

UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE AUTOMATICA ȘI CALCULATOARE
SPECIALIZAREA ADMINISTRAREA BAZELOR DE DATE



PROIECT DE DIZERTAȚIE

DEZAMBIGUIZAREA
INTEROGĂRILOR ÎN PAGINI WEB

Coordonator științific:
Conf. Dr. Ing. Alexandru Boicea

Absolvent:
Soltaniuc Vlad

BUCUREȘTI

2020

Cuprins

1. Natural Language Processing și Wordnet	4
1.1. Introducere.....	4
1.2. Wordnet.....	4
1.2.1. Matrice Lexicală.....	5
1.2.2. Sense tagging.....	6
1.2.3. Clasificarea cuvintelor.....	7
1.2.4. Principii de baza ale Wordnet-ului:.....	7
2. Word Sense Desambiguation (WSD).....	9
2.1. Knowledge Based VS Machine Learning Based VS Hybrid Approaches.....	10
2.1.1. Knowledge Based Approaches.....	10
2.1.1.1. WSD using selectional preferences and arguments	10
2.1.1.2. Selețional preferences (Teoria lingvistică recentă)	11
2.1.1.2.1. Critică	12
2.1.1.3. Overlap based approaches.....	12
2.1.1.4. Algoritmul lui Lesk.....	12
2.1.1.4.1. Critică	13
2.1.1.5. Algoritmul lui Lesk extins	13
2.1.1.5.1. Critică	14
2.1.1.6. Algoritmul lui Walker	14
2.1.1.7. WSD folosind Conceptual Density (Agirre and Rigau, 1996).....	15
2.1.1.7.1. Critică	17
2.1.1.8. WSD folosind algoritmul Random Walk (Page Rank) (sinha and Mihalcea, 2007)	17
2.1.1.8.1. Un overview Page Rank	19
2.1.1.8.2. Aplicarea Page Rank pentru WSD	19
2.1.1.8.3. Alți algoritmi link based	20
2.1.1.8.4. Critică	20
2.1.1.9. Compararea abordărilor Knowledge based.....	20
2.1.1.9.1. Concluzia abordărilor Knowledge based.....	20
2.1.2. Abordări Supervizate.....	21
2.1.2.1. Naive Bayes.....	21
2.1.2.2. Decision List Algorithm.....	23
2.1.2.2.1. Critică	24
2.1.2.3. Exemplar Based WSD (k-nn)	24
2.1.2.4. WSD folosind SVMs	25

2.1.2.5.	WSD folosind Perceptron Trained HMM (Hidden Markov Model)	25
2.1.2.5.1.	Compararea abordărilor Supervizate	25
2.1.3.	Abordări Semi-Supervizate	26
2.1.3.1.	Idea de bază a WSD Semi-Supervizat.....	26
2.1.3.2.	Decision List Algorithm Semi-Supervizat.....	26
2.1.3.2.1.	Compararea abordărilor Semi-Supervizate.....	26
2.1.4.	Abordări Nesupervizate.....	27
2.1.4.1.	Hyperlex (Veronis Jean. 2004. HyperLex: Lexical cartography for information retrieval. Computer Speech & Language, 18(3):223-252	27
2.1.4.1.1.	Detectarea ROOT HUBS	27
2.1.4.1.2.	Delimitarea componentelor	28
2.1.4.1.3.	Dezambiguizare.....	28
2.1.4.1.4.	Adaugarea greutăților pe muchii	29
2.1.4.1.5.	Critică	30
2.1.4.2.	Algoritmul lui Yarowsky (WSD folosind categoriile de Thesaurus a lui Rogert) 30	
2.1.4.2.1.	Dezambiguizare	30
2.1.4.2.2.	Critică	31
2.1.4.3.	Lin's Approach	31
2.1.4.3.1.	Similaritate și Hypernymy	32
2.1.4.4.	WSD folosind Parallel Corpora	33
2.1.4.5.	Compararea abordărilor Nesupervizate	33
2.1.4.6.	Abordări Nesupervizate – Concluzie	34
3.	Algoritm	35
3.1.	Interfața grafică	37
3.2.	Cum putem folosi resursa Wordnet? (exemple practice în Pycharm):.....	38
4.	Istoria pe scurt	40
5.	Concluzie.....	40

1. Natural Language Processing și Wordnet

(Articole folosite pentru învățarea și cercetarea limbajului natural și Wordnet [1] [6] [7])

1.1. Introducere

Word sense disambiguation este un subiect foarte important în Natural Language Processing, deoarece majoritatea cuvintelor din limbajul natural sunt ambiugue. Oamenii pot înțelege propoziții ce conțin asemenea cuvinte, indentificand sensul corect din context, dar pentru a automatiză acest proces avem nevoie de algoritmi. Creierul uman este destul de competent în privința dezambiguizării. Limbajul uman s-a dezvoltat reflectând capacități înnăscute oferite de rețelele neuronale ale creierului.

1.2. Wordnet

Wordnet este o baza de date ce conține cunoștințe lexicale, organizate în funcție de sensul cuvintelor. Spre deosebire de un dicționar, în urma unei căutări rezultatul va fi un „meaning unit”.

Teoria Psiholinguistică

Memoria lexicală umană este structurată ca o ierarhie. În urmă unui experiment psihologic s-a demonstrat că Psiholinguistica timpul de răspuns depinde de nivelul ierarhic al unui cuvânt.

Canarul poate cânta? – răspuns rapid

Canarul poate zbura? – răspuns încetinit

Canarii au piele? – cel mai mare timp de răspundere

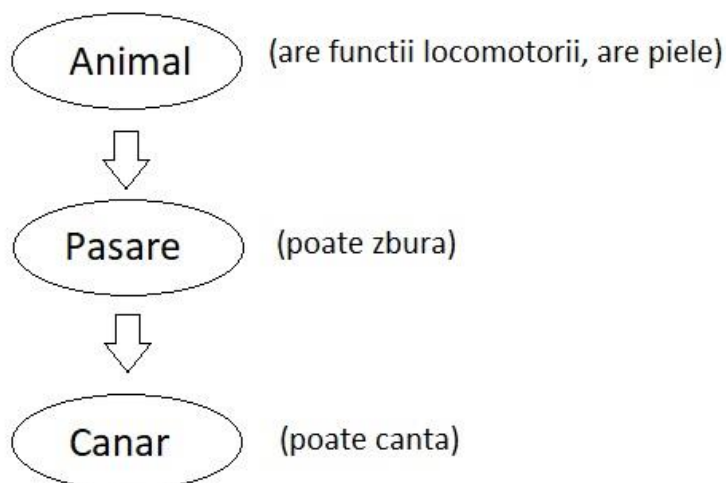



Fig. 1. Ierarhia cuvântului “canar”


În cazul în care vrem să aflăm dacă Canarul are funcții locomotorii, trebuie să avansăm în ierarhie până ajungem la „animal” unde sunt stocate informațiile necesare.

1.2.1. Matrice Lexicală

Sensul cuvintelor	Forma cuvintelor				
	Apple	Bank	Synonym	Equivalent
00001-Fruit	😊				
00002-Financial		😊			
00003-Aquatic		😊			
.....			😊	😊	
000099					



Polisemie



Sinonime

Tabel 1. Matrice Lexicală

Primul wordnet a fost creat pentru limba engleză, dezvoltat la Princeton pe parcursul a 15 ani. Eurowordnet, este o structura cu legături între diferite limbi europene, construită în 1998 pe parcursul a 3 ani, cu finanțare de la Comisia Europeană.

Un exemplu de Wordnet poate fi wordnet-ul Indian, unde mai multe wordneturi sunt interconectate pentru a formă IndoWordnet:

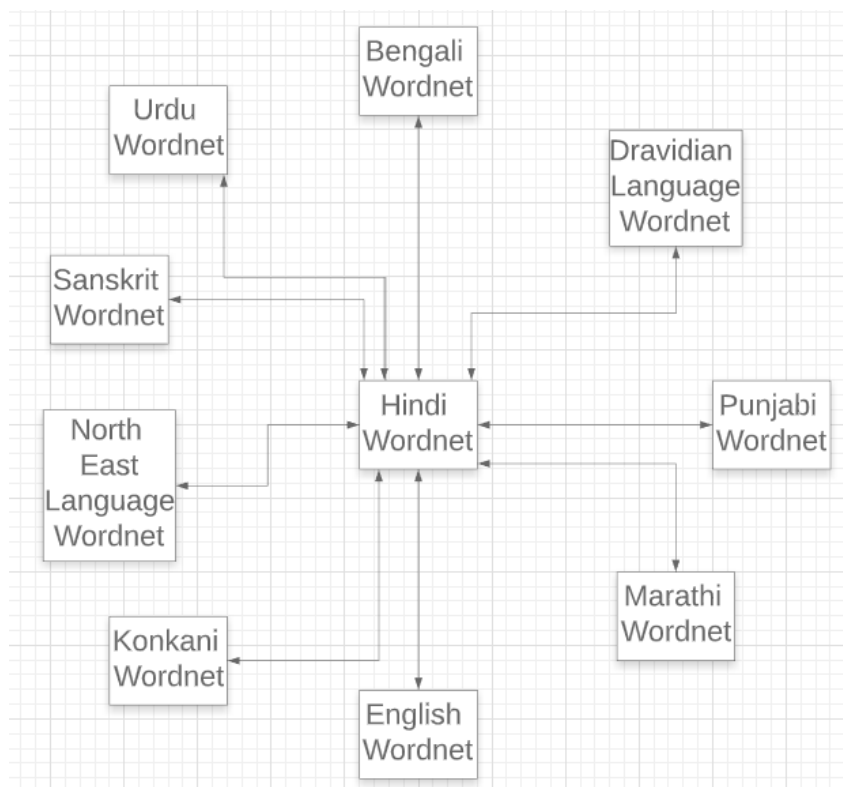


Fig. 2. Wordnet-urile Indiene interconectate

O întrebare fundamentală de design este: „Syntagmatic vs Paradigmatic relations?”

Cuvintele sunt legate sintagmatic dacă ele apar împreună în aceeași sintagma, unitate sintactică, de obicei o propoziție.

Ex: Un câine latră mult. (câine și latră apar în aceeași propoziție = sunt legate syntagmatically)

Cuvintele sunt legate paradigmatically dacă ele apar legate într-o resursă lexicală. Legătură dintre acestea se numește hypernymy (sau generalizare).

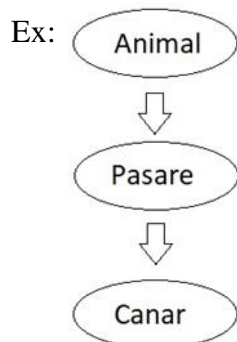


Fig. 1. Ierarhia cuvântului “canar”

Când auzim un cuvânt, ne vin în minte multe alte cuvinte, exemplu:

Pisica:

- animal, mamifer – paradigmatic
- mew, purr, blanoasa – syntagmatic

Într-un wordnet putem găsi albele tipuri de relații.

Motivul fundamental de a folosi un wordnet este dezambiguizarea cuvintelor, adică determinarea sensului corect al unui cuvânt.

Ex: Folosim umbrele pentru a ne feri de ploaie. VS Umbrele noastre apar datorită unei surse de lumina.

„Am doi copii foarte deștepți, acum se joacă în parc” VS „Am făcut două copii Xerox pentru proiect”

1.2.2. Sense tagging

- Având un text, trebuie să dăm sensul corect cuvintelor.
- Are nevoie de Word Sense Disambiguation (WSD)
- Se poate folosi în “Question answering”, “Machine Translation”, “Text Mining”

Un corpus este o colecție de text coerent. În “Pos marked corpus” categoriile de cuvinte sunt identificate și marcate. În “Sense marked corpus” cuvintele sunt marcate cu id-ul sensului corespunzător.

Ex Sense marked corpus: Cei doi copii_1138(sens de persoană) au umbrele_3123(sens de obiect).

1.2.3. Clasificarea cuvintelor

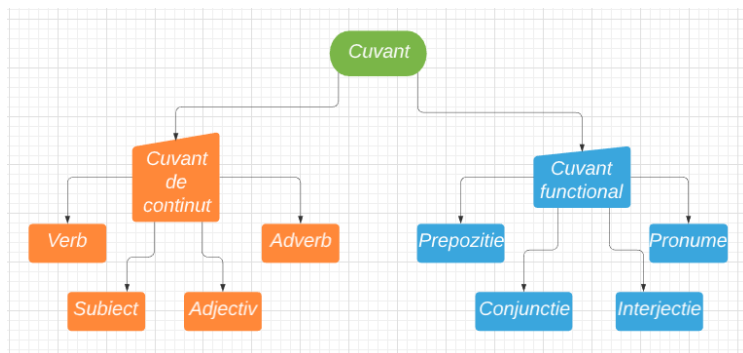


Fig. 3. Clasificarea cuvintelor

Într-un limbaj pot apărea des cuvinte de conținut noi, de obicei datorită unei avansări tehnologice, de exemplu termenul “Xerox”. Cuvintele funcționale definesc un limbaj, deoarece ele nu se schimbă des.

1.2.4. Principii de baza ale Wordnet-ului:

- Cuvintele din limbile naturale sunt polisemantice
- Când mai multe sinonime sunt folosite în același context, apare un sens unic
- Fiecare cuvânt este văzut ca un amalgam de caracteristici semantice
- Competitorul Principiului de baza a Wordnetului, “Relațional Semantics” este “Componential Semantics”

Exemplu “Componential Semantics”:

Fie cuvintele “pisica” și “tigru”, vom decide ținând cont de “componential attributes”.

Cu blană	Carnivor	Mare	Domesticabil
-------------	----------	------	--------------

Tabel 2. Componential attributes

Pentru “pisica” vom avea (Y,Y,N,Y)

Pentru “tigru” vom avea (Y,Y,Y,N)

Este foarte greu să avem o lista completă și corectă de atribute, făcută în aceeași ordine de mai multe persoane, de aceea se preferă “Relațional Semantics”.

Relatiile existente într-un wordnet

1. Synonymy (Egalitate)
2. Hypernymy / Hyponymy (Generalizare și Specializare)
3. Antonymy (Opoziție)
4. Meronymy / Holonymy (desemnarea unei noțiuni printr-un alt termen decât cel exact)
5. Gradation (relație gradată de antonimie)
6. Entailment (relație de implicare)
7. Troponymy (tipul unei relații)

1,3,5 sunt relații lexicale (cuvânt la cuvânt)

2,4,6,7 sunt relații semantice (synset la synset)

Synset (house):

1. house – (a dwelling that serves as living quarters for one or more families; “he has a house on Cape Cod”; “she felt she had to get out of the house”)
2. house – (an official assembly having legislative powers; “the legislature has two houses”)
3. house – (a building in which something is sheltered or located; “they had a large carriage house”)
4. family, household, house, home, menage – (a social unit living together; “he moved his family to Virginia”; “It was a good Christian household”; “I waited until the whole house was asleep”; “the teacher asked how many people made up his home”)
5. theater, theatre, house – (a building where theatrical performances or motion-picture shows can be presented; “the house was full”)
6. firm, house, business firm – (members of a business organization that owns or operates one or more establishments; “he worked for a brokerage house”)
7. house – (aristocratic family line; “the House of York”)
8. house – (the members of a religious community living together)
9. house – (the audience gathered together in a theatre or cinema; “the house applauded”; “he counted the house”)
10. house – (play in which children take the roles of father or mother or children and pretend to interact like adults; “the children were playing house”)
11. sign of the zodiac, star sign, sign, mansion, house, planetary house – ((astrology) one of 12 equal areas into which the zodiac is divided)
12. house – (the management of a gambling house or casino; “the house gets a percentage of every bet”)

Crearea Synset-urilor

3 principii:

- Minimality
- Coverage
- Replaceability

{home,house} -> Structure

{home} -> Motherland

{house} -> Astrological position

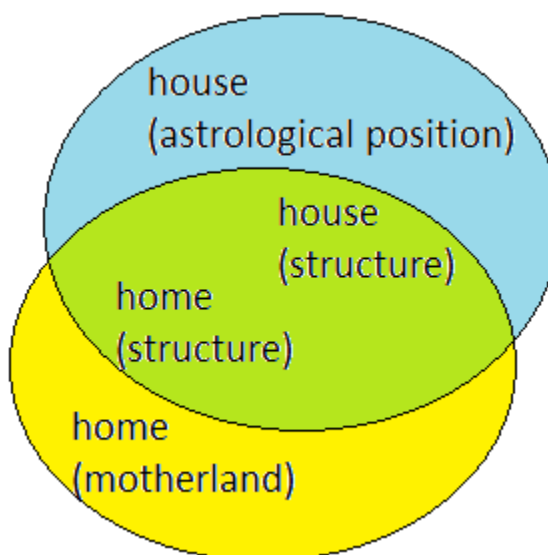


Fig. 4. Sferele synset-urilor

Synsetul {house} este ambiguu, dacă adăugăm, home și family unitatea nu va mai fi ambiguă {family, house, home} (sunt organizate în funcție de frecvența).

Metoda 1:

Un Synset se poate construi prin alegerea tuturor sensurilor dintr-un dictionar, obtinand sinonime pentru fiecare sens. Acest proces dureaza foarte mult.

Metoda 2:

Un Synset se poate construi dintr-un wordnet al unei alte limbi din aceeași familie.

- Traducere, inserție, ștergere.

Synset engleză: {earthquake, quake, temblor, seism} – (shaking and vibration at the surface of the earth resulting from underground movement along a fault plane or from volcanic activity)

Synset română: {cutremur, seism} – (mișcare puternică și bruscă, verticală, orizontală sau de torsiune a scoartei pământului, provocată de dislocări subterane, de erupții vulcanice)

2. Word Sense Desambiguation (WSD)

(Articole folosite pentru învățarea Metodelor de Dezambiguizare [8] [9] [10] [11])

Definiția problemei:

- Obține sensul
 - Unui set de cuvinte țintă
 - Tuturor cuvintelor
- Consultând
 - Un “sense repository” (ca și wordnet)
 - Un “thesaurus” (acesta nu are relații semantice precum wordnet)
- Folosind
 - Contextul în care sunt găsite cuvintele

Exemplu: cuvântul “operation”

- operation((**computer science**) data processing in which the result is completely specified by a rule (especially the processing that results from a single instruction)) “it can perform millions of operations per second”
- operation, **military operation** (activity by a military or naval force (as a maneuver or campaign)) “it was a joint operation of the navy and air force”
- **operation, surgery**, surgical operation, surgical procedure, surgical process (a medical procedure involving an incision with instruments; performed to repair damage or arrest disease in a living body) “they will schedule the operation as soon as an operating room is available”; “he died while undergoing surgery”
- mathematical process, **mathematical operation**, operation ((mathematics) calculation by mathematical methods) “the problems at the end of the chapter demonstrated the mathematical processes involved in the derivation”; “they were learning the basic operations of arithmetic”

WordNet ne ofera:

- a. Membrii Synset-ului și cuvântul în cauza
- b. Gloss-ul ce conține explicația cuvântului
- c. Exemplul de propoziție ce conține sensul
- d. Conceptele semantice (synsets)

Contextul în care apare cuvântul ne da:

- a. Cuvintele vecine
- b. Proprietatile vecinilor

WSD este o acțiune între informația din WordNet și informația din context.

Principiu

Să se dea un sens că și sens dezambiguit care are se potrivește cel mai bine considerând WordNet-ul și contextul.

2.1. Knowledge Based VS Machine Learning Based VS Hybrid Approaches

- Knowledge Based Approaches
 - Depind de resurse de cunoștințe precum WordNet, Thesaurus, etc.
 - Pot folosi reguli gramatice pentru dezambiguizare
 - Pot folosi reguli codate de mâna pentru dezambiguizare
- Machine Learning Based Approaches
 - Depind de Corpus evidence
 - Model de antrenament folosind corpus cu taguri sau fără
- Hybrid Approaches
 - Folosește corpus evidence, dar și relații semantice din WordNet

2.1.1. Knowledge Based Approaches

2.1.1.1. WSD using selectional preferences and arguments

Sensul 1

- This airlines *serves* dinner in the evening flight.
- serve(Verb)
 - agent
 - object – edible

Sensul 2

- This airlines *serves* the sector between Agra&Delhi.
- serve(Verb)
 - agent
 - object – sector

Necesită foarte multe resurse:

- Argument-structure of verbs.
- Selectional preferences of arguments.
- Description of properties of words such that meeting the selectional preference criteria can be decided.

WSD folosind selectional preference are nevoie:

1. Resursă pentru verbe
 - a. VerbNet
 - b. VerbOcean
 - c. Propbank
2. Proprietățile substantivelor
 - a. WordNet

2.1.1.2. Selectional preferences (Teoria lingvistică recentă)

- Există cuvinte ce au nevoie de argumente, precum verbe, prepoziții, adjective și catodată substantive. Aceste argumente sunt de obicei substantive.
- Argumentele trebuie să aibă proprietatea necesară pentru a îndeplini cerință. Trebuie să satisfacă Selectional Preferences.

Ex: Give (verb)

agent – animate

obj – direct

obj – indirect

I gave him the book

I gave him the book(yesterday in the school) -> adjunct

- Cum ajută acest lucru WSD-ul?
 - Un tip de informație contextuală este informația ce descrie tipul de argumente ale unui cuvânt.

Verb Argument frame

Structura ce tinde către un cuvânt se numește Argument Frame. Selection Preference referinduse la proprietățile “supply words” ce întâlnesc tendințele setului anterior.

Exemplu 1 de Argument frame:

Propoziție: *I am fond of X (Propoziția nu se poate considera completa dacă X nu există)*

Fond

{

Arg1: *Prepositional Phrase (PP)*

{

PP: *of NP (PP trebuie să înceapă cu “of” și să fie urmat de Noun Phrase)*

{

N: *somebody/something (Noun-Substantivul)*

}

}

}

Exemplu 2 de Argument frame

Verb: give

Give

{

agent: <the give> animate (Persoana care da)

direct object: <the thing given> (Obiectul care este dat)

indirect object: <beneficiary> animate/organization (Celui care i se da obiectul)

}

[I]_{agent} gave a [book]_{doj} to [Andrei]_{iobj}*

2.1.1.2.1. Critică

Are nevoie de un volum mare de materiale în formă machine-readable:

- Argument-structure of verbs
- Selectional preferences of arguments
- Descrierea proprietăților cuvintelor încât să fie conform cu Selectional preferences
 - Ex: This flight serves the “region” between Bucharest and Ploiesti
 - Cum decidem dacă “region” este compatibil cu “sector”? Folosim o resursă lexicală.

Acuratețea Selection preference based WSD este de 44% conform studiilor efectuate pe Brown corpus (un corpus creat în 1960 la universitatea din Rhode Island).

2.1.1.3. Overlap based approaches

- Are nevoie de un Machine Readable Dictionary (MDR).
- Găsește overlap-ul dintre caracteristicile diferitelor sensuri ale unui cuvânt ambiguu (sense bag) și caracteristicile cuvintelor din context (context bag)
- Aceste caracteristici pot fi definiții de sens, propoziții exemplu, hypernoms, etc.
- Caracteristicile pot avea și greutăți (weights)
- Sensul care are cel mai mare overlap este selectat ca și sensul corect.

2.1.1.4. Algoritmul lui Lesk

Sense Bag: conține cuvintele din definiția unui sens candidat al cuvântului ambiguu.

Context Bag: conține cuvintele din definiția fiecărui sens al fiecărui cuvânt din context.

Ex: “On burning *coal* we get *ash*.”

Informații din wordnet:

- Substantivul “ash” are 3 sensuri (primele 2 din text cu taguri)
- (2) ash – (the residue that remains when something is burned)

- (1) ash, ash tree – (any of various delicious pinnate-leaved ornamental or timber trees of the genus *Fraxinus*)
- ash – (strong elastic wood of any of various ash trees; used for furniture and tool handles and sporting goods such as baseball bats)
- The verb ash has 1 sense (no senses from tagged texts)
- ash – (convert into ashes)

2.1.1.4.1. Critică

- Substantivele proprii în contextul unui cuvânt ambiguu pot fi dezambiguizatori puternici

Ex: “Brett Lee” este un indicator puternic către domeniul “sports”

Brett Lee plays cricket

(astfel ne dăm seama că este vorba de spot, ci nu de insectă)

- Substantivele proprii nu sunt mereu prezente în WordNet. Astfel acest tip de abordare eșuează în obținerea indiciilor puternice oferite de substantivele proprii.
- Acuratețea este de 50%, testat pe 10 cuvinte foarte polismantice din limba engleză.

Exemplu:

1. On burning coal we get ash. (“ash” este cuvântul de dezambiguizat, “burning” este indiciul)
2. On burning the ash, we found that its roots were deep into the ground. (aici ash are sensul de copac, deoarece avem indiciul “roots”, dar acest indiciu nu există și în sensul de copac al cuvântului “ash” din WordNet. Astfel Overlap based approach nu poate dezambiguiza acest exemplu, dar dacă ne folosim de cuvintele din Sense bag, descoperim cuvântul “tree”, iar dacă ne uităm la părțile unui “tree” prin Meronymy o să obținem cuvântul “root” care se va potrivi că și indiciu) Cuvântul “burning” este un indiciu ce ne induce în eroare.

2.1.1.5. Algoritmul lui Lesk extins

Algoritmul original este sensibil cu cuvintele existente în definiție. Exstensia include Gloss-urile sensurilor legate semantic din WordNet (ex. hypernyms, hyponyms, etc.)

$$\text{SCORE}_{\text{ext}}(S) = \sum_{s' \subseteq \text{rel}(S) \text{ or } s=s'} \sum |\text{context}(w) \cap \text{gloss}(s')|$$

Unde: $\text{gloss}(S)$ este gloss-ul sensului S din resursă lexicală;

$\text{context}(W)$ este gloss-ul fiecărui sens al fiecărui cuvânt din context;

$\text{rel}(s)$ ne da sensurile legate de S în WordNet prin intermediul unor relații.

Exemplu: On burning the ash, we found that the root was deep into the ground.

“root” este indiciul, sens bag-ul conține cuvântul tree, cuvântul tree are cuvântul root prin Meronymy astfel “root” din context se potrivește cu “root” din sense repository astfel se va dezambiguiza.

“On combustion of coal we get ash”

Informații din wordnet:

- Substantivul “ash” are 3 sensuri (primele 2 din text cu taguri)
- (2) ash – (the residue that remains when something is burned)
- (1) ash, ash tree – (any of various delicious pinnate-leaved ornamental or timber trees of the genus Fraxinus)
- ash – (strong elastic wood of any of various ash trees; used for furniture and tool handles and sporting goods such as baseball bats)
- The verb ash has 1 sense (no senses from tagged texts)
- ash – (convert into ashes)

Cuvântul “combustion” nu apare în cele de mai sus deci se va returna 0, dar dacă folosim Algoritmul lui Lesk extins.

Din WordNet (prin hyponymy)

ash – (the residue that remains when something is burned)

- ⇒ fly ash – (fine solid particles of ash that are carried into the air when fuel is combusted)
- ⇒ bone ash – (ash left when bones burn; high in calcium phosphate; used as fertilizer and in bone china)

2.1.1.5.1. Critică

Sunt regiuni mai mari pentru matching în WordNet, lucru ce aduce o șansă mai bună de a găsi un Match, dar și de a avea un “Topic Drift”. (dacă cuvântul combustion ar mai fi avut un alt sens atunci am fi putut înlocui un rezultat eronat)

2.1.1.6. Algoritmul lui Walker

Este bazat pe Thesaurus în loc de WordNet

Pasul1: Pentru fiecare sens al cuvântului țintă se găsește categoria în thesaurus al cărui sens aparține.

Pasul2: Se calculează scorul pentru fiecare sens folosind cuvinte din context. Un cuvânt din context va adăuga 1 la scorul sensului dacă categoria thesaurus al cuvântului se potrivește cu categoria sensului.

Exemplu: *The money in this bank fetches an interest of 8% per annum.*

Cuvântul țintă: **bank**

Cuvintele indiciu din context: **money, interest, annum, fetch**

	Sens1: Finance	Sens2: Location
Money	+1	0
Interest	+1	0
Fetch	0	0
Annum	+1	0
Total	3	0

Tabel 7. Scorul conform algoritmului lui Walker

Algoritmul lui Walker depinde de informația ontologică a cuvintelor – conceptul lor, ierarhie, sens, etc.

2.1.1.7. WSD folosind Conceptual Density (Agirre and Rigau, 1996)

Selectează un sens pe baza “relatedness” al sensului cuvântului în raport cu contextul.

Relatedness este măsurat în conceptual distance (ex: cât de aproape este conceptul reprezentat de cuvânt și conceptul reprezentat de cuvintele din context)

Acest tip de abordare folosește ierarhi structurate semantic net (WordNet) pentru a găsi conceptual distance.

Cu cât conceptul distace este mai mică cu atât va fi mai mare conceptual density. (ex: dacă toate cuvintele din context sunt indicatori puternici al unui concept atunci acel concept va avea o densitate mare)

Formula Conceptual Density

- Conceptual distance între două cuvinte are trebui să fie proporțională lungimii caii dintre cele două cuvinte într-un Tree ierarhic (WordNet)
- Conceptual distance între două cuvinte are trebui să fie proporțională cu adâncimea conceptelor din ierarhie.

$$CD(c,m) = \frac{\sum_{i=0}^{m-1} nhyp_i^{0.20}}{descendants_i}$$

Unde:

c = concept;

nhyp = mean number of hyponyms;

h = height of the sub-hierarchy;

m = no, of senses of the word and senses of context words contained in the sub-ierarchy;

CD = Conceptual Density;

0.2 is the smoothing factor.

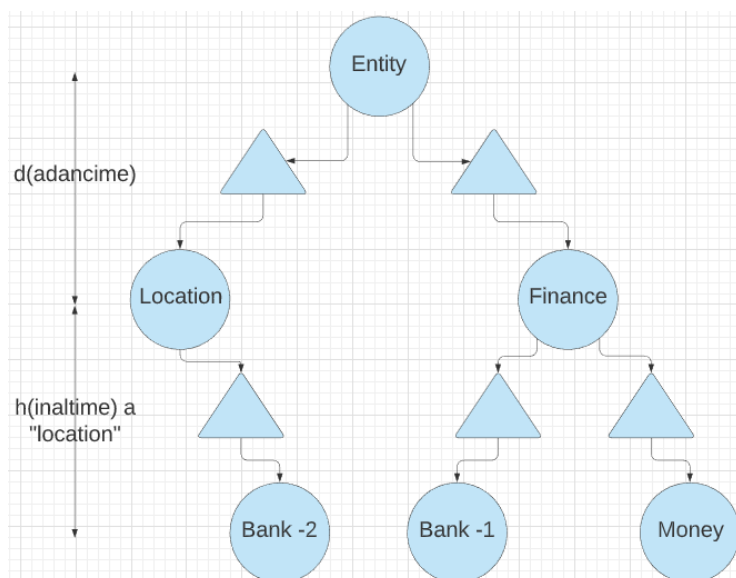


Fig.6. Aplicarea densității conceptuale

Exemplu:

Distanță între pisica și câine:

$$d(\text{pisica}, \text{câine}) = 2$$

Distanță între abstract și concret:

$$d(\text{abstract}, \text{concret}) = 2 \quad (\text{de la abstract la părinte} = 1 + \text{de la părinte la concret} = 1)$$



Fig.7. Densitatea conceptuală - exemplu

Astfel putem zice că similaritatea dintre Pisica și Câine este egală cu similaritatea dintre Abstract și Concret? Nu, deoarece Abstract și Concret sunt concepte foarte mari cu aproape nici o similaritate, dar Pisica și Câine sunt ambele animale cu 4 picioare, carnivore, există diferențe între acestea, dar sunt mult mai similare decât Abstract și Concret. Deci nu trebuie să ne lăsăm păcăliți de distanță dintre două noduri, trebuie să considerăm nivelul la care aceste concepte există (cât de general, cât de specific sunt ele).

Exemplu de Conceptual Density:

The jury(2) praised the administration(3) and operation(8) of Bucharest Police Department(1).
 (“(n)” numărul de sensuri, operation având 8 este foarte ambiguu)

Pasul 1: Se crează un grafic cu substantivele din context, sensurile lor și hypernyms.

Pasul 2: Se calculează Conceptual density a sub-ierarhiilor

Pasul 3: Conceptul cu cel mai mare scor CD este selectat

Pasul 4: Se selectează sensurile ce se află sub conceptul câștigător și sensurile corespunzătoare cuvintelor.

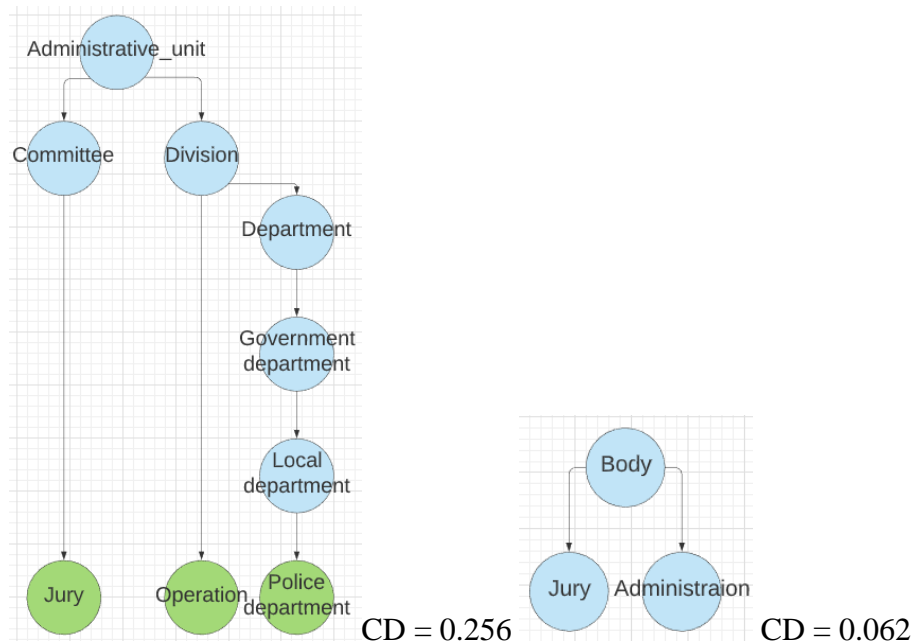


Fig.8. Exemplu “Jury” pentru Conceptual density

2.1.1.7.1. Critică

Rezolvă ambiguitatea lexicală a substantivelor găsind o combinație de sensuri ce maximizează Conceptual Density dintre acestea.

Avantaj: Nu este necesar un corpus cu tagguri

Dezavantaj: Nu poate folosi indicii puternice date de substantivele proprii din context. Această este o problema generală a abordărilor Overlap.

Acuratețea: 54% pe Brown corpus

2.1.1.8. WSD folosind algoritmul Random Walk (Page Rank) (sinha and Mihalcea, 2007)

“The church bells no longer rung on Sundays.”

church

- 1: one of the groups of Christians who have their own beliefs and forms of worship
- 2: a place for public (especially Christian) worship
- 3: a service conducted in a church

bell

- 1: a hollow device made of metal that makes a ringing sound when struck
- 2: a push button at an outer door that gives a ringing or buzzing signal when pushed
- 3: the sound of a bell

ring

1: make a ringing sound

2: ring or echo with sound

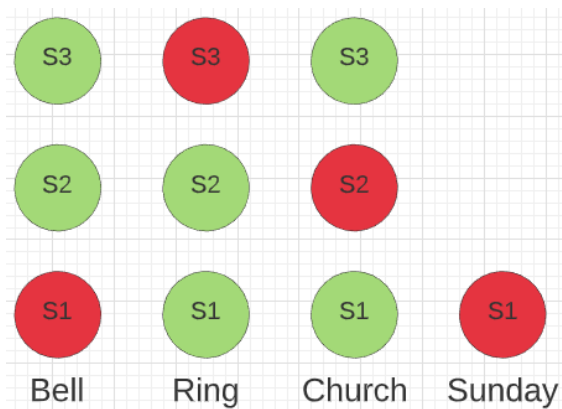
3: make(bells) ring, often for the purposes of musical edification

Sunday

1: first day of the week; observed as a day of rest and worship by most Christians

Roșu=sensul potrivit

Pasul 1: Se adaugă vectori către toate sensurile posibile



Pasul 2: Se adaugă greutatea muchiilor folosind similaritate semantică (Metodă Lesk)

Pasul 3: Se folosește graph based ranking algorithm pentru a găsi scorul fiecărui vector(fiecărui sens de cuvânt)

Pasul 4: Se selectează sensurile cu cel mai mare scor

Fig. 9. Sensuri înainte de aplicarea Page Rank

Astfel rezultă:

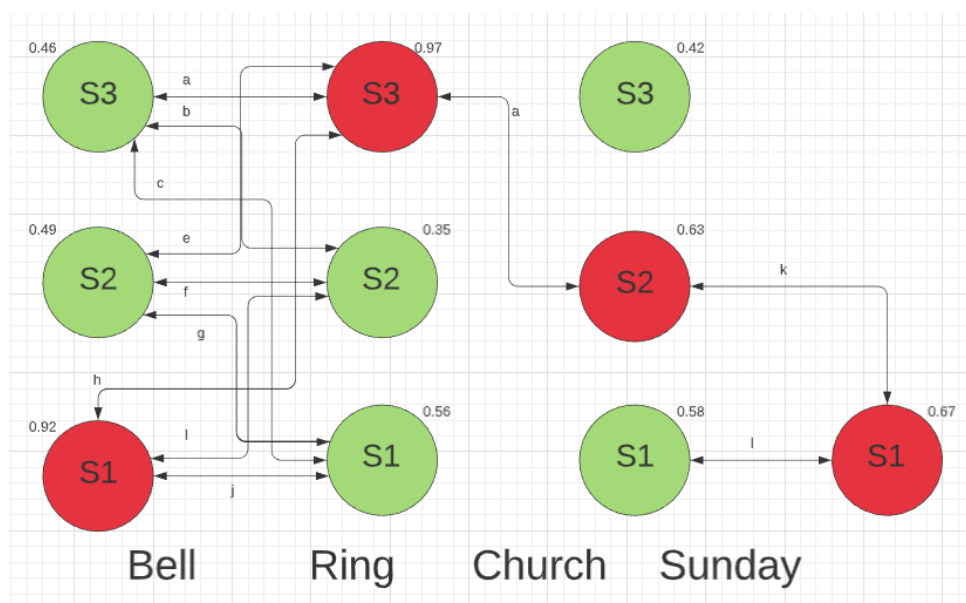


Fig. 10. Sensuri după de aplicarea Page Rank

2.1.1.8.1. Un overview Page Rank

Este un algoritm creat la Universitatea Stanford de Larry Page (de aici vine numele de Page-Rank) și Sergey Brin că și parte dintr-un proiect de cercetare despre un noi tip de motor de căutare. Prima lucrare despre acest proiect, ce descrie PageRank și primul prototip folosit de motoarele de căutare Google, a fost publicat în 1998. După un scurt timp, Page și Brin au înființat Google Inc., compania ce deține Google search engine. Unul din mulții factori care determina rankingul căutării Google, PageRank, continuă și în ziua de astăzi să ofere baza oricarui tool de căutare Google. Pagerank este o distribuție a probabilității folosită pentru a reprezenta șansele unei persoane care face click-uri random pe link-uri, de a accesa o pagină anume.

Exemplu: Fie doar patru websituri, A, B, C, D

Prima aproximare făcută ar fi egală între toate cele 4 documente, deci fiecare începe cu o estimare de 0.25.

Dacă paginile B,C și D au link-uri doar către A fiecare ar oferi un punctaj PageRank de 0.25 către A. Deci în final am avea:

$$PR(A) = PR(B) + PR(C) + PR(D) = 0.75$$

Să presupunem că pagină B are un link către pagina C și un link către pagină A, iar D are link-uri către toate celelalte 3 pagini.

Valoarea link-voturilor este împărțită între toate linkurile de pe pagină. Astel B da un vot cu valoare 0.125 către A și un vot cu valoare 0.125 către C. Pagină D oferă doar 0.083/link (o treime) către A,B,C.

$$PR(A) = PR(B)/2 + PR(C)/1 + PR(D)/3$$

În general $PR(u) = \sum PR(v)/L(v)$, v aparține $B(u)$, unde $B(u)$ este setul de pagini la care "u" este linked și $L(v)$ este numărul de link-uri de la "v".

Pentru a captura fenomenul în care un user se oprește din clickuri formulă va deveni:

$PR(u) = (1-d)/N + d \cdot \sum PR(v)/L(v)$, v aparține $B(u)$; unde N =mărimea colecției de documente, iar d este un factor de amortizare;

2.1.1.8.2. Aplicarea Page Rank pentru WSD

Fie un graf $G = (V, E)$

$In(V_i)$ = predecesorii lui V_i

$Out(V_i)$ = succesorii lui V_i

$$S(V_i) = \sum_{j \in In(V_i)} \frac{1}{Out(V_j)} S(V_j)$$

Într-un graf cu greutate, se selectează la întâmplare o muchie cu o probabilitate de a selecta pe cele cu greutatea mai mare.

$$WS(V_i) = \sum_{j \in In(V_i)} \frac{w_{ji}}{\sum_{v_k \in Out(v_i)} w_{jk}} WS(V_j)$$

2.1.1.8.3. Alți algoritmi link based

- HITS inventat de Jon Kleinberg (folosit de Teoma si Ask.com)
- IBM CLEVER project
- TrustRank

2.1.1.8.4. Critică

Se bazează pe plimbări aleatorii pe grafice care codifică dependențele de etichete.

Avantaje:

- Nu are nevoie de date cu taguri (un WordNet este suficient).
- Greutățile de pe muchii conțin similaritățile semantice dintre definiții.
- Ține cont de datele globale recursiv extrase din tot graful.

Dezavantaje:

- Acuratețe mică

Acuratețe:

- 54% pe corpusul SEMCOR care are un baseline accuracy de 37%

2.1.1.9. Compararea abordarilor Knowledge based

Algoritm	Acuratețe
WSD folosind Selectional Restrictions	44% pe corpusul Brown
Lesk's algorithm	50-60% pe exemple scurte din "Pride and Prejudice" și din "News stories"
Extended Lesk's algorithm	32% pe exemple Lexicale din Senseval 2
WSD using conceptual density	54% pe corpusul Brown
WSD using Random Walk Algorithms	54% pe corpusul SEMCOR care are un baseline de 37%
Walker's algorithm	50% pe 10 cuvinte polisemantice din limba Engleză.

Tabel 8. Compararea abordarilor Knowledge based

2.1.1.9.1. Concluzia abordărilor Knowledge based

Dezavantajele folosirii Selectional Restrictions în WSD:

- Are nevoie de Knowledge Base

Dezavantajele folosirii abordării Overlap Based:

- Definițiile din dicționar sunt de obicei foarte scurte
- Conținutul unui dicționar rar ține cont de distributional constraints ale diferitelor sensuri(ex. “cigarette” și “ash” nu apar împreună în dicționar)
- Sparse match
- MRD nu conține substantive proprii, astfel eșuând în depistarea indicilor puternice date de acestea.

2.1.2. Abordări Supervizate

2.1.2.1. Naive Bayes

Folosim algoritmul acesta pentru a găsi sensul câștigător:

$$s^{\wedge} = \operatorname{argmax}_{s \in \text{senses}} PR(s/V_w)$$

s^{\wedge} este sensul țintă al unui cuvânt w

Argmax maximizează pe probabilitatea diferitelor sensuri ale cuvântului w

V_w este un vector de caracteristică construit pe w și cuvintele din acest enviroment.

Ex: I went to the **bank** to *withdraw* some *money*.

cuvânt de dezambiguizat: “bank” (acesta este w)indicii: “withdraw”, “money”

deci vom avea $V_{\text{bank}} = \langle I, \text{went}, \text{to}, \text{the}, \text{to}, \text{withdraw}, \text{some}, \text{money} \rangle$ cuvântul ”bank” lipsește din vector deoarece acesta este cuvântul de dezambiguizat.

V_w conține :

- Informatii privitoare la ce parte de propoziție este cuvântul w , în exemplul de față “bank” este un substantiv.
- Caracteristici semantice și sintactice ale cuvântului w :

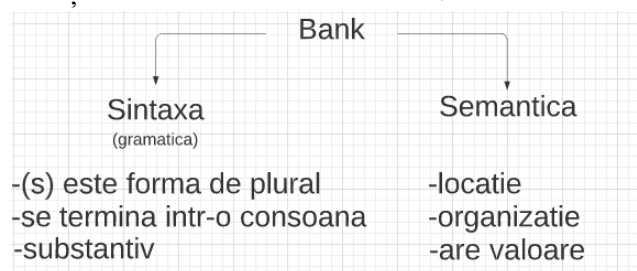


Fig. 11. Informații despre cuvântul “Bank”

- Vector de colocatie, de obicei conținând următorul cuvânt(+1), cuvântul de după cuvântul următor(+2),-2,-1 și informații privitoare la ce parte de propoziție sunt acestea. (alcătuit din cuvinte)

Ex: I went to the **bank** to *withdraw* some *money*.

1. Ignorăm cuvintele funcționale (pentru reamintire se introduce o schemă explicativă)

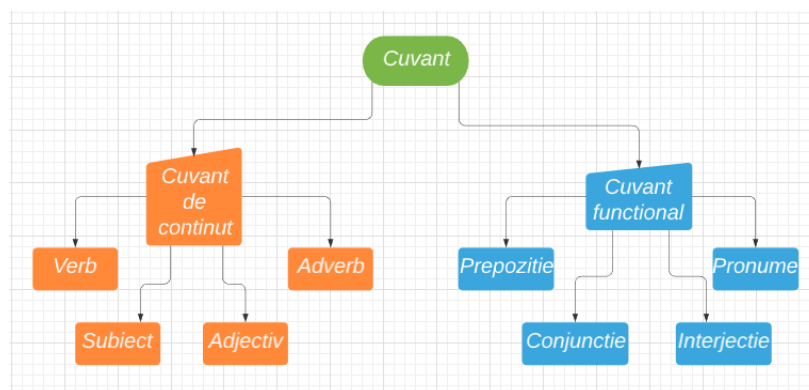


Fig. 3. Clasificarea cuvintelor

2. Se folosesc cuvintele de conținut: -2,-1,+1,+2

<I,went,withdraw,mony> acesta va fi folosit pentru WSD

- Vectorul de coocurență (de câte ori *w* apare într-o mulțime de cuvinte din jurul sau). (alcătuit din numere)

Ex: I went to the **bank** to *withdraw* some *money*.

Ne alegem un “window” de 2 cuvinte înainte de “bank”(to,the) și 2 cuvinte după “bank”(to,withdraw) => cuvântul “bank” apare o singură dată.

Aplicând regula Bayes și asumarea de independență Naive:

$$s^{\wedge} = \operatorname{argmax}_{s \in \text{senses}} \operatorname{PR}(s) \cdot \prod_{i=1}^n \operatorname{PR}(V_w^i | s)$$

bank:

vectorul de caracteristici $V_{\text{bank}} \langle N, \text{organization, plural-s, went, withdraw, money} \rangle$

$$P(V_{\text{bank}} | \text{bank}) = ??$$

$$P(\langle N, \text{organization, plural-s, went, withdraw, money} \rangle | \text{bank}) =$$

$$P(N|\text{bank}) \cdot P(\text{organization}|\text{bank}) \cdot P(\text{plural-s}|\text{bank}) \cdot P(\text{went}|\text{bank}) \cdot P(\text{withdraw}|\text{bank}) \cdot P(\text{money}|\text{bank})$$

nota: |bank se refera la un sens al cuvântului bank.

Prin independență se înțelege că, caracteristică “N”(noun=substantiv) este independență față de caracteristică “organization”, independență față de caracteristică “plural-s”, etc.

Fie $\text{Sens1}_{\text{bank}} = \text{sens financiar}$

$\text{Sens2}_{\text{bank}} = \text{river bank}$

$P(N|\text{Sens1}_{\text{bank}})$ vs $P(N|\text{Sens2}_{\text{bank}})$ această caracteristică nu va avea impact.

$P(\text{Organizație}|\text{Sens1}_{\text{bank}}) > P(\text{Organizație}|\text{Sens2}_{\text{bank}})$ această caracteristică o să încline balanța în favoarea sensului financiar.

Estimarea Parametrilor

$\operatorname{PR}(s)$ – probabilitatea unui sens

$\operatorname{PR}(V_w^i | s)$ – probabilitatea unei caracteristici individuale

Această explicație va fi făcută cu privire la WORDNET:

$$PR(s) = \text{count}(s,w)/\text{count}(w)$$

$$PR(V_w^i|s) = PR(V_w^i,s)/PR(s) = \frac{\text{count}(V_w^i,s,w)/\text{count}(w)}{\text{count}(s,w)/\text{count}(w)} = \frac{\text{count}(V_w^i,s,w)}{\text{count}(s,w)}$$

2.1.2.2. Decision List Algorithm

Se asumă “One sense per collocation” – cuvintele vecine ne dau indicii puternice și consistente despre sensul țintă.

1. Se colectează un set mare de colocații pentru cuvântul ambiguu.
2. Se calculează word-sense probability distributions pentru toate colocațiile.
3. Se calculează log-likelihood ratio.

$$LOG \frac{PR(\text{SenseA}|\text{COLLOCATION}_i)}{PR(\text{SenseB}|\text{COLLOCATION}_i)}$$

Asumand că există doar două sensuri, această formulă poate fi extinsă pentru k sensuri.

4. Cu cât log-likelihood este mai mare, cu atât de mult avem mai multă evidență predictivă.
5. Colocațiile sunt ordonate într-un decision list, cu cele mai predictive colocații pe primele rank-uri.

Exemplu: cuvântul “plant”

Date de antrenament:

Sens	Exemple pentru antrenament (keywords in context)
A	<p>used to strain microscopic plant life from the ...</p> <p>... zonal distribution of plant life, ...</p> <p>close-up studies of plant life and natural ...</p> <p>too rapid growth of aquatic plant life in water ...</p> <p>... the proliferation of plant and animal life ...</p> <p>establishment phase of the plant virus life circle ...</p>
B	<p>computer manufacturing plant and adjacent ...</p> <p>discovered at St. Louis plant manufacturing</p> <p>... copper manufacturing plant found that they</p> <p>copper wire manufacturing plant, for example ...</p> <p>a cement manufacturing plant in ...</p> <p>polystyrene manufacturing plant at its ...</p> <p>company manufacturing plant is in ...</p>

Tabel 9. Setul de antrenament

Astfel se determină:

Cuvant	Sens
growth	A
car	B
hight	A
union	B
equipment	B
assembly	B
nuclear	B
flower	A
job	B
crude	A
species	A

Tabel 10. Rezultatul algoritmului

Ex: ... plucking *flowers* affects **plant** *growth* ...
flowers și *growth* fac parte din sfera sensului A

2.1.2.2.1. Critică

Pozitive:

- Nu este nevoie de un corpus cu taguri. Implementarea se face ușor.
- Algoritm semi-supervizat simplu ce extinde un algoritm supervizat deja existent.
- Înțelegere ușoară a listei de decizie rezultată.
- Poate folosi indiciile date de substantive proprii din corpus. (lucru ce este foarte greu pentru alți algoritmi)

Negative:

- Clasificatorul este word-specific
- Un nou clasificator trebuie antrenat pentru fiecare cuvânt ce vrem să îl dezambiguizăm.

Acuratețe:

- În medie avem o acuratețe de 96%, când este testat pe o lista de 12 cuvinte polisemantice.

2.1.2.3. Exemplar Based WSD (k-nn)

Un exemplar based classifier este construit pentru fiecare cuvânt ce trebuie dezambiguit.

Pasul 1: Pentru fiecare propoziție marcată cu sens ce conține un cuvânt ambiguu, se construiește un exemplu pentru antrenament:

- Informații privitoare la ce parte de propoziție este cuvântul w și vecinii acestuia (POS)
- Colocatii locale
- Vector de Coocurență
- Caracteristici morfologice
- Dependențe sintactice subject-verb

Pasul 2: Se dă o propoziție de test ce conține cuvântul ambiguu și un exemplu de test este construit.

Pasul 3: Exemplul de test este comparat cu toate exemplele de antrenament și cele mai apropiate k exemple de antrenament sunt selectate.

Pasul 4: Sensul ce se găsește cel mai des în aceste exemple " k " este selectat ca și sensul corect.

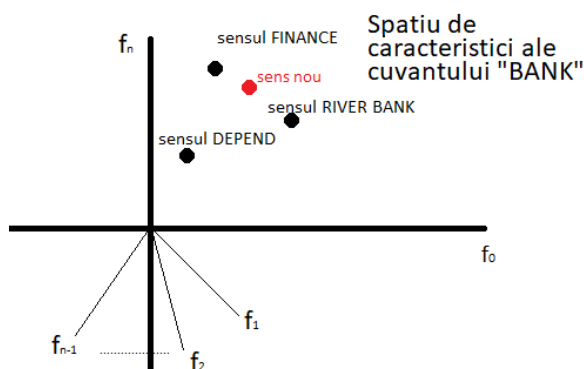


Fig. 12. Reprezentarea unui sens nou prin intermediul k-nn

sensul nou apare în următoarea propoziție:

Banks are closed on National holidays.

În urmă construirii unui vector putem determina cel mai apropiat sens, în cazul de față sensul nou este apropiat cel mai mult de sensul “FINANCE”, astfel acesta va fi selectat că și sensul corect.

2.1.2.4. WSD folosind SVMs

SVM este un clasificator binar care găsește un hyper plane cu cea mai mare margine care separă exemple de antrenament în două clase. SVM-urile sunt clasificatori binari, astfel un clasifior separat este construit pentru fiecare sens al unui cuvânt.

Faza de antrenament: Folosind un corpus cu taguri, pentru fiecare sens al cuvântului se antrenează un SVM folosind următoarele caracteristici:

- POS (Informații privitoare la ce parte de propoziție este cuvântul w), dar și POS vecinilor săi.
- Colocații locale
- Vector de coocurență
- Caracteristici bazate pe relații sintactice (ex: titlu, POS al titlului, etc.)

Faza de testare: Dându-se o propoziție de test, se construiește un test exemplu folosind caracteristicile de mai sus și date ca input către fiecare clasificator binar.

Sensul corect este selectat pe baza unei etichete date de fiecare clasificator.

2.1.2.5. WSD folosind Perceptron Trained HMM (Hidden Markov Model)

WSD este tratat ca și o secvență de etichetare. Spațiul claselor este redus folosind “super senses” din WordNet, în loc de “actual senses”. Un HMM discriminativ este antrenat folosind următoarele caracteristici:

- POS al w, dar și POS al vecinilor acestuia.
 - Colocații locale.
 - Formă cuvantului și a vecinilor săi.
- Ex: dacă s = “Merrill Lynch & Co” atunci $\text{shape}(s) = Xx * Xx * \& Xx$

Este bun pentru NER deoarece există etichete ca “person”, “location”, “time” etc., care sunt incluse în super sensul setului de etichete.

2.1.2.5.1. Compararea abordărilor Supervizate

Algoritm	Precizie	Recall	Corpus	Acuratețe
Naive Bayes	64.13%	Ne raportat	Senseval3 – All Words Task	60.90%
Decision Lists	96%	Ne aplicabil	Testat pe un set de 12 cuvinte polisemantice	63.9%
Exemplar Based Disambiguation(k-nn)	68.6%	Ne raportat	WSj6 ce conține 192 cuvinte de conținut	63.7%
SVM	72.4%	72.4%	Senseval3 – O mostră lixicala (folosit pentru dezambiguizarea a 57 de cuvinte)	55.2%
Perceptron trained HMM	67.60%	73.74%	Senseval3 – All Words Task	60.90%

Tabel 11. Compararea abordărilor supervizate

O acuratețe de 60-65% sau mai mare este considerată bună în WSD.

2.1.3. Abordări Semi-Supervizate

2.1.3.1. Ideea de bază a WSD Semi-Supervizat

1. Avem nevoie de seed training data;
2. Antrenare folosind seed data;
3. Etichetare unseen data;
4. Corectarea etichetelor manual;
5. Se antrenează iar, folosind mai multe date;
6. Repetă pașii 3,4 până când nivelul satisfacator de acuratețe este atins.

2.1.3.2. Decision List Algorithm Semi-Supervizat

Este bazat pe algoritmul supervizat al lui Yarowsky ce folosește Decision Lists.

Pasul 1: Se antrenează folosind puține date din seed.

Pasul 2: Se clasifică tot sample setul folosind un clasificator antrenat.

Pasul 3: Se crează noi date de seed adăugând membrii care sunt etichetați cu probabilitate mare că Sens-A sau că Sens-B.

Pasul 4: Se reantreneaza clasificatorul folosind noile seed dată.

Se folosește de proprietatea “One sense per discourse”. Identifica cuvinte ce sunt etichetate cu un scor de confidența mic și le etichetează cu sensul dominant din documentul respectiv.

2.1.3.2.1. Compararea abordărilor Semi-Supervizate

Algoritm	Precizie	Corpus	Acuratețe
Decision Lists Supervizate	96.1%	Testat pe un set de 12 cuvinte polisemantice	63.9%
Decision Lists Semi-Supervizate	96.1%	Testat pe un set de 12 cuvinte polisemantice	63.9%

Tabel 12. Compararea abordărilor Semi-Supervizate

Funcționează în mod egal cu versiunea să supervizată, chiar dacă are nevoie de mult mai puține date.

2.1.4. Abordări Nesupervizate

Întrebare cheie în WSD Nesupervizat

Funcționează în mod egal cu versiunea să supervizată, chiar dacă are nevoie de mult mai puține date?

Acesta problema se rezolva având un lexicon de etichete, care nu este folosit pentru a marca datele de antrenament, ci pentru a da etichetele în momentul în care am determinat sensul cuvântului. Deci aici trebuie rezolvată lacună dintre decizia sensului de cuvânt și eticheta cuvântului.

2.1.4.1. Hyperlex (Veronis Jean. 2004. HyperLex: Lexical cartography for information retrieval. *Computer Speech & Language*, 18(3):223-252

În loc de a folosi “dictionary defined senses”, se extrage “senses from the corpus itself”. Aceste sensuri din corpus corespund clusterelor ale cuvintelor ce au un context similar.

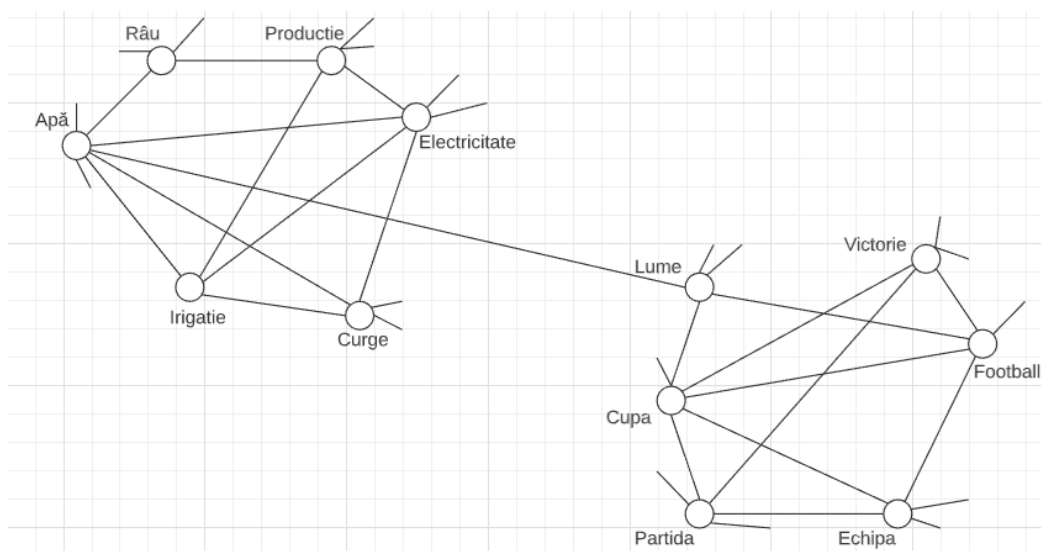


Fig. 13. Graf al coocurenței cuvântului Francez “barrage”-tradus în română

Plant

Sens-1: After *watering*, the **plant** *bloomed*.

Sens-2: The chemical *manufacturing* **plant** was *built* in 2 years.

Dacă se găsesc cuvintele indiciu atunci se va alege sensul corespunzător.

2.1.4.1.1. Detectarea ROOT HUBS

Diferite moduri de folosire a cuvântului țintă formează bundel-uri foarte interconectate (componente cu densitate mare). În fiecare component cu densitate mare unul din noduri (hub) are un grad mai mare decât celelalte.

Pasul 1: Se construiește un graf de coocurență G.

Pasul 2: Se aranjează nodurile în G în ordine descrescătoare în funcție de “în-degree”.

Pasul 3: Se selectează nodul din G care are cea mai mare frecvență. Acest nod va fi hub-ul primului component de densitate mare.

Pasul 4: Se șterge acest hub și toți vecini din G.

Pasul 5: Se repetă pașii 3 și 4 pentru a se detecta hub-urile altor componente cu densitate mare.

Termenul francez “barrage” are 4 sensuri:

1. Eau (construction, engineering-work, river, project, reservoir, flood)
2. Routier (vehicle, truck, member, driver, policeman, group)
3. Frontiere (Algeria, military, efficiency, army, Switzerland, post)
4. Match (winner, victory, encounter, qualification, shot, soccer)
- 5.

2.1.4.1.2. Delimitarea componentelor

Se atașează fiecare nod la cel mai apropiat root hub. Distanță dintre două noduri este măsurată ca și cea mai mică sumă de greutatea ale muchiilor ce fac legătură dintre ele.

Pasul 1: Se adaugă cuvântul țintă la graful G

Pasul 2: Se crează un Minimum Spanning Tree (MST) pe G folosind cuvântul țintă ca și root.

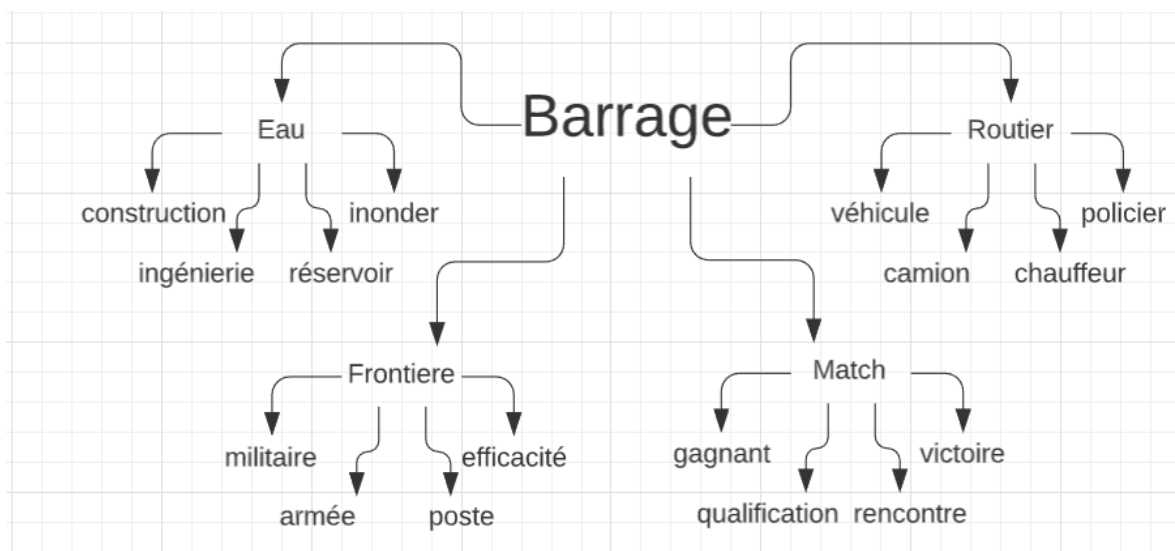


Fig. 14. MST pentru cuvântul francez “Barrage”

2.1.4.1.3. Dezambiguizare

Fiecărui nod din MST îi este dat un vector de scor cu atâtea dimensiuni ca și numărul de componente.

$$S_i = \frac{1}{1+d(h_i,v)} \text{ dacă } v \text{ aparține componentei } i,$$

$$S_i = 0 \text{ dacă } v \text{ nu aparține componentei } i.$$

Ex: pluie(ploaie) aparține componentei EAU (apa)

și $d(\text{eau}, \text{pluie}) = 0.82$, $S_{\text{pluie}} = (0.55, 0, 0, 0)$.

Pasul 1: Pentru un context dat, se adaugă vectorii de scor ale tuturor cuvintelor din context.

Pasul 2: Se selectează componentul care primește cea mai mare greutate.

2.1.4.1.4. Adaugarea greutăților pe muchii

Fiecărei muchi i se atribuie o greutate care scade cu cât frecvența de asociere a cuvintelor crește.

$$W_{A,B} = 1 - \max[P(A|B), P(B|A)]$$

$$P(A|B) = f_{A,B}/f_B, P(B|A) = f_{B,A}/f_A$$

$P(\text{eau}|\text{ouverage}) = 183/479 = 0.38$; $P(\text{ouverage}|\text{eau}) = 183/1057 = 0.17$, $w = 1 - 0.38 = 0.62$;

	EAU(Apa)	~EAU	Total
OUVERAGE(Muncă)	183	296	479
~ OUVERAGE	874	5556	6430
Total	1057	5852	6909
POTABLE(potabilă)	63	0	63
~ POTABLE	994	5852	6846
Total	1057	5852	6909

Tabel 13. Scorul greutatilor de pe muchii

Exemplu:

Le **barrage** recueille l’eau a la saison des pluies.

Barajul colectează apă în sezoanele ploioase.

Scorul pentru contextul “Le **barrage** recueille l’eau a la saison des pluies”

	EAU	ROUTIER	FRONTIERE	MATCH
S_{eau}	1.00	0.00	0.00	0.00
S_{saison}	0.00	0.00	0.00	0.39
S_{pluie}	0.55	0.00	0.00	0.00
Total	1.55	0.00	0.00	0.39

Tabel 14. EAU este castigator

Un coeficient de reliabilitate (p) poate fi calculat ca diferența(δ) dintre cel mai bun scor și cel de dinaintea celui mai bun scor. $p = 1 - (1/(1 + \delta))$

2.1.4.1.5. Critică

Se folosește de structura “small world” a grafurilor de coocurență.

Pozitive:

- Nu are nevoie de date cu etichete
- Extrage automat o “use list” de cuvinte dintr-un corpus
- Nu se bazează pe sensuri de cuvânt definite de dicționar

Negative:

- Clasificatorul este word-specific
- Un nou clasificator trebuie antrenat pentru fiecare cuvânt ce vrem să îl dezambiguizăm

Acuratete:

- În medie avem o acuratețe de 96%, când este testat pe o lista de 10 cuvinte polisemantice.

2.1.4.2. Algoritmul lui Yarowsky (WSD folosind categoriile de Thesaurus a lui Rogert)

Bazat pe următoarele 3 observații:

- Clasele de cuvinte conceptual diferite (ex. “Animals” și “Machines”) tind să apară contexte diferite.
- Sensuri diferite de cuvânt aparțin unei clase conceptual diferite (ex. “crane”)
- Un discriminator bazat pe context pentru clasele conceptuale poate servi ca și discriminator bazat pe context pentru membrii din aceste clase.

Trebuie să identificăm cuvintele proeminente din contextul colectiv din categoria thesaurus-ului și să adăugăm greutate în mod corespunzător.

$$\text{Weight}(\text{word}) = \text{Salience}(\text{Word}) = \frac{PR(w|RCat)}{PR(w)}$$

ANIMAL/INSECT
species(2.3), family(1.7), bird(2.6), egg(2.2), coat(2.5), female(2.0), eat(2.2), nest(2.5)
TOOLS/MACHINERY
tool(3.1), machine(2.7), engine(2.6), blade(3.8), cute(2.2), saw(2.5), lever(2.0), wheel(2.2), piston(2.5)

Tabel 15. Greutati conform categoriei

2.1.4.2.1. Dezambiguizare

Prezicearea categoriei corecte pentru un cuvânt ambigu folosind greutățile cuvintelor din contextul acestuia.

$$\text{ARGMAX}_{RCat} \sum_{w \text{ in context}} \log\left(\frac{PR(w|RCat) * PR(Rcat)}{PR(w)}\right)$$

Ex: ...lift water and to grind grain. Treadmills attached to *cranes* were used to lift heavy..

TOOLS/MACHINERY	Weight	ANIMAL/INSECT	Weight
lift	2.44	Water	0.76
grain	1.68		
used	1.32		
heavy	1.28		
Treadmills	1.16		
attached	0.58		
grind	0.29		
water	0.11		
TOTAL	11.30	TOTAL	0.76
Tabel 16. Greutati pentru exemplul dat			

2.1.4.2.2. Critică

Pozitive:

- Bazat pe o rețea lexicală (Thesaurus) + Corpus
- Poate captura indiciile date de substantivele proprii din corpus
- Clasificatorul nu este word-specific. Va funcționa și pe cuvinte rare.

Negative:

- Performanța este slabă pentru:
 - Disticții minore de sens dintr-o categorie
Ex: cele două sensuri ale cuvântului “drug” în domeniul medical.
 - Idiomuri
Ex: cuvântul “hand” în “on the other hand” și “close at hand”

Acuratețe:

- În medie avem o acuratețe de 92%, când este testat pe o lista de 12 cuvinte polisemantice.

2.1.4.3. Lin’s Approach

Două cuvinte diferite pot avea înțelesuri similare dacă apar în contexte locale similare.

Ex: The **facility** will **employ** 500 new employees.

Sensul cuvântului “facility”

- **installation**
- proficiency
- adeptness
- readiness
- toilet/bathroom

Subiecții cuvântului “employ”

Word	Freq	Log Likelihood
Organization	64	50.4
Plant	14	31.0
Company	27	28.6
Industry	9	14.6
Unit	9	9.32
Aerospace	2	5.81
Memory device	1	5.79
Pilot	2	5.37

Tabel 17. În acest caz Sensul 1 “installation” este sensul câștigător.

2.1.4.3.1. Similaritate și Hypernymy

$$\text{sim}(A,B) = \frac{\log(P(\text{common}(A,B)))}{\log(P(\text{describe}(A,B)))}$$

Fie A “hill” și B “Coast”, atunci comunalitatea dintre A și B este “A este un GeoForm și B este un GeoForm”.

$$\text{sim}(\text{Hill}, \text{Coast}) = \frac{2 * \log(P(\text{GeoForm}))}{\log(P(\text{Hill})) + \log(P(\text{Coast}))}$$

În general, similaritatea este direct proporțională cu probabilitate că două cuvinte să aibă aceeași super clasa (Hypernym)

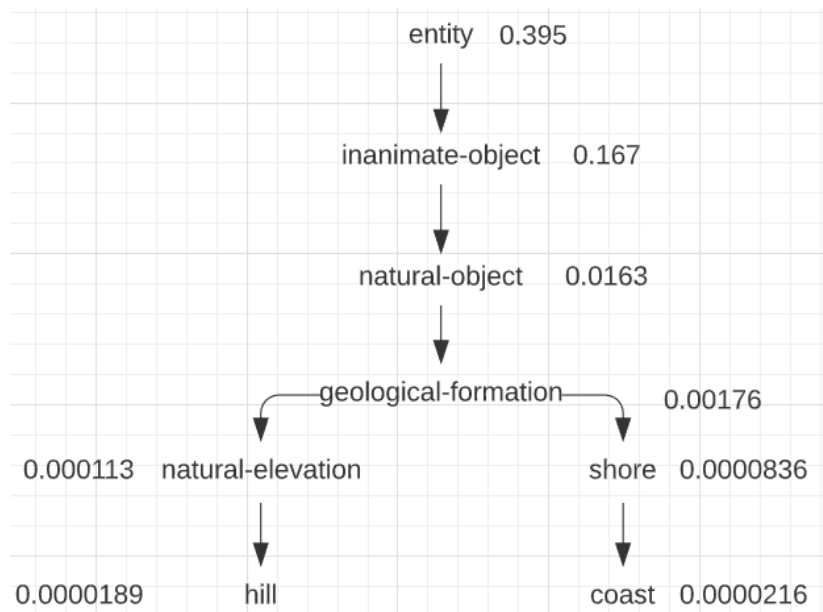


Fig. 15. Pentru a maximiza similaritatea se selectează sensul care are același hypernym ca și majoritatea cuvintelor Selector.

2.1.4.4. WSD folosind Parallel Corpora

Un cuvânt care are mai multe sensuri într-o limba va avea o altă traducere în altă limba, lucru bazat pe contextul în care este folosit. Traducerile pot fi considerate că indicatori contextuali ai sensului de cuvânt.

Model de Sens

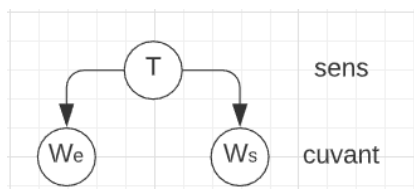


Fig. 16. Modelul de Sens

We fiind un cuvânt în limba engleză și Ws fiind un cuvânt în altă limba.

Model de Concept

$$P(W_e, W_s, T) = P(T).P(W_e|T).P(W_s|T)$$

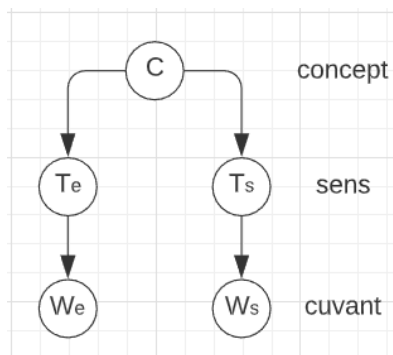


Fig. 17. Modelul de Concept

$$P(W_e, W_s, T_e, T_s, C) = P(C).P(T_e|C).P(T_s|C).P(W_e|T_e).P(W_s|T_s)$$

2.1.4.5. Compararea abordărilor Nesupervizate

Algoritm	Precizie	Recall	Corpus	Baseline
Algoritmul lui Lin	68.5% Rezultatul a fost considerat corect dacă similaritatea dintre sensul prezis și cel actual este mai mare decât 0.27	Ne raportat	Antrenat folosind corpusul WSJ ce conține 25 de milioane de cuvinte. Testat pe 7 fișiere SemCor ce conțin 2832 substantive polisemantice.	64.2%
Hyperlex	97%	82% (cuvinte care nu a fost etichetate confident, thresholdul nu a fost etichetat)	Testat pe un set de 10 cuvinte polisemantice Franceze	73%

WSD folosind categoriile din Thesaurus lui Roget	92% (media de polisemie a fost 3)	Ne raportat	Testat pe un set de 12 cuvinte polisemantice Engleze	Ne raportat
WSD folosind paralel corpora	SM: 62.4% CM:67.2%	SM: 61.6% CM: 65.1%	Antrenat folosind un corpus paralel Englez-Spaniol. Testat folosind Senseval 2-All Words task(doar substantivele au fost folosite)	Ne raportat

Tabel 18. Compararea abordărilor Nesupervizate

2.1.4.6. Abordări Nesupervizate – Concluzie

Generalități:

- Combină avantajele abordării supervizate și a abordării bazate pe knowledge.
- Că și abordările supervizate ele extrag evidente din corpus.
- Că și abordările bazate pe knowledge ele nu au nevoie de un corpus etichetat.

Algoritmul lui Lin:

- O abordare generală cu scopul de a avea acoperire mare.

Hyperlex:

- A fost prima abordare unde se folosesc proprietatile cuvintelor mici pentru a extrage automat evidente din corpus.
- Un clasificator Word-specific.
- Algoritmul ar eșua în deosebirea dintre înțelesurile fine ale unui cuvânt (ex: sensul medicinal și narcotic al cuvântului “drug”)

3. Algoritm

(Articole folosite pentru învățarea NLTK și Python [2] [3] [4] [5] [12])

Indiferent de ce algoritm folosim, ideea principală a dezambiguizării rămâne aceeași. Sensul unui cuvânt este determinat de contextul acestuia. Pentru a arată cât de important este contextul am creat un algoritm ce poate dezambiguiza cuvântul englezesc “bat”.

Fie două înțelesuri pentru acest cuvânt:

1. Înțelesul de bată de cricket, obiect folosit în sportul numit cricket.
2. Înțelesul de liliac, animal nocturn.

Că și date de intrare folosim două fișiere, fișierul 1 conține un text coerent referitor la sensul 1, folosind cuvinte din sfera de influență al acestuia (“sport”, “cricket”), respectiv și fișierul 2 (“nocturnal”, “night”, “fly”).

Pentru a putea realiza acest algoritm am folosit python. Se importă și folosește librăria Natural Language Toolkit (NLTK). O să o folosim pentru date de antrenament, tokenizatori, lematizatori, stemizatori și pentru acces la corpusul Wordnet. NLTK este creat de Universitatea Princeton.

Trebuie să facem distincție între lematizare și stemizare. Stemizarea taie părți din cuvinte pentru a-și îndeplini scopul (începutul sau sfârșitul), folosind o listă de prefixe și sufixe ce pot fi găsite. Această abordare poate fi corectă, dar are și posibilitatea de a eșua.

Forma	Sufixul	Stema
stud ies	-es	studi
stud ying	-ing	study
nin as	-as	nin
nin ez	-ez	nin

Tabel 19. Exemple de stemizare în engleză și spaniolă

Lematizarea folosește un vocabular și analiză morfologică a cuvântului pentru a-și îndeplini scopul, eliminând inflexiuni astfel rezultând formă de bază din dicționar, cunoscută sub denumirea de “lema”. Lemele sunt formele de bază a formelor cu inflecții, stemele nu sunt. De aceea dicționarele sunt liste de leme ci nu steme.

Formă	Informație Morfologică	Lema
stud ies	Persoană a 3, timp prezent al verbului study	study
stud ying	Gerunziul verbului study	study
nin as	Feminin, număr plural al substantivului nino	nino
nin ez	Număr singular al cuvântului ninez	ninez

Tabel 20. Exemple de lematizare în engleză și spaniolă

Funcția “Filtru” primește un query și întoarce o listă de token-uri care sunt lematizați. Această funcție este aplicată și pe inputul userului și pe setul de antrenament. “Stopwords” sunt cuvintele cu frecvență mare într-o limbă, acestea nu contribuie la topicul unei propoziții. În

engleză avem “a”, “an”, “of”, “the”, “at”, etc. Ignorăm aceste cuvinte pentru a ne putea concentra mai bine pe ideea principală.

```
def Filtru(propozitie):  
    propozitie_filtrata = []  
    lematizez = WordNetLemmatizer()  
    StopWords = set(stopwords.words("english"))  
    cuvinte = word_tokenize(propozitie)  
  
    for cuvint in cuvinte:  
        if cuvint not in StopWords:  
            propozitie_filtrata.append(lematizez.lemmatize(cuvint))  
    return propozitie_filtrata
```

Fig. 18. Funcția “Filtru”

Funcția “Verificare_Similaritate” verifică similaritatea dintre propoziția utilizatorului și datele de antrenament. Tokenizăm cuvintele pentru a putea folosi sinonimele din Wordnet. Aici avem două variabile ce vor rezolva ambiguitatea, “depth” și “closeness” ce vor avea rezultate între 0 și 1. Cu cât avem mai multe date de antrenament cu atât mai bun devine algoritmul.

```
def Verificare_Similaritate(cuvant1, cuvant2):  
    cuvant1 = cuvant1 + ".n.01"  
    cuvant2 = cuvant2 + ".n.01"  
    try:  
        cuvant1 = wordnet.synset(cuvant1)  
        cuvant2 = wordnet.synset(cuvant2)  
        return cuvant1.wup_similarity(cuvant2)  
    except:  
        return 0
```

Fig. 19. Funcția “Verificare_Similaritate”

Funcția “Creare_Sinonime” stochează sinonimele cuvântului care o apelează. Sinonime sunt folosite în “Verificare_Similaritate” pentru a afla scorul de similaritate.

```
def Creare_Sinonime(cuvant):  
    sinonime = []  
  
    for sinonim in wordnet.synsets(cuvant):  
        for i in sinonim.lemmas():  
            sinonime.append(i.name())  
  
    return sinonime
```

Fig. 20. Funcția “Creare_Sinonime”

După ce aflăm similaritatea, folosim un alt filtru “Stem_Lem”, pentru lematizarea și stematizarea tokenilor. În lista de propoziții filtrate, vom avea tokenii și sinonimele acestora.

```
def Stem_Lem(propozitie):
    propozitie_filtrata = []
    lematizez = WordNetLemmatizer()
    stematizez = PorterStemmer()

    StopWords = set(stopwords.words("english"))
    cuvinte = word_tokenize(propozitie)

    for cuvant in cuvinte:
        if cuvant not in StopWords:
            propozitie_filtrata.append(lematizez.lemmatize(stematizez.stem(cuvant)))
            for i in Creare_Sinonime(cuvant):
                propozitie_filtrata.append(i)
    return propozitie_filtrata
```

Fig. 21. Funcția “Stem_Lem”

În funcție de ce propoziția dată de utilizator programul va încerca să determine sensul corect al cuvântului “bat” și va întoarce “Obiect” dacă sensul este acela de bată de cricket, iar “Animal” dacă sensul este acela de animal nocturn.

```
Propozitie: bat has a handle
Obiect
Propozitie: bat can fly
Animal
Propozitie: bat that can see
Animal
Propozitie: bat used to play cricket
Obiect
Propozitie: bat that gives birth
Animal
Propozitie: quality of bat
Animal
```

Fig. 22. Rezultatele algoritmului

După cum putem observă algoritmul identifica corect sensul în majoritatea cazurilor, dar în cazul “quality of bat” primim un răspuns eronat deoarece cuvântul quality nu este în setul de antrenament. Cele două fișiere primite ca date de intrare sunt similare cu definiția unui cuvânt, existența într-un vocabular sau Wordnet.

3.1. Interfața grafică

Algoritmul poate fi folosit prin intermediul consolei, dar pentru o accesare usoara am creat o interfața grafică folosind Tkinter in PyCharm (Python)



Fig. 23. Interfața grafică

Dupa ce se introduce propozitia in prima linie, se apasa butonul “Incepe Dezambiguizarea”. Rezultatul poate fi ANIMAL sau OBIECT, putem observa tokenii folositi, lemele si sinonimele, precum si scorul de similaritate si numarul de cuvinte gasite in fisiere.

3.2. Cum putem folosi resursa Wordnet? (exemple practice în Pycharm):

1. Putem obține synset-urile (cuvântului “program”)

```
from nltk.corpus import wordnet  
  
syns = wordnet.synsets("program")  
  
print(syns)
```

Fig. 24. Aflare synset-uri

Rezultat: [Synset('plan.n.01'), Synset('program.n.02'), Synset('broadcast.n.02'), Synset('platform.n.02'), Synset('program.n.05'), Synset('course_of_study.n.01'), Synset('program.n.07'), Synset('program.n.08'), Synset('program.v.01'), Synset('program.v.02')]

2. Putem afla lema (cuvantului “program”)

```
from nltk.corpus import wordnet  
  
syns = wordnet.synsets("program")  
|  
print(syns[0].lemmas())
```

Fig. 25. Aflare leme

Rezultat:[Lemma('plan.n.01.plan'), Lemma('plan.n.01.program'), Lemma('plan.n.01.programme')]

3. Putem afla definiția (“cuvântului program”)

```
from nltk.corpus import wordnet  
  
syns = wordnet.synsets("program")  
  
print(syns[0].definition())
```

Fig. 26. Aflare definiții

Rezultat: A series of steps to be carried out or goals to be accomplished

4. Putem obține propoziții exemplu (pentru “program”)

```
from nltk.corpus import wordnet  
  
syns = wordnet.synsets("program")  
|  
print(syns[0].examples())
```

Fig. 27. Aflare propoziții exemplu

Rezultat: ['they drew up a six-step plan', 'they discussed plans for a new bond issue']

5. Putem obține sinonime și antonime (cuvântului “good”)

```
from nltk.corpus import wordnet

synonyms = []
antonyms = []

for syn in wordnet.synsets("good"):
    for l in syn.lemmas():
        synonyms.append(l.name())
        if l.antonyms():
            antonyms.append(l.antonyms()[0].name())

print(set(synonyms))
print(set(antonyms))
```

Fig. 28. Aflare sinonime și antonime

Rezultat:

Sinonime

{'soundly', 'safe', 'just', 'dear', 'adept', 'commodity', 'salutary', 'ripe', 'well', 'in_force', 'in_effect', 'beneficial', 'dependable', 'thoroughly', 'respectable', 'sound', 'right', 'proficient', 'upright', 'estimable', 'honorable', 'full', 'serious', 'unspoilt', 'skillful', 'undecomposed', 'goodness', 'good', 'practiced', 'skilful', 'secure', 'trade_good', 'near', 'effective', 'unspoiled', 'honest', 'expert'}

Antonime

{'evil', 'ill', 'bad', 'badness', 'evilness'}

6. Putem testa similaritatea dintre cuvinte (“ship”, “boat”)

```
w1 = wordnet.synset("ship.n.01")
w2 = wordnet.synset("boat.n.01")

#similaritatea wu and palmer
print(w1.wup_similarity(w2))
```

Fig. 29. Testarea similarității dintre doua cuvinte

Rezultat: 0.9090909090909091 (acesta se interpretează că un procent, adică a 90% similare)

7. Testăm similaritatea între “ship”-“car”, “ship”-“cat”

```
w1 = wordnet.synset("ship.n.01")
w2 = wordnet.synset("car.n.01")

#similaritatea wu and palmer
print(w1.wup_similarity(w2))

w1 = wordnet.synset("ship.n.01")
w2 = wordnet.synset("cat.n.01")

#similaritatea wu and palmer
print(w1.wup_similarity(w2))
```

Fig. 30. Testarea similarității dintre 3 cuvinte

Rezultat: ship-car 0.6956521739130435

(sunt ambele mijloace de transport de aceea avem o similaritate de 70%)

ship-cat 0.32

4. Istoria pe scurt

Din anii 1940 domeniul WSD a evoluat foarte mult:

- memorandumul din 1949 a lui Warren Weaver ce a introdus ideea de a folosi contextul imediat (cele mai apropiate cuvinte) al unui cuvânt ce trebuie dezambiguit.
- 1960 Bar-Hillel a publicat ideile sale prin care a demonstrat că un computer nu poate dezambigua fără un model de date.
- 1970 WSD este creat un sistem de interpretare semantică prin intermediul inteligenței artificiale, dar în acea vreme sistemele WSD erau bazate strict pe reguli și codate manual, astfel informația ce putea fi folosită avea un volum limitat.
- 1980 au apărut resurse lexicale imense, precum “Oxford Advanced Learner’s Dictionary of Current English” (OALD). Codarea manuală a fost înlocuită cu informații extrase automat din aceste tipuri de resurse, dar dezambiguizarea era bazată tot pe knowledge-based și dictionary-based.
- 1990 revoluția statistică a influențat domeniul de lingvistică computațională, iar WSD a devenit o problema de paradigmă asupra căreia se aplică tehnici supervizate de învățare automată.
- 2000 tehnicile supervizate au ajuns într-un stadiu în care acuratețea devine stagnantă, astfel direcția a trebuit schimbată la adaptarea pe domeniu, sisteme semi-supervizate și nesupervizate. Totuși sistemele supervizate continuă să aibă cea mai bună performanță.

5. Concluzie

În concluzie Word Sense Disambiguation(WSD) se referă la determinarea sensului potrivit pentru un anumit cuvânt dintr-o propoziție folosind contextul în care acesta se află. Există diferite moduri de abordare, fiecare cu avantaje și dezavantaje:

- Abordările Knowledge-based au nevoie de un dicționar cu definiții, de o resursă ca WordNet, lucru foarte greu de creat(este nevoie de mulți ani de muncă pentru crearea unui corpus etichetat), în unele limbi inexistent.
- Abordările Supervizate au nevoie de un dataset etichetat, antrenamentul durează foarte mult.
- Abordările Nesupervizate nu au nevoie de un dataset etichetat, dar nu pot atinge gradul de performanță al celor Supervizate.

Indiferent de metoda aleasă conform variabilelor, precum: cantitatea de materiale disponibile limbii respective, timpul de procesare, etc, se dorește identificarea sensului corect al cuvântului ce trebuie dezambiguit.

Bibliografie

- [1]Lectures in Natural Language Processing – Department of Computer Science & Engineering, IIT Bombay
- [2]WordNet – Natural Language Processing With Python and NLTK
- [3]Lecture 41 – Word Sense Disambiguation – Natural Language Processing - Michigan
- [4]WordNet and Word Sense Disambiguation (WSD) with NLTK
- [5]www.tutorialspoint.com/python
- [6]Queries on the Web - Daniele Braga,Stefano Ceri, Florian Daniel, Davide Martinenghi
- [7]Scalable Query Result Caching for Web Applications - Charles Garrod, Amit Manjhi, Anastasia Ailamaki, Bruce Maggs, Todd Mowry, Christopher Olston, Anthony Tomasic
- [8]UsingWord-Sense Disambiguation Methods to ClassifyWeb Queries by
Intent - Emily Pitler, Ken Church
- [9]Word Sense Disambiguation - Algorithms and Applications - Eneko Agirre
- [10]Personalizing PageRank for Word Sense Disambiguation - Eneko Agirre, Aitor Soroa
- [11]WebTables: Exploring the Power of Tables on the Web - Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, Yang Zhang
- [12]www.tutorialspoint.com/python/python_gui_programming.htm

Anexa

Codul algoritmului de dezambiguizare al cuvântului “Bat”:

Se dau doua fisiere ce contin text explicativ pentru un sens al cuvântului ambiguu cu ajutorul carora se determina sensul cuvântului "bat" din propozitia utilizatorului.

```
import nltk
import codecs
from nltk.tokenize import PunktSentenceTokenizer
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer
from nltk.corpus import wordnet
from tkinter import *
from PIL import ImageTk, Image

root = Tk()

# 1.Se sterg StopWords
# 2.Stemming
# 3.Se fac tokenuri

def Stem_Lem(propozitie):

    propozitie_filtrata = []
    lematizez = WordNetLemmatizer()
    stematizez = PorterStemmer()

    StopWords = set(stopwords.words("english"))
    cuvinte = word_tokenize(propozitie)

    for cuvant in cuvinte:
```

```
    if cuvânt not in StopWords and cuvânt != "bat" and cuvânt != "Bat" and cuvânt != "bats"  
    and cuvânt != "Bats":
```

```
        propozitie_filtrata.append(lematizez.lemmatize(stematizez.stem(cuvânt)))
```

```
    for i in Creare_Sinonime(cuvânt):
```

```
        propozitie_filtrata.append(i)
```

```
    return propozitie_filtrata
```

```
# Aduagam sinonime
```

```
def Creare_Sinonime(cuvânt):
```

```
    sinonime = []
```

```
    for sinonim in wordnet.synsets(cuvânt):
```

```
        for i in sinonim.lemmas():
```

```
            sinonime.append(i.name())
```

```
    return sinonime
```

```
# Verificam similaritatea cuvintelor
```

```
def Verificare_Similaritate(cuvant1, cuvant2):
```

```
    cuvant1 = cuvant1 + ".n.01"
```

```
    cuvant2 = cuvant2 + ".n.01"
```

```
    try:
```

```
        cuvant1 = wordnet.synset(cuvant1)
```

```
        cuvant2 = wordnet.synset(cuvant2)
```

```
    return cuvant1.wup_similarity(cuvant2)
```

```
except:
```

```
    return 0
```

```
def Filtru(propozitie):
```

```
    propozitie_filtrata = []
```

```
    lematizez = WordNetLemmatizer()
```

```
    StopWords = set(stopwords.words("english"))
```

```
    cuvinte = word_tokenize(propozitie)
```

```
    for cuvant in cuvinte:
```

```
        if cuvant not in StopWords and cuvant != "bat" and cuvant != "Bat" and cuvant != "bats"
        and cuvant != "Bats":
```

```
            propozitie_filtrata.append(lematizez.lemmatize(cuvant))
```

```
    return propozitie_filtrata
```

```
def Actionare_Buton_1():
```

```
    Animal_txt = codecs.open("Animal.txt", 'r', 'utf-8')
```

```
    propozitie1 = Animal_txt.read().lower()
```

```
    Obiect_txt = codecs.open("Obiect.txt", 'r', "utf-8")
```

```
    propozitie2 = Obiect_txt.read().lower()
```

```
    propozitie3 = textBox_1.get().lower()
```

```
    propozitie_filtrata1 = []
```

```
    propozitie_filtrata2 = []
```

```
    propozitie_filtrata3 = []
```

```
    count1 = 0;
```

```
    count2 = 0;
```

```
    similaritate_propozitii31 = 0
```

```
    similaritate_propozitii32 = 0
```

```
    propozitie_filtrata1 = Filtru(propozitie1)
```

```
propozitie_filtrata2 = Filtru(propozitie2)
```

```
propozitie_filtrata3 = Filtru(propozitie3)
```

```
textBox_3.config(state="normal")
```

```
textBox_3.delete(0, 'end')
```

```
textBox_3.insert(0, "Tokenii: ")
```

```
textBox_3.insert(9, propozitie_filtrata3)
```

```
textBox_3.config(state="disabled")
```

```
for i in propozitie_filtrata3:
```

```
    for j in propozitie_filtrata1:
```

```
        count1 = count1 + 1
```

```
        similaritate_propozitii31 = similaritate_propozitii31 + Verificare_Similaritate(i, j)
```

```
    for j in propozitie_filtrata2:
```

```
        count2 = count2 + 1
```

```
        similaritate_propozitii32 = similaritate_propozitii32 + Verificare_Similaritate(i, j)
```

```
propozitie_filtrata1 = []
```

```
propozitie_filtrata2 = []
```

```
propozitie_filtrata3 = []
```

```
propozitie_filtrata1 = Stem_Lem(propozitie1)
```

```
propozitie_filtrata2 = Stem_Lem(propozitie2)
```

```
propozitie_filtrata3 = Stem_Lem(propozitie3)
```

```
textBox_4.config(state="normal")
```

```
textBox_4.delete('1.0', END)
```

```
textBox_4.insert(INSERT, "Leme si sinonime: " + "\n" + "\n")
```

```
textBox_4.insert(INSERT, propozitie_filtrata3)
```

```
textBox_4.config(state="disabled")
```

```
prop1_count = 0
```

```
prop2_count = 0
```

```
for i in propozitie_filtrata3:
```

```
    for j in propozitie_filtrata1:
```

```
        if(i == j):
```

```
            prop1_count = prop1_count + 1
```

```
    for j in propozitie_filtrata2:
```

```
        if(i == j):
```

```
            prop2_count = prop2_count + 1
```

```
textBox_5.config(state="normal")
```

```
textBox_5.delete('1.0', END)
```

```
textBox_5.insert(INSERT, "Scor similaritate Animal = " + str(similaritate_propozitii31) + "\n" + "Scor similaritate Obiect = " + str(similaritate_propozitii32))
```

```
textBox_5.config(state="disabled")
```

```
textBox_6.config(state="normal")
```

```
textBox_6.delete('1.0', END)
```

```
textBox_6.insert(INSERT, "Nr cuvinet gasite in fisierul Animal = " + str(prop1_count) + "\n" + "Nr cuvinte gasite in fisierul Obiect = " + str(prop2_count))
```

```
textBox_6.config(state="disabled")
```

```
if((prop1_count + similaritate_propozitii31) > (prop2_count + similaritate_propozitii32)):
```

```
    textBox_2.config(state="normal")
```

```
    textBox_2.delete(0, 'end')
```

```
    textBox_2.insert(0, "Animal")
```

```
    textBox_2.config(state="disabled")
```

```
else:
```

```
    textBox_2.config(state="normal")
```

```
textBox_2.delete(0, 'end')  
textBox_2.insert(0, "Obiect")  
textBox_2.config(state="disabled")
```

```
#print ("\nGATA")
```

```
#Interfata Grafica GUI
```

```
root.title('Dezambiguizarea cuvantului "Bat"')
```

```
#Propozitia de dezambiguizat
```

```
textBox_1 = Entry(root, width=30, font = "Arial 44  
bold", justify="center", bg="#262626", fg="#ff9300", disabledbackground="#262626", disabledfore  
ground="#ff9300", highlightcolor="#ff9300", highlightthickness=3, bd=0)
```

```
textBox_1.grid(row=0, column = 0)
```

```
textBox_1.grid(columnspan=2)
```

```
#Buton Dezambiguizare
```

```
buton_1 = Button(root, text="Incepe Dezambiguizarea", command=Actionare_Buton_1,  
width=20, font = "Times 20  
bold", justify="center", bg="#262626", fg="#ff9300", disabledforeground="#262626", highlightbac  
kground="#262626", highlightcolor="#262626", highlightthickness=5, bd=0, activebackground="#  
262626", activeforeground="red")
```

```
buton_1.grid(row=1, column = 0)
```

```
buton_1.grid(columnspan=2)
```

```
#Rezultatul - Sensul corect - Animal/Obiect
```

```
textBox_2 = Entry(root, width=8, font = "Arial 44  
bold", justify="center", bg="#262626", fg="#ff9300", disabledbackground="#262626", disabledfore  
ground="#ff9300", highlightcolor="#ff9300", highlightthickness=3, bd=0)
```

```
textBox_2.grid(row=2, column=0)
```

```
textBox_2.config(state="disabled")
```

```
#Imagine rezultat
```

```

path = "Animal-Obiect.jpg"
img = ImageTk.PhotoImage(Image.open(path))
panel = Label(root,image = img)
panel.grid(row=3,column=0)

#Prop filtrata
textBox_3 = Entry(root, width=33, font = "Arial 20
bold",justify="center",bg="#262626",fg="#ff9300",disabledbackground="#262626",disabledfore
ground="#ff9300",highlightcolor="#ff9300",highlightthickness=3,bd=0)
textBox_3.grid(row=2,column=1)
textBox_3.config(state="disabled")

#Prop Stem_Lem
textBox_4 = Text(root, height = 10, width = 45, font = "Arial 15
bold",bg="#262626",fg="#ff9300",highlightcolor="#ff9300",highlightthickness=3,bd=0)
textBox_4.grid(row=3,column=1)
textBox_4.config(state="disabled")

#Scor similaritate
textBox_5 = Text(root, height = 2, width = 45, font = "Arial 15
bold",bg="#262626",fg="#ff9300",highlightcolor="#ff9300",highlightthickness=3,bd=0)
textBox_5.grid(row=4,column=0)
textBox_5.config(state="disabled")

#Nr cuv gasite
textBox_6 = Text(root, height = 2, width = 45, font = "Arial 15
bold",bg="#262626",fg="#ff9300",highlightcolor="#ff9300",highlightthickness=3,bd=0)
textBox_6.grid(row=4,column=1)
textBox_6.config(state="disabled")

#spatiu
spatiu = Label(root, height = 1, width = 50)
spatiu.grid(row=5)
spatiu.grid(columnspan=2)
root.mainloop()

```