	58.634	58.244	58.438	1.338	br-j55	56.989	56.316	56.65	1.373
br-a15	50.133	49.8	49.965	1.346	br-j56	60.804	60.534	60.668	1.321
br-b13	58.324	57.591	57.955	1.322	br-j57	57.883	57.502	57.691	1.366
br-b20	59.952	59.667	59.809	1.381	br-j58	57.179	56.162	56.665	1.416
br-c01	61.313	60.813	61.061	1.327	br-j59	59.069	56.313	57.658	1.412
br-c02	59.263	58.88	59.07	1.313	br-j60	61.25	61.059	61.154	1.323
br-c04	57.212	56.796	57.003	1.361	br-j70	61.895	61.028	61.458	1.178
br-d01	51.335	51.276	51.305	1.434	br-k01	58.192	56.593	57.381	1.336
br-d02	58.531	58.333	58.431	1.412	br-k02	55.741	55.126	55.431	1.338
br-d03	63.215	62.527	62.869	1.341	br-k03	55.555	55.142	55.347	1.393
br-d04	49.821	49.643	49.731	1.455	br-k04	55.005	54.492	54.747	1.335
br-e01	51.473	51.068	51.269	1.341	br-k05	53.667	53.276	53.47	1.37
br-e02	58.5	57.828	58.162	1.336	br-k06	61.041	60.561	60.8	1.26
br-e04	54.718	54.315	54.515	1.359	br-k07	51.491	50.573	51.027	1.32
br-e21	56.183	56.019	56.1	1.38	br-k08	52.292	52.098	52.194	1.365
br-e24	56.818	55.724	56.265	1.347	br-k09	53.4	52.867	53.132	1.395
br-e29	58.488	57.792	58.137	1.345	br-k10	57.579	56.712	57.142	1.354
br-f03	56.913	56.609	56.76	1.36	br-k11	58.158	57.821	57.989	1.265
br-f10	58.16	57.434	57.794	1.297	br-k12	58.849	58.205	58.525	1.308
br-f19	54.946	54.652	54.798	1.343	br-k13	59.952	59.88	59.915	1.313
br-f43	54.555	54.148	54.35	1.351	br-k14	52.669	51.808	52.234	1.372
br-g01	55.902	55.408	55.653	1.373	br-k15	55.995	55.243	55.616	1.351
br-g11	56.666	56.49	56.577	1.373	br-k16	63.06	62.854	62.956	1.261
br-g15	60.58	60.192	60.385	1.381	br-k17	56.901	56.459	56.679	1.368
br-h01	50.784	50.727	50.755	1.322	br-k18	49.753	49.388	49.569	1.401
br-j01	67.497	65.477	66.471	1.316	br-k19	55.234	54.707	54.969	1.452
br-j02	61.075	59.323	60.186	1.22	**Media**	**57.831**	**57.236**	57.530	**1.334**
br-j03	59.701	58.89	59.292	1.365	**F(%)**			**57.532**	
br-j04	59.65	59.044	59.345	1.327	**MIN**	49.753	49.388	49.569	1.178
br-j05	62.764	61.517	62.134	1.252	**MAX**	69.264	69.037	69.150	1.464
br-j06	63.939	63.393	63.664	1.266					
br-j07	65.657	64.249	64.945	1.292					
br-j08	58.751	57.746	58.244	1.288					
br-j09	62.788	62.33	62.558	1.191					
br-j10	62.724	62.133	62.427	1.286					
br-j11	55.995	55.183	55.586	1.307					
br-j12	67.23	67.017	67.123	1.257					
br-j13	54.783	52	53.355	1.321					
br-j14	69.264	69.037	69.15	1.227					
br-j15	66.104	65.069	65.582	1.191					
br-j16	68.125	67.973	68.048	1.181					
br-j17	63.981	63.679	63.829	1.26					
br-j18	53.236	53.236	53.236	1.396					
br-j19	57.363	57.024	57.192	1.34					
br-j20	56.058	54.392	55.212	1.423					
br-j22	57.692	57.692	57.692	1.397					

Tabela C.2: Rezultatele algoritmului SynWSD pe corpusul SemCor2.0 în limba engleză (măsura de atracţie semantică este `mi`). Inventarul de sensuri este dat de categoriile SUMO iar evaluarea este strictă.

Fişier	P(%)	R(%)	F(%)	S/C	Fişier	P(%)	R(%)	F(%)	S/C
br-a01	79.669	79.574	79.621	1.078	br-j23	81.015	77.25	79.087	1.046
br-a02	74.442	73.958	74.199	1.072	br-j37	81.104	80.59	80.846	1.062
br-a11	67.808	67.53	67.668	1.11	br-j52	79.569	79.285	79.426	1.094
br-a12	70.955	70.869	70.911	1.1	br-j53	86.373	86.28	86.326	1.09
br-a13	74.402	74.193	74.297	1.112	br-j54	82.626	82.541	82.583	1.085
br-a14	73.889	73.005	73.444	1.113	br-j55	77.777	77.685	77.73	1.133
br-a15	75.898	75.697	75.797	1.085	br-j56	73.919	73.59	73.754	1.065
br-b13	72.689	72.46	72.574	1.076	br-j57	79.56	79.299	79.429	1.089
br-b20	82.078	81.494	81.784	1.063	br-j58	82.038	81.829	81.933	1.109
br-c01	77.855	77.674	77.764	1.072	br-j59	84.342	83.959	84.15	1.082
br-c02	77.993	77.825	77.908	1.081	br-j60	76.586	76.427	76.506	1.071
br-c04	76.126	75.849	75.987	1.071	br-j70	82.056	82.056	82.056	1.073
br-d01	80.626	80.626	80.626	1.092	br-k01	77.593	77.252	77.422	1.114
br-d02	86.56	85.585	86.069	1.084	br-k02	78.642	77.949	78.293	1.114
br-d03	84.118	83.66	83.888	1.067	br-k03	77.238	76.951	77.094	1.12
br-d04	81.622	81.235	81.428	1.095	br-k04	79.929	79.463	79.695	1.073
br-e01	75.48	75.14	75.309	1.097	br-k05	80.267	79.975	80.12	1.098
br-e02	72.117	71.816	71.966	1.111	br-k06	77.303	77.303	77.303	1.092
br-e04	79.091	78.842	78.966	1.065	br-k07	80.983	79.745	80.359	1.098
br-e21	72.983	72.912	72.947	1.081	br-k08	78.731	78.148	78.438	1.104
br-e24	75.05	74.67	74.859	1.1	br-k09	77.443	77.057	77.249	1.137
br-e29	76.096	75.108	75.598	1.078	br-k10	77.712	77.083	77.396	1.1
br-f03	81.41	81.236	81.322	1.084	br-k11	74.825	74.391	74.607	1.106
br-f10	73.714	73.212	73.462	1.104	br-k12	77.826	77.571	77.698	1.087
br-f19	80.266	77.433	78.824	1.089	br-k13	78.297	77.738	78.016	1.109
br-f43	75.969	75	75.481	1.09	br-k14	75.558	75.029	75.292	1.115
br-g01	80	79.47	79.734	1.075	br-k15	78.58	78.292	78.435	1.122
br-g11	83.141	82.45	82.794	1.073	br-k16	77.342	77.342	77.342	1.064
br-g15	84.573	83.671	84.119	1.106	br-k17	75.616	75.322	75.468	1.114
br-h01	72.004	72.004	72.004	1.061	br-k18	82.432	82.029	82.23	1.138
br-j01	74.193	74.108	74.15	1.091	br-k19	80.884	80.691	80.787	1.131
br-j02	71.149	71.004	71.076	1.111	**Media**	**78.042**	**77.658**	77.848	**1.090**
br-j03	75	74.745	74.872	1.088	**F(%)**			**77.849**	
br-j04	78.76	78.76	78.76	1.076	**MIN**	66.321	66.149	66.234	1.038
br-j05	82.097	82.023	82.059	1.083	**MAX**	86.560	86.280	86.326	1.143
br-j06	76.256	76.093	76.174	1.098					
br-j07	74.335	74.259	74.296	1.086					
br-j08	82.251	81.589	81.918	1.078					
br-j09	81.751	81.581	81.665	1.088					
br-j10	75.63	75.313	75.471	1.1					
br-j11	75.838	75.585	75.711	1.08					
br-j12	81.223	81.138	81.18	1.093					
br-j13	74.891	74.81	74.85	1.143					
br-j14	82.932	82.932	82.932	1.071					
br-j15	82.902	82.672	82.786	1.062					
br-j16	77.578	77.491	77.534	1.073					
br-j17	83.773	83.773	83.773	1.068					
br-j18	76.618	76.618	76.618	1.087					
br-j19	81.323	81.227	81.274	1.08					
br-j20	66.321	66.149	66.234	1.095					
br-j22	71.794	71.794	71.794	1.038					

Tabela C.3: Rezultatele algoritmului SynWSD pe corpusul SemCor2.0 în limba engleză (măsura de atracţie semantică este `dice`). Inventarul de sensuri este dat de domeniile IRST iar evaluarea este strictă.

Fişier	P(%)	R(%)	F(%)	S/C	Fişier	P(%)	R(%)	F(%)	S/C
br-a01	42.519	41.941	42.228	1.742	br-j23	40.319	39.685	39.999	1.7
br-a02	45.327	45.011	45.168	1.82	br-j37	45.759	45.279	45.517	1.701
br-a11	51.204	50.595	50.897	2.237	br-j52	32.238	31.975	32.105	1.691
br-a12	45.047	44.626	44.835	1.903	br-j53	36.257	35.028	35.631	1.727
br-a13	41.795	41.411	41.602	2.13	br-j54	43.055	42.054	42.548	1.704
br-a14	44.473	43.797	44.132	2.249	br-j55	39.299	39.147	39.222	1.912
br-a15	41.602	41.388	41.494	2.033	br-j56	48.058	47.826	47.941	1.859
br-b13	40.874	40.566	40.719	1.771	br-j57	39.495	39.004	39.247	1.663
br-b20	40	39.643	39.82	1.842	br-j58	40.254	39.748	39.999	1.8
br-c01	50.943	49.541	50.232	1.79	br-j59	41.505	40.461	40.976	1.896
br-c02	40.534	39.959	40.244	1.742	br-j60	43.566	42.779	43.168	1.707
br-c04	42.857	41.59	42.213	1.915	br-j70	46.468	46.296	46.381	1.771
br-d01	32.239	31.931	32.084	1.611	br-k01	33.574	33.096	33.333	1.916
br-d02	46.33	45.701	46.013	1.848	br-k02	42.882	42.429	42.654	2.016
br-d03	49.681	49.263	49.471	1.819	br-k03	35.772	35.627	35.699	1.713
br-d04	40.12	39.562	39.839	1.606	br-k04	41.129	40.718	40.922	1.937
br-e01	42.682	42.253	42.466	1.851	br-k05	33.398	33.013	33.204	1.794
br-e02	44.202	43.884	44.042	1.719	br-k06	40.796	39.888	40.336	1.88
br-e04	41.416	41.063	41.238	1.723	br-k07	33.105	31.868	32.474	1.803
br-e21	34.949	34.375	34.659	1.615	br-k08	39.347	38.317	38.825	1.675
br-e24	36.744	36.5	36.621	1.77	br-k09	34.122	33.723	33.921	1.875
br-e29	47.544	46.718	47.127	1.616	br-k10	40.429	40.213	40.32	1.806
br-f03	39.405	39.042	39.222	1.667	br-k11	37.07	36.155	36.606	1.768
br-f10	39.759	39.207	39.481	1.791	br-k12	42.486	41.788	42.134	1.771
br-f19	47.692	47.058	47.372	1.655	br-k13	40.94	40.17	40.551	1.773
br-f43	36.363	35.789	36.073	1.623	br-k14	38.138	37.657	37.895	1.782
br-g01	42.795	41.983	42.385	1.8	br-k15	36.641	36.501	36.57	1.845
br-g11	41.796	41.58	41.687	1.683	br-k16	47.326	46.954	47.139	1.906
br-g15	44.489	43.426	43.951	1.787	br-k17	39.183	38.631	38.905	1.714
br-h01	42.49	42.413	42.451	1.664	br-k18	37.213	37.001	37.106	1.816
br-j01	52.286	52.286	52.286	1.667	br-k19	37.281	37.209	37.244	1.836
br-j02	44.833	44.521	44.676	1.633	**Media**	**41.588**	**41.070**	41.326	**1.787**
br-j03	37.36	37.153	37.256	1.767	**F(%)**			**41.327**	
br-j04	42.075	41.76	41.916	1.939	**MIN**	28.192	27.790	27.989	1.282
br-j05	42.35	41.727	42.036	1.86	**MAX**	52.286	52.286	52.286	2.249
br-j06	35.903	35.903	35.903	1.702					
br-j07	42.537	42.066	42.3	1.777					
br-j08	42.782	42.123	42.449	1.629					
br-j09	46.538	46.538	46.538	1.725					
br-j10	49.831	49.166	49.496	1.763					
br-j11	45.454	45.098	45.275	1.845					
br-j12	45.265	43.847	44.544	1.928					
br-j13	38.979	38.489	38.732	1.892					
br-j14	46.818	46.292	46.553	1.672					
br-j15	50.642	50.273	50.456	1.816					
br-j16	52.164	51.693	51.927	1.719					
br-j17	43.774	43.352	43.561	1.762					
br-j18	45.873	45.323	45.596	1.774					
br-j19	35.714	35.262	35.486	1.668					
br-j20	28.192	27.79	27.989	1.92					
br-j22	35.897	35	35.442	1.282					

Tabela C.4: Rezultatele algoritmului SynWSD pe corpusul SemCor2.0 în limba română (măsura de atracţie semantică este `prob`). Inventarul de sensuri este dat de ILI iar evaluarea este relaxată.

Fişier	P(%)	R(%)	F(%)	S/C	Fişier	P(%)	R(%)	F(%)	S/C
br-a01	52.952	52.233	52.59	1.419	br-j23	43.912	43.222	43.564	1.377
br-a02	51.168	50.812	50.989	1.415	br-j37	56.36	55.769	56.062	1.383
br-a11	53.915	53.273	53.592	1.433	br-j52	41.273	40.936	41.103	1.679
br-a12	57.547	57.009	57.276	1.436	br-j53	49.902	48.21	49.041	1.461
br-a13	44.582	44.171	44.375	1.461	br-j54	43.055	42.054	42.548	1.325
br-a14	50.385	49.62	49.999	1.465	br-j55	52.918	52.713	52.815	1.431
br-a15	49.095	48.843	48.968	1.516	br-j56	51.699	51.449	51.573	1.446
br-b13	50.38	50	50.189	1.395	br-j57	53.151	52.489	52.817	1.447
br-b20	49.213	48.775	48.993	1.442	br-j58	54.872	54.184	54.525	1.453
br-c01	55.66	54.128	54.883	1.349	br-j59	45.922	44.863	45.386	1.435
br-c02	52.057	51.318	51.684	1.397	br-j60	52.573	51.624	52.094	1.402
br-c04	49.648	48.181	48.903	1.449	br-j70	59.107	58.888	58.997	1.436
br-d01	41.698	41.3	41.498	1.48	br-k01	49.097	48.398	48.744	1.427
br-d02	46.788	46.153	46.468	1.472	br-k02	46.975	46.478	46.725	1.403
br-d03	55.414	54.947	55.179	1.414	br-k03	48.78	48.582	48.68	1.459
br-d04	41.129	40.556	40.84	1.481	br-k04	48.991	48.502	48.745	1.409
br-e01	44.308	43.863	44.084	1.378	br-k05	45.631	45.105	45.366	1.444
br-e02	51.811	51.438	51.623	1.405	br-k06	56.166	54.916	55.533	1.352
br-e04	50.107	49.787	49.946	1.379	br-k07	43.15	41.538	42.328	1.397
br-e21	48.662	47.861	48.258	1.372	br-k08	46.833	45.607	46.211	1.433
br-e24	47.147	46.833	46.989	1.414	br-k09	49.508	49.122	49.314	1.473
br-e29	53.045	52.123	52.579	1.367	br-k10	49.016	48.754	48.884	1.411
br-f03	45.539	45.119	45.328	1.429	br-k11	52.441	51.146	51.785	1.325
br-f10	54.325	53.465	53.891	1.41	br-k12	54.259	53.467	53.86	1.442
br-f19	53.358	52.751	53.052	1.416	br-k13	51.13	50.256	50.689	1.441
br-f43	49.732	48.947	49.336	1.401	br-k14	50.364	49.729	50.044	1.436
br-g01	48.172	47.257	47.71	1.47	br-k15	49.618	49.429	49.523	1.486
br-g11	49.222	48.969	49.095	1.455	br-k16	58.019	57.563	57.79	1.362
br-g15	44.489	43.426	43.951	1.385	br-k17	50.305	49.698	49.999	1.397
br-h01	50	49.908	49.953	1.36	br-k18	47.709	47.438	47.573	1.496
br-j01	57.256	57.256	57.255	1.379	br-k19	46.601	46.511	46.555	1.526
br-j02	65.499	65.043	65.27	1.343	**Media**	**51.188**	**50.570**	50.876	**1.413**
br-j03	57.434	57.116	57.274	1.395	**F(%)**			**50.877**	
br-j04	49.056	48.689	48.871	1.415	**MIN**	41.129	40.556	40.840	1.282
br-j05	47.672	47.058	47.363	1.335	**MAX**	65.499	65.043	65.270	1.679
br-j06	60.792	60.792	60.792	1.42					
br-j07	50.746	50.184	50.463	1.333					
br-j08	48.611	47.945	48.275	1.394					
br-j09	59.23	59.23	59.23	1.336					
br-j10	60.202	59.5	59.848	1.327					
br-j11	56.213	55.882	56.047	1.422					
br-j12	51.501	49.888	50.681	1.406					
br-j13	48.087	47.482	47.782	1.429					
br-j14	57.499	56.853	57.174	1.35					
br-j15	57.299	57.194	57.246	1.326					
br-j16	64.692	64.108	64.398	1.296					
br-j17	56.225	55.684	55.953	1.328					
br-j18	52.184	51.558	51.869	1.449					
br-j19	47.619	47.016	47.315	1.353					
br-j20	52.289	51.543	51.913	1.513					
br-j22	56.41	55	55.696	1.282					

Tabela C.5: Rezultatele algoritmului SynWSD pe corpusul SemCor2.0 în limba română (măsura de atracţie semantică este `mi`). Inventarul de sensuri este dat de categoriile SUMO iar evaluarea este strictă.

Fişier	P(%)	R(%)	F(%)	S/C	Fişier	P(%)	R(%)	F(%)	S/C
br-a01	78.346	77.281	77.809	1.088	br-j23	82.634	81.335	81.979	1.049
br-a02	69.392	68.909	69.149	1.072	br-j37	77.385	76.573	76.976	1.051
br-a11	77.409	76.488	76.945	1.075	br-j52	82.956	82.281	82.617	1.149
br-a12	78.773	78.037	78.403	1.058	br-j53	80.701	77.966	79.309	1.066
br-a13	79.876	79.141	79.506	1.102	br-j54	81.547	79.651	80.587	1.061
br-a14	76.349	75.189	75.764	1.071	br-j55	78.404	78.1	78.251	1.083
br-a15	79.844	79.434	79.638	1.051	br-j56	76.699	76.328	76.513	1.082
br-b13	74.144	73.584	73.862	1.07	br-j57	78.361	77.385	77.869	1.134
br-b20	80.449	79.732	80.088	1.08	br-j58	76.906	75.941	76.42	1.076
br-c01	81.603	79.357	80.464	1.101	br-j59	77.896	76.1	76.987	1.096
br-c02	74.074	73.022	73.544	1.106	br-j60	72.61	71.299	71.948	1.091
br-c04	78.922	76.59	77.738	1.114	br-j70	84.572	84.259	84.415	1.089
br-d01	75.868	75.143	75.503	1.092	br-k01	75.27	74.199	74.73	1.131
br-d02	84.403	83.257	83.826	1.066	br-k02	75.8	75	75.397	1.071
br-d03	83.227	82.526	82.875	1.082	br-k03	77.642	77.327	77.484	1.121
br-d04	75.403	74.353	74.874	1.096	br-k04	79.233	78.443	78.836	1.09
br-e01	69.715	69.014	69.362	1.075	br-k05	77.864	76.967	77.412	1.091
br-e02	75.724	75.179	75.45	1.114	br-k06	75.332	73.654	74.483	1.079
br-e04	82.655	82.127	82.39	1.07	br-k07	80.365	77.362	78.834	1.105
br-e21	68.729	67.598	68.158	1.061	br-k08	77.735	75.7	76.704	1.147
br-e24	72.986	72.5	72.742	1.08	br-k09	79.764	79.142	79.451	1.137
br-e29	74.459	73.166	73.806	1.082	br-k10	75.134	74.733	74.932	1.073
br-f03	79.553	78.821	79.185	1.104	br-k11	72.513	70.723	71.606	1.097
br-f10	72.489	71.485	71.983	1.124	br-k12	76.851	75.729	76.285	1.079
br-f19	80.998	80.075	80.533	1.076	br-k13	76.695	75.384	76.033	1.113
br-f43	67.914	66.842	67.373	1.069	br-k14	73.54	72.612	73.073	1.122
br-g01	79.569	78.059	78.806	1.103	br-k15	75	74.714	74.856	1.091
br-g11	85.146	84.707	84.925	1.093	br-k16	72.871	72.298	72.583	1.061
br-g15	83.061	81.075	82.055	1.124	br-k17	79.226	78.269	78.744	1.087
br-h01	78.388	78.244	78.315	1.082	br-k18	80.534	80.075	80.303	1.158
br-j01	79.92	79.92	79.92	1.053	br-k19	78.64	78.488	78.563	1.104
br-j02	73.555	73.043	73.298	1.091	**Media**	**77.461**	**76.516**	76.984	**1.089**
br-j03	78.624	78.188	78.405	1.096	**F(%)**			**76.986**	
br-j04	80.943	80.337	80.638	1.064	**MIN**	60.240	59.382	59.807	1.049
br-j05	74.674	73.713	74.19	1.063	**MAX**	85.214	84.707	84.925	1.158
br-j06	69.823	69.823	69.823	1.121					
br-j07	70.895	70.11	70.5	1.057					
br-j08	81.076	79.965	80.516	1.086					
br-j09	83.076	83.076	83.076	1.08					
br-j10	78.246	77.333	77.786	1.062					
br-j11	77.514	77.058	77.285	1.088					
br-j12	79.907	77.404	78.635	1.092					
br-j13	78.142	77.158	77.646	1.122					
br-j14	80.454	79.55	79.999	1.072					
br-j15	81.934	81.785	81.859	1.049					
br-j16	83.599	82.844	83.219	1.109					
br-j17	85.214	84.393	84.801	1.066					
br-j18	77.669	76.738	77.2	1.111					
br-j19	82.051	81.012	81.528	1.067					
br-j20	60.24	59.382	59.807	1.101					
br-j22	64.102	62.5	63.29	1.051					

Tabela C.6: Rezultatele algoritmului SynWSD pe corpusul SemCor2.0 în limba română (măsura de atracţie semantică este `dice`). Inventarul de sensuri este dat de domeniile IRST iar evaluarea este strictă.

Fișier	P(%)	R(%)	F(%)	S/C	Fișier	P(%)	R(%)	F(%)	S/C
br-a01	72.1	27.154	39.45	1.156	br-j23	69.876	27.395	39.359	1.16
br-a02	71.071	25.911	37.976	1.121	br-j37	63.963	22.468	33.254	1.153
br-a11	70.695	26.33	38.369	1.183	br-j52	61.403	16.666	26.216	1.157
br-a12	66.312	22.888	34.03	1.205	br-j53	67.796	25.723	37.295	1.166
br-a13	68.613	26.367	38.094	1.204	br-j54	72.576	27.066	39.427	1.174
br-a14	65.467	24.202	35.339	1.226	br-j55	74.061	25.619	38.069	1.16
br-a15	67.247	25.63	37.114	1.226	br-j56	67.697	29.228	40.828	1.134
br-b13	67.663	26.073	37.641	1.111	br-j57	68.975	27.272	39.088	1.168
br-b20	66.666	27.283	38.719	1.139	br-j58	70.155	22.998	34.64	1.166
br-c01	75.644	30.697	43.671	1.189	br-j59	70.512	25.028	36.943	1.166
br-c02	69.627	26.157	38.027	1.189	br-j60	73.965	31.568	44.25	1.097
br-c04	63.186	27.912	38.719	1.31	br-j70	79.559	37.102	50.604	1.056
br-d01	62.781	19.373	29.609	1.112	br-k01	70.253	24.395	36.214	1.227
br-d02	62.113	27.139	37.773	1.172	br-k02	69.256	22.601	34.08	1.212
br-d03	77.014	35.403	48.507	1.161	br-k03	66.095	23.915	35.121	1.171
br-d04	66.551	22.921	34.098	1.117	br-k04	64.596	24.27	35.283	1.164
br-e01	68.535	24.746	36.362	1.236	br-k05	67.68	21.601	32.749	1.262
br-e02	68.867	22.86	34.325	1.157	br-k06	72.471	28.988	41.411	1.165
br-e04	64.722	24.526	35.572	1.125	br-k07	65.942	23.184	34.306	1.3
br-e21	63.43	19.029	29.275	1.116	br-k08	64.21	22.592	33.423	1.277
br-e24	63.366	19.452	29.766	1.155	br-k09	69.961	22.942	34.553	1.197
br-e29	65.714	27.38	38.654	1.142	br-k10	66.776	23.495	34.759	1.141
br-f03	66.666	24.093	35.394	1.162	br-k11	73.333	24.217	36.41	1.164
br-f10	74.011	29.738	42.428	1.231	br-k12	69.602	26.805	38.704	1.15
br-f19	70.466	29.09	41.179	1.178	br-k13	75.347	25.833	38.474	1.201
br-f43	69.459	27.34	39.236	1.167	br-k14	69.314	22.403	33.861	1.155
br-g01	70.212	29.139	41.185	1.292	br-k15	67.1	25.121	36.556	1.179
br-g11	63.947	25.233	36.186	1.234	br-k16	76.363	32.026	45.126	1.189
br-g15	69.946	28.068	40.06	1.196	br-k17	68.928	24.935	36.621	1.2
br-h01	70.37	23.404	35.125	1.037	br-k18	66.885	24.938	36.33	1.324
br-j01	70.588	27.617	39.701	1.076	br-k19	63.44	21.096	31.662	1.189
br-j02	76.736	22.643	34.967	1.121	**Media**	**69.773**	**26.638**	38.403	**1.163**
br-j03	70.588	23.103	34.812	1.034	**F(%)**			**38.556**	
br-j04	68.888	28.353	40.171	1.098	**MIN**	54.393	16.666	25.665	1.034
br-j05	76.562	30.984	44.115	1.1	**MAX**	82.783	43.449	56.505	1.324
br-j06	76.176	27.641	40.563	1.135					
br-j07	72.307	28.804	41.196	1.1					
br-j08	72.274	30.684	43.078	1.137					
br-j09	79.104	38.605	51.887	1.1					
br-j10	76	31.799	44.837	1.13					
br-j11	73.469	28.093	40.644	1.131					
br-j12	77.753	37.934	50.99	1.107					
br-j13	57.966	18.486	28.032	1.132					
br-j14	82.389	42.997	56.505	1.075					
br-j15	82.783	41.658	55.425	1.133					
br-j16	79.345	43.449	56.15	1.173					
br-j17	81.419	36.792	50.681	1.062					
br-j18	75.641	31.175	44.152	1.076					
br-j19	62.781	19.716	30.008	1.184					
br-j20	54.393	16.795	25.665	1.184					
br-j22	63.636	17.948	27.999	1.181					

Tabela C.7: Rezultatele algoritmului SynWSD pe corpusul SemCor2.0 în limba engleză (combinator `int`). Inventarul de sensuri este dat de ILI iar evaluarea este relaxată.

Fișier	P(%)	R(%)	F(%)	S/C	Fișier	P(%)	R(%)	F(%)	S/C
br-a01	62.13	20.388	30.701	1.414	br-j23	51.19	16.895	25.405	1.339
br-a02	61.146	22.273	32.652	1.471	br-j37	62.735	23.251	33.927	1.292
br-a11	67.901	32.738	44.176	1.802	br-j52	38.62	11.405	17.609	1.324
br-a12	76.271	31.542	44.628	1.389	br-j53	47.096	13.747	21.281	1.38
br-a13	61.157	22.699	33.109	1.487	br-j54	63.829	23.255	34.089	1.228
br-a14	59.171	25.316	35.46	1.804	br-j55	66.285	22.48	33.573	1.348
br-a15	59.763	25.964	36.2	1.485	br-j56	57.591	26.57	36.363	1.371
br-b13	63.297	22.452	33.146	1.303	br-j57	56.647	20.331	29.922	1.283
br-b20	58.72	22.494	32.527	1.43	br-j58	57.668	19.665	29.328	1.435
br-c01	67.171	30.504	41.955	1.449	br-j59	60.666	19.077	29.026	1.56
br-c02	61.235	22.109	32.488	1.376	br-j60	63.761	25.09	36.01	1.275
br-c04	58.064	24.545	34.504	1.543	br-j70	68.888	28.703	40.522	1.288
br-d01	49.285	13.193	20.814	1.221	br-k01	52.197	16.903	25.536	1.571
br-d02	60.795	24.208	34.627	1.579	br-k02	61.578	20.598	30.869	1.5
br-d03	63.673	32.842	43.333	1.408	br-k03	52.229	16.599	25.191	1.222
br-d04	54.374	17.296	26.243	1.187	br-k04	62.189	24.95	35.612	1.626
br-e01	57.763	18.712	28.267	1.403	br-k05	54.32	16.89	25.767	1.425
br-e02	63.131	22.482	33.156	1.313	br-k06	62.433	21.892	32.417	1.46
br-e04	59.195	21.914	31.986	1.413	br-k07	58.823	17.582	27.072	1.477
br-e21	53.107	15.46	23.948	1.225	br-k08	60.795	20	30.098	1.414
br-e24	58.115	18.5	28.065	1.465	br-k09	52.976	17.348	26.136	1.488
br-e29	64.114	25.868	36.862	1.191	br-k10	59.685	20.284	30.277	1.554
br-f03	63.749	18.784	29.017	1.325	br-k11	66.666	20.458	31.308	1.373
br-f10	55.721	22.178	31.727	1.283	br-k12	59.887	19.343	29.241	1.378
br-f19	65.437	26.944	38.17	1.239	br-k13	63.551	23.247	34.041	1.434
br-f43	55.214	15.789	24.555	1.239	br-k14	65.363	21.081	31.88	1.351
br-g01	50.555	19.198	27.828	1.35	br-k15	63.428	21.102	31.668	1.314
br-g11	67.934	21.477	32.636	1.228	br-k16	61.722	25.343	35.932	1.449
br-g15	56.451	20.916	30.522	1.338	br-k17	60	21.73	31.905	1.388
br-h01	48.356	18.829	27.104	1.338	br-k18	53.926	19.544	28.69	1.382
br-j01	68.2	32.405	43.934	1.255	br-k19	56.172	17.635	26.842	1.345
br-j02	73.705	32.173	44.793	1.247	**Media**	**59.845**	**22.214**	32.254	**1.373**
br-j03	49.753	18.669	27.15	1.172	**F(%)**			**32.401**	
br-j04	58.851	23.033	33.108	1.354	**MIN**	34.848	10.926	16.636	1.120
br-j05	55.502	21.323	30.809	1.511	**MAX**	76.271	34.061	46.690	1.804
br-j06	55.319	17.18	26.217	1.12					
br-j07	55.072	21.033	30.44	1.381					
br-j08	62.272	23.458	34.078	1.231					
br-j09	69.819	29.807	41.778	1.234					
br-j10	68.84	31.666	43.378	1.38					
br-j11	59.907	25.49	35.763	1.419					
br-j12	59.693	26.174	36.391	1.561					
br-j13	49.693	14.568	22.53	1.472					
br-j14	66.834	29.887	41.303	1.19					
br-j15	74.206	34.061	46.69	1.361					
br-j16	69.014	33.182	44.816	1.206					
br-j17	61.835	24.662	35.26	1.222					
br-j18	63.157	25.899	36.734	1.304					
br-j19	52.147	15.37	23.742	1.368					
br-j20	34.848	10.926	16.636	1.484					
br-j22	72.727	20	31.372	1.181					

Tabela C.8: Rezultatele algoritmului SynWSD pe corpusul SemCor2.0 în limba română (combinator `int`). Inventarul de sensuri este dat de ILI iar evaluarea este relaxată.

Bibliografie

[1] Eneko Agirre and German Rigau. Word sense disambiguation using conceptual density. In *Proceedings of COLING'96*, pages 16–22, Copenhagen, Danmark, 1996.

[2] Susan Armstrong. Multext: Multilingual Text Tools and Corpora. *Lexikon und Text*, pages 107–119, 1996.

[3] Verginica Barbu Mititelu and Radu Ion. Automatic import of verbal syntactic relations using parallel corpora. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP-2005*, pages 329–333, Borovets, Bulgaria, September 2005.

[4] Verginica Barbu Mititelu and Radu Ion. Cross-language transfer of syntactic relations using parallel corpora. In *Proceedings of Cross-Language Knowledge Induction Workshop, EuroLan 2005*, Cluj-Napoca, Romania, July–August 2005. Babeş-Bolyai University.

[5] Doug Beeferman, Adam Berger, and John Lafferty. A model of lexical attraction and repulsion. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 373–380, Somerset, New Jersey, 1997. Association for Computational Linguistics.

[6] Andrew Borthwick. *A Maximum Entropy Approach to Named Entity Recognition*. PhD thesis, Computer Science Department, New York University, September 1999.

[7] Thorsten Brants. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference, ANLP-2000*, pages 224–231, Seattle, WA, April 29 – May 3 2000.

[8] Brill, Eric. A simple rule-based part-of-speech tagger. In *Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*, pages 152–155, Trento, Italy, April 1992.

[9] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. Word-sense disambiguation using statistical methods. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL-91)*, pages 264–270, Berkeley, CA., June 1991.

[10] Peter F. Brown, Jennifer C. Lai, and Robert L. Mercer. Aligning sentences in parallel corpora. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pages 169–176, Berkeley, California, June 1991. Association for Computational Linguistics.

[11] Rebecca Bruce and Janyce Wiebe. Word sense disambiguation using decomposable models. In *Proceedings of the ACL-94, 32nd Annual Meeting of the Association for Computational Linguistics*, pages 139–145, Las Cruces, US, 1994.

[12] Alexander Budanitsky and Graeme Hirst. Semantic distance in WordNet: An experimental, application oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources, NAACL 2001*, pages 29–34, Pittsburgh, PA, USA, 2001.

[13] Andrew Carnie. *Syntax*. Blackwell Publishers, Oxford, December 2001.

[14] Alexandru Ceauşu. Maximum Entropy Tiered Tagging. In Janneke Huitink and Sophia Katrenko, editors, *Proceedings of the Eleventh ESSLLI Student Session*, pages 173–179, Malaga, Spain, August 2006.

[15] Dumitru Chiţoran and Alexandra Cornilescu. *Elements of English Sentence Semantics*. The University of Bucharest Press, 1985. vol. I.

[16] Silviu Cucerzan and David Yarowsky. Language independent named entity recognition combining morphological and contextual evidence. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in NLP and Very Large Corpora*, pages 90–99, College Park, MD, 1999.

[17] Ito Dagan, Alon Itai, and Ulrike Schwall. Two Languages Are More Informative Than One. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL-91)*, pages 130–137, Berkeley, CA., June 1991.

[18] Institutul de Lingvistică "Iorgu Iordan". *Dicţionarul explicativ al limbii române*. Editura Univers Enciclopedic, 1998. Academia Română.

[19] de Loupy, Claude and El-Beze, Marc and Marteau, Pierre-François. Word Sense Disambiguation using HMM Tagger. In *Proceedings of the First International Conference on Language Resources and Evaluation*, pages 1255–1258, Grenade, Spain, May 1998.

[20] Mona Talat Diab. *Word Sense Disambiguation Within a Multilingual Framework*. PhD thesis, University of Maryland, College Park, May 2003.

[21] Ludmila Dimitrova, Tomaž Erjavec, Nancy Ide, Heiki J. Kaalep, Cszaba Oravetz, Vladimir Petkevič, and Dan Tufiş. Multext-East: Overview of the project. In *Proceedings of the ALLC-ACH '98 Conference*, Debrecen, Hungary, July 5-10 1998.

[22] Jason Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen, August 1996.

[23] Tomaž Erjavec. MULTEXT-East Version 3: Multilingual Morphosyntactic Specifications, Lexicons and Corpora. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC '04*, Lisbon, Portugal, 2004.

[24] Christiane Fellbaum. A lexical database of English: The mother of all WordNets. In P. Vossen, editor, *EuroWordNet*, pages 137–148. Kluwer, Dordrecht, Holland, 1998.

[25] Christiane Fellbaum, editor. *WordNet. An Electronic Lexical Database*. MIT Press, May 1998.

[26] John Rupert Firth. *Studies In Linguistic Analisys*, chapter A Synopsis of Linguistic Theory, pages 1–32. Basil Blackwell, Oxford, 3rd edition, 1957.

[27] Corina Forăscu and Radu Ion. TimeBank 1.2: O versiune adnotată în limba română. In Corina Forăscu, Dan Tufiş, and Dan Cristea, editors, *Lucrările atelierului RESURSE LINGVISTICE ŞI INSTRUMENTE PENTRU PRELUCRAREA LIMBII ROMÂNE (ConsILR 2006)*, pages 69–74, Iaşi, Romania, noiembrie 2006.

[28] Nelson W. Francis and Henry Kučera. *Brown Corpus Manual*. Department of Linguistics, Brown University, Providence, Rhode Island, 1979. 1964, Revised 1971, Revised and Amplified 1979.

[29] William A. Gale, Kenneth W. Church, and David Yarowsky. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26(5):415–439, 1993.

[30] Ulrich Germann. A Deterministic Dependency Parser for Japanese. In *MT Summit VII: MT in the Great Translation Era*, pages 547–555, Singapore, November 1999. Asia-Pacific Association for Machine Translation.

[31] Gregory Grefenstette and Pasi Tapanainen. What is a word, what is a sentence? Problems of tokenization. In *Proceedings of the 3rd International Conference on Computational Lexicography*, pages 79–87, Budapest, 1994.

[32] Jan Hajič. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In E. Hajičová, editor, *Issues of Valency and Meaning. Studies in Honour of Jarmila Panevová*, pages 106–132. Karolinum, Charles University Press, Prague, Czech Republic, 1998.

[33] Marti A. Hearst. Noun homograph disambiguation using local context in large corpora. In *Proceedings of the 7th Annual Conference of the University of Waterloo Center for the New Oxford English Dictionary*, pages 1–22, Oxford, UK, 1991.

[34] Ales Horák and Pavel Smrž. VisDic - Wordnet Browsing and Editing Tool. In *Proceedings of the Second International WordNet Conference - GWC 2004*, pages 136–141, Brno, Czech Republic, 2004.

[35] Florentina Hristea and Marius Popescu. A dependency grammar approach to syntactic analysis with special reference to Romanian. In Florentina Hristea and Marius Popescu, editors, *Building Awareness in Language Technology*. University of Bucharest Publishing House, Bucharest, Romania, 2003.

[36] James W. Hunt and Thomas G. Szymanski. A fast algorithm for computing longest common subsequences. *Communications of the Association for Computing Machinery*, 20(5):350–353, May 1977.

[37] Nancy Ide and Jean Véronis. Introduction to the special issue on word sense disambiguation: The state of the art. *Computational Linguistics*, 24(1):1–40, 1998.

[38] Radu Ion. Word sense disambiguation with lexical attraction models. Poster at First Central European Student Conference in Linguistics, CESCL 2006, Budapest, Hungary, May 29–31 2006. Research Institute for Linguistics of the Hungarian Academy of Sciences.

[39] Radu Ion and Verginica Barbu Mititelu. Towards ROMANCE FrameNet. The Translation Task. ROMANCE FrameNet Workshop and Kick-off Meeting, EuroLan 2005, Cluj-Napoca, România, July 25–August 6 2005. Babeş-Bolyai University.

[40] Radu Ion and Verginica Barbu Mititelu. Constrained lexical attraction models. In *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference*, pages 297–302, Menlo Park, Calif., USA, 2006. AAAI Press.

[41] Radu Ion, Alexandru Ceauşu, and Dan Tufiş. Dependency-based phrase alignment. In *Proceedings of the 5th Language and Resources Evaluation Conference (LREC 2006)*, pages 1290–1293, Genoa, Italy, May 22–28 2006.

[42] Radu Ion and Dan Tufiş. Multilingual Word Sense Disambiguation Using Aligned Wordnets. *Romanian Journal on Information Science and Technology, Special Issue on BalkaNet*, 7(1–2):198–214, 2004.

[43] Emil Ionescu. *Manual de lingvistică generală*. Editura ALL, Bucureşti, România, 1992.

[44] Timo Järvinen and Pasi Tapanainen. A dependency parser for English. Technical Reports TR-1, Department of General Linguistics, University of Helsinki, March 1997.

[45] Timo Järvinen and Pasi Tapanainen. Towards an implementable dependency grammar. In S. Kahane and A. Polguere, editors, *Processing of Dependency-Based Grammars, COLING-ACL'98*, pages 1–10, Montreal, Canada, 1998. Association for Computational Linguistics.

[46] Stephen C. Johnson. Hierarchical Clustering Schemes. *Psychometrika*, 32(3):241–254, 1967.

[47] Lauri Karttunen, Jean-Pierre Chanod, Gregory Grefenstette, and Anne Schiller. Regular expressions for language engineering. *Natural Language Engineering*, 2(4):305–238, 1996.

[48] Adam Kilgarriff. What is Word Sense Disambiguation Good For? In *Natural Language Processing in the Pacific Rim (NLPRS '97)*, pages 209–214, Phuket, Thailand, December 1997.

[49] Adam Kilgarriff and Joseph Rosenzweig. English Senseval: report and results. In *In Proceedings of the 2nd International Conference on Language Resources and Evaluation, LREC 2000*, pages 1239–1244, Athens, Greece, May–June 2000.

[50] Henry Kučera and Nelson W. Francis. *Computational analysis of present-day American English.* Brown University Press, Providence, Rhode Island, 1967.

[51] Yoong Keok Lee, Hwee Tou Ng, and Tee Kiah Chia. Supervised word sense disambiguation with support vector machines and multiple knowledge sources. In *Proceedings of SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 137–140, Barcelona, Spain, 2004.

[52] Michael Lesk. Automatic sense disambiguation : How to tell a pine cone from an ice cream cone. In *Proceedings of the 1986 SIGDOC Conference, Association for Computing Machinery*, pages 24–26, New York, 1986.

[53] Dekang Lin. Using syntactic dependency as local context to resolve word sense ambiguity. In *Meeting of the Association for Computational Linguistics*, pages 64–71, 1997.

[54] Dekang Lin. Dependency-Based Evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation*, Granada, Spain, May 1998.

[55] Monica Lupu, Diana Trandabăţ, and Maria Husarciuc. A Romanian SemCor Aligned to the English and Italian MultiSemCor. In *Proceedings of the Romance FrameNet Workshop and Kick-off Meeting, EuroLAN 2005*, pages 20–27, Babes-Bolyai University, Cluj-Napoca, România, July 2005.

[56] Bernardo Magnini and Gabriela Cavaglia. Integrating Subject Field Codes into WordNet. In Gavrilidou M., Crayannis G., Markantonatu S., Piperidis S., and Stainhaouer G., editors, *Proceedings of LREC-2000, Second International Conference on Language Resources and Evaluation*, pages 1413–1418, Athens, Greece, June 2000.

[57] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1st edition, June 1999.

[58] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1994.

[59] Joel Martin, Rada Mihalcea, and Ted Pedersen. Word Alignment for Languages with Scarce Resources. In *In Proceedings of the ACL2005 Workshop on "Building and Using Parallel Corpora: Datadriven Machine Translation and Beyond"*, pages 65–74, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[60] Ana Masalagiu. Realizarea de resurse româneşti. Master's thesis, Universitatea Alexandru Ioan Cuza, Facultatea de Informatică, Iaşi, România, iunie 2006.

[61] Igor Mel'čuk. *Dependency Syntax: theory and practice*. State University of New York Press, Albany, NY, 1988.

[62] Rada Mihalcea. Instance based learning with automatic feature selection applied to word sense disambiguation. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, pages 266–271, Taiwan, August 2002.

[63] Rada Mihalcea and Dan Moldovan. A method for word sense disambiguation of unrestricted text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 1999)*, College Park, MA, 1999.

[64] Rada Mihalcea and Ted Pedersen. An Evaluation Exercise for Word Alignment. In *Proceedings of the HLT-NAACL 2003 Workshop: Building and Using Parallel Texts Data Driven Machine Translation and Beyond*, pages 1–10, Edmonton, Canada, May 2003.

[65] Rada F. Mihalcea and Dan I. Moldovan. A highly accurate bootstrapping algorithm for word sense disambiguation. *International Journal on Artificial Intelligence Tools*, 10(1–2), 2001.

[66] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Introduction to WordNet: An online lexical database. *International Journal of Lexicography (special issue)*, 3(4):235–312, 1990. Revised August 1993.

[67] George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, pages 303–308, Plainsboro, New Jersey, 1993.

[68] Mladenić, Dunja. Automatic Word Lemmatization. In Tomaž Erjavec and Jerneja Gros, editors, *Proceedings B of the 5th International Multi-Conference Information Society IS-2002*, pages 153–159, Ljubljana, Slovenia, October 14-15 2002.

[69] Richard Montague. The proper treatment of quantification in ordinary English. In Richard Thomason, editor, *Formal Philosphy: Selected Papers of Richard Montague*. Yale University Press, New Haven, CT, 1973.

[70] Robert C. Moore. Fast and Accurate Sentence Alignment of Bilingual Corpora. In *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas on Machine Translation: From Research to Real Users*, pages 135–144, London, UK, 2002. Springer-Verlag.

[71] Robert C. Moore. On Log-Likelihood Ratios and the Significance of Rare Events. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 333–340, Barcelona, Spain, 2004.

[72] Eugene W. Myers. An O(ND) Difference Algorithm and its Variations. *Algorithmica*, 1(2):251–266, 1986.

[73] Hwee Tou Ng and Hian Beng Lee. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In Arivind Joshi and Martha Palmer, editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 40–47, San Francisco, 1996. Morgan Kaufmann Publishers.

[74] Ng, Hwee Tou. Getting serious about word sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, pages 1–7, Washington, D.C., USA, 1997.

[75] Ian Niles and Adam Pease. Towards a Standard Upper Ontology. In Chris Welty and Barry Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, Ogunquit, Maine, October 2001.

[76] Ian Niles and Adam Pease. Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. In *Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE 03)*, Las Vegas, Nevada, June 2003.

[77] Joakim Nivre. An efficient algorithm for projective dependency parsing. In Gertjan van Noord, editor, *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160, 2003.

[78] Joakim Nivre. *Inductive Dependency Parsing*, volume 34 of *Text, Speech and Language Technology*. Springer, Dordrecht, The Netherlands, 2006.

[79] Joakim Nivre and Jens Nilsson. Three algorithms for deterministic dependency parsing. In *Proceedings of NoDaLiDa-2003*, 2003.

[80] Kemal Oflazer. Dependency parsing with an extended finite-state approach. *Computational Linguistics*, 29(4):515–544, 2003.

[81] Praharshana Perera and René Witte. A Self-Learning Context-Aware Lemmatizer for German. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, pages 636–643, Vancouver, British Columbia, Canada, October 6–8 2005. Association for Computational Linguistics.

[82] Thanh Phong Pham, Hwee Tou Ng, and Wee Sun Lee. Word sense disambiguation with semi-supervised learning. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 1093–1098, Pittsburgh, Pennsylvania, USA, 2005.

[83] Felix Pîrvan and Dan Tufiş. Tagsets Mapping and Statistical Training Data Cleaning-up. In *Proceedings of the 5th LREC Conference*, Genoa, Italy, 22–28 May 2006.

[84] Plisson, Jöel and Lavrač, Nada and Mladenić, Dunja. A Rule Based Approach to Word Lemmatization. In *Proceedings of SiKDD 2004 at 7th International Multi-conference Information Society, IS-2004*, pages 83–86, Ljubljana, Slovenia, October 12-15 2004.

[85] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[86] Georgiana Puşcaşu, Adrian Iftene, Ionuţ Pistol, Diana Trandabăţ, Dan Tufiş, Alin Ceauşu, Dan Ştefănescu, Radu Ion, Constantin Orăşan, Iustin Dornescu, Alex Moruz, and Dan Cristea. Developing a Question Answering System for the Romanian-English Track at CLEF 2006. In *Proceedings of the 7th Workshop of the Cross-Language Evaluation Forum, CLEF2006*, page 10, Alicante, Spain, September 20–22 2006. To be publised in Springer Lecture Notes in Computer Science.

[87] Lawrence L. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[88] Adwait Ratnaparkhi. A Maximum Entropy Model for Part-of-Speech Tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics, Somerset, New Jersey, 1996.

[89] Jeffrey C. Reynar and Adwait Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, Washington D.C., 1997.

[90] Giuseppe Riccardi, Srinivas Bangalore, and Philip D. Sarin. Learning Head-Dependency Relations from Unannotated Corpora. In *Proceedings IEEE Automatic Speech Recognition and Understanding Workshop*, pages 281–284, Keystone, Colorado, USA, December 1999.

[91] Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. Multiword Expressions: A Pain in the Neck for NLP. In Alexander Gelbukh, editor, *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics (CICLING 2002)*, pages 1–15, Mexico City, Mexico, 2002. Springer.

[92] Helmut Schmid. Part-Of-Speech Tagging with Neural Networks. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, pages 172–176, Kyoto, Japan, 1994.

[93] Hinrich Schütze. Word Space. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Sys-*

tems, pages 895–902. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[94] Hinrich Schütze. Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97–124, 1998.

[95] John Sinclair, editor. *Collins Cobuild English Dictionary*. Collins, 1995. Patrick Hanks, managing editor.

[96] Benjamin Snyder and Martha Palmer. The English all-words task. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[97] Jiri Stetina, Sadao Kurohashi, and Makoto Nagao. General word sense disambiguation method based on a full sentential context. In *Proceedings of the Coling-ACL'98 Workshop "Usage of WordNet in Natural Language Processing Systems"*, pages 1–8, Montreal, 1998.

[98] Mark Stevenson and Yorick Wilks. The interaction of knowledge sources in word sense disambiguation. *Computational Linguistics*, 27(3):321–349, 2001.

[99] Pasi Tapanainen and Timo Järvinen. A non-projective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71, Washington D.C., USA, April 1997. Association for Computational Linguistics.

[100] Dan Tufiş. Tiered Tagging and Combined Classifiers. In F. Jelinek and E. Nöth, editors, *Lecture Notes in Artificial Intelligence 1692*, Text, Speech and Dialogue, pages 28–33. Springer, 1999.

[101] Dan Tufiş. Using a Large Set of Eagles-compliant Morpho-Syntactic Descriptors as a Tagset for Probabilistic Tagging. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, pages 1105–1112, Athens, May 2000.

[102] Dan Tufiş. A cheap and fast way to build useful translation lexicons. In *Proceedings of COLING2002*, pages 1030–1036, Taipei, China, 2002.

[103] Dan Tufiş, Ana Maria Barbu, and Radu Ion. TREQ-AL: A word-alignment system with limited language resources. In *Proceedings of the NAACL 2003 Workshop on Building and Using Parallel Texts;*

Romanian-English Shared Task, pages 36–39, Edmonton, Canada, December 2003.

[104] Dan Tufiş, Ana Maria Barbu, and Radu Ion. Extracting Multilingual Lexicons from Parallel Corpora. *Computers and the Humanities*, 38(2):163–189, 2004. ISI publication.

[105] Dan Tufiş, Eduard Barbu, Verginica Barbu Mititelu, Radu Ion, and Luigi Bozianu. The Romanian Wordnet. *Romanian Journal on Information Science and Technology, Special Issue on BalkaNet*, 7(1–2):105–122, 2004.

[106] Dan Tufiş, Verginica Barbu Mititelu, Luigi Bozianu, and Cătălin Mihăilă. Romanian WordNet: New Developments and Applications. In *Proceedings of the 3rd Conference of the Global WordNet Association*, pages 337–344, Seogwipo, Jeju, Republic of Korea, January 2006.

[107] Dan Tufiş, Alexandru Ceauşu, Radu Ion, and Dan Ştefănescu. An integrated platform for high-accuracy word alignment. JRC Enlargement and Integration Workshop: Exploiting parallel corpora in up to 20 languages, Arona, Italy, September 26–27 2005.

[108] Dan Tufiş, Dan Cristea, and Sofia Stamou. BalkaNet: Aims, Methods, Results and Perspectives. A General Overview. *Romanian Journal on Information Science and Technology, Special Issue on BalkaNet*, 7(1–2):9–43, 2004.

[109] Dan Tufiş, Peter Dienes, Cszaba Oravecz, and Tamas Váradi. Principled Hidden Tagset Design for Tiered Tagging of Hungarian. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, pages 1421–1426, Athens, May 2000.

[110] Dan Tufiş and Liviu Dragomirescu. Tiered Tagging Revisited. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, pages 39–42, Lisabona, 2004.

[111] Dan Tufiş and Radu Ion. Word Sense Clustering Based on Translation Equivalence in Parallel Texts. A Case Study in Romanian. In Corneliu Burileanu, editor, *Speech Technology and Human-Computer Dialogue (SpeD 2003)*, pages 13–26, Bucharest, April 2003. Romanian Academy Publishing House.

[112] Dan Tufiş and Radu Ion. Interlingual wordnets validation and word-sense disambiguation. In *Proceedings of the Natural Language Understanding and Cognitive Systems Symposium*, pages 97–105, Porto, April 2004.

[113] Dan Tufiş and Radu Ion. Evaluating the word sense disambiguation accuracy with three different sense inventories. In Bernadette Sharp, editor, *Proceedings of the 2nd International Workshop on Natural Language Understanding and Cognitive Science, NLUCS 2005*, pages 118–127, Miami, U.S.A., May 2005.

[114] Dan Tufiş and Radu Ion. Multiple Sense Inventories and Test-Bed Corpora. In Corneliu Burileanu, editor, *Trends in Speech Technology, Proceedings of the 3rd Conference Speech Technology and Human-Computer Dialogue (SpeD 2005)*, pages 49–58, Cluj-Napoca, Romania, May 2005. Romanian Academy Publishing House.

[115] Dan Tufiş, Radu Ion, Eduard Barbu, and Verginica Barbu. Cross-Lingual Validation of Multilingual Wordnets. In Petr Sojka, Karel Pala, Pavel Smrz, Christine Fellbaum, and Piek Vossen, editors, *Proceedings of the Second International WordNet Conference – GWC 2004*, pages 332–340, Brno, Czech Republic, January 2004.

[116] Dan Tufiş, Radu Ion, and Verginica Barbu Mititelu. Word sense disambiguation and annotation transfer in parallel text. JRC Enlargement and Integration Workshop: Exploiting parallel corpora in up to 20 languages, Arona, Italy, September 26–27 2005.

[117] Dan Tufiş, Radu Ion, Alexandru Ceauşu, and Dan Ştefănescu. Combined aligners. In *Proceedings of the ACL 2005 Workshop on Building and Using Parallel Corpora: Data-driven Machine Translation and Beyond*, pages 107–110, Ann Arbor, Michigan, USA, June 2005. Association for Computational Linguistics.

[118] Dan Tufiş, Radu Ion, Alexandru Ceauşu, and Dan Ştefănescu. Improved Lexical Alignment by Combining Multiple Reified Alignments. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, pages 153–160, Trento, Italy, April 2006. ISI publication.

[119] Dan Tufiş, Radu Ion, and Nancy Ide. Fine-Grained Word Sense Disambiguation Based on Parallel Corpora, Word Alignment, Word Clustering and Aligned Wordnets. In *Proceedings of the 20th International*

Conference on Computational Linguistics, COLING 2004, pages 1312–1318, Geneva, Switzerland, August 2004. COLING.

[120] Dan Tufiş, Radu Ion, and Nancy Ide. Word Sense Disambiguation as a Wordnets Validation Method in Balkanet. In *Proceedings of the 4th Language and Resources Evaluation Conference (LREC 2004)*, pages 741–744; 1071–1074, Lisbon, Portugal, May 2004. plenary talk, plus demonstration.

[121] Dan Tufiş and Elena Irimia. RoCoNews - A Hand Validated Journalistic Corpus of Romanian. In *Proceedings of the 5th LREC Conference*, Genoa, Italy, 22–28 May 2006.

[122] Pascal Vaillant. A chart-parsing algorithm for efficient semantic analysis. *The Association for Computational Linguistics and Chinese Language Processing*, 2:1044–1050, 2002.

[123] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT(13):260–269, April 1967.

[124] Piek Vossen, editor. *EuroWordNet: A multilingual database with lexical semantic networks*, volume 32 of *Computers and Humanities*. Springer Netherlands, 1998. nos. 2–3.

[125] Larry Wall, Tom Christiansen, and Jon Orwant. *Programming Perl*. O'Reilly Media, 3rd edition, July 2000.

[126] Ludwig Wittgenstein. *Cercetări filozofice*. Editura Humanitas, 2004. Traducere din germană de Mircea Dumitru şi Mircea Flonta.

[127] David Yarowsky. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings, COLING-92*, pages 454–460, Nantes, 1992.

[128] David Yarowsky. One sense per collocation. In *ARPA Human Language Technology Workshop*, pages 266–271, Princeton, NJ, 1993.

[129] David Yarowsky. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 88–95, Las Cruces, NM, 1994.

[130] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA, 1995.

[131] Deniz Yuret. *Discovery of linguistic relations using lexical attraction.* PhD thesis, Department of Computer Science and Electrical Engineering, MIT, May 1998.

Index