Universitatea „Lucian Blaga” din Sibiu
Facultatea de Inginerie „Hermann Oberth”
Departamentul de Calculatoare şi Inginerie Electrică

CONTRIBUŢII LA PROIECTAREA SISTEMELOR DE CLASIFICARE A DOCUMENTELOR

Teză de doctorat

Autor:
Asist. univ. Radu George CREŢULESCU, MSc

Conducător ştiinţific:
Prof. univ. dr. ing. Lucian N. VINŢAN, m.c. ASTR

SIBIU, 2011

Cuprins

Partea I. Introducere

What a good thing Adam had. When he said a good thing he knew nobody had said it before.
Mark Twain

The secret of getting ahead is getting started.
Agatha Christie

1 Introducere. Scop şi obiective principale

Majoritatea informaţiilor din lumea reală se găsesc în format text. Aceste date sunt considerate ca având un format semistructurat deoarece conţin puţine metainformaţii despre structura lor. Modelarea şi implementarea de tehnici pentru lucrul cu date semistructurate s-au dezvoltat continuu în ultimii ani. Mai mult decât atât, aplicaţiile de regăsire a informaţiilor ca şi metode de indexare a datelor din baze de date au fost adaptate astfel încât să funcţioneze cu aceste documente nestructurate.

Metodele generale de regăsire a informaţiei nu mai sunt utile pentru căutarea în colecţii mari de date nestructurate sau semistructurate. De obicei, în urma unei interogări utilizatorul obţine puţine rezultate relevante conform cu interogarea formulată asupra documentelor disponibile. Fără anumite cunoştinţe iniţiale referitoare la aceste colecţii mari de date este dificil pentru utilizator să formuleze interogări eficiente pentru extragerea de informaţii „interesante", relevante şi utile pentru utilizator. Pentru a putea compara diferite documente din punct de vedere al gradului de relevanţă şi utilitate precum şi pentru găsirea de reguli pentru organizarea lor devine tot mai imperioasă proiectarea şi utilizarea unor unelte specializate în acest sens.

Machine Learning-ul oferă două abordări privind modalităţile ca o „maşină să înveţe" documente text şi anume, învăţarea supervizată denumită în continuare „clasificare" şi cea nesupervizată denumită în continuare „clustering".

În învăţarea supervizată există două etape distincte: într-o primă fază se alege un set de documente deja preclasificate acest set fiind utilizat apoi ca şi set de antrenament. Pe baza acestei mulţimi de antrenament se va extrage o schemă de clasificare. Această schemă de clasificare este evaluată şi validată utilizând o mulţime de documente de test. În a doua etapă schema de clasificare obţinută este utilizată apoi în clasificarea altor documente din setul de testare. Setul de antrenare (mai mic) este diferit faţă de setul de testare.

Învăţarea nesupervizată constă în aplicarea unor anumiți algoritmi de grupare pentru un set de date dat direct, fără a se specifica criteriul de grupare şi fără a utiliza o etapă de antrenare cum este cazul la învăţarea supervizată. Apoi, într-o a doua etapă, denumită şi validarea clusteringului se aplică diverse măsuri (interne sau externe) pentru a putea analiza corectitudinea clusterelor create.

În prezenta lucrare am plecat de la premisa utilizării metodelor pur computaţionale în regăsirea informaţiilor din documente text. Cu toate că în unele cazuri am încercat adăugarea unei anumite „cantităţi de informaţie semantică" în diverşii algoritmi utilizaţi, abordarea din punct de vedere semantic depăşeşte cadrul prezentei teze.

1.1 *Scop şi obiective*

Scopul general al acestei lucrări este acela de a contribui la îmbunătăţirea performanţelor sistemelor de clasificare şi clustering pentru documente text, prin metode avansate de învăţare nesupervizată şi supervizată.

Pentru a îndeplini acest scop am avut în vedere următoarele aspecte:

- analizarea critică a stadiului actual al regăsirii informaţiilor din documente text în funcţie de metodele de învăţare utilizate: învăţare nesupervizată (clustering) şi respectiv învăţare supervizată (clasificare);
- demonstrarea utilităţii modelului de reprezentare a documentelor bazat pe arbori de sufixe (STDM - Suffix Tree Document Model) în anumiţi algoritmi de clustering;
- elaborarea şi evaluarea unor noi metrici pentru determinarea similarităţii între documente reprezentate cu ajutorul modelului STDM în algoritmi de clustering;
- investigarea punctelor mai „slabe" ale unor metaclasificatori - dezvoltaţi într-o anterioară teză de doctorat elaborată sub conducerea domnului profesor Lucian Vinţan de către domnul ing. Daniel Morariu (2007) - şi determinarea unor soluţii pentru îmbunătăţirea acurateţei de clasificarea a acestora;
- îmbunătăţirea acurateţei de clasificare a documentelor text prin elaborarea unor metaclasificatori hibrizi bazaţi pe algoritmi de clasificare de tip SVM şi Bayes respectiv pe algoritmi genetici şi reţele neuronale pe partea de selecţie a clasificatorului optimal (postclasificare).

1.2 *Structura tezei de doctorat*

Direcţiile de cercetare abordate în prezenta teză sunt prezentate în Fig. 1.1. s-au desfăşurat în mai multe etape, pe două direcţii mari de cercetare: clustering şi clasificare de documente text. În fiecare etapă, informaţiile primite, sunt procesate şi apoi transmise mai departe. Fiecare etapă poate include unul sau mai mulţi algoritmi. Rezultatele fiecărui algoritm utilizat sunt evaluate prin diferite măsuri de validare.

Prezenta lucrare este structurată pe patru părţi, cuprinzând şapte capitole. În prima parte, care conţine capitolele unu şi doi sunt prezentate noţiuni introductive. Astfel, primul capitol prezintă scopul şi obiectivele tezei, o schemă de ansamblu a direcţiilor de cercetare propuse respectiv structura tezei de doctorat.

În capitolul al doilea se prezintă câteva consideraţii generale privind procesarea automată a documentelor de tip text. Sunt trecute în revistă aspecte importante legate de pregătirea datelor - preprocesare, curăţire, transformare şi integrare - în „mineritul datelor" (data mining) în general. Având în vedere scopul general al acestei lucrări sunt prezentate problemele specifice ale „mineritului textului" (text mining) şi ale „mineritului WEB" (WWW mining). De asemenea, se prezintă critic-comparativ aspecte generale legate de învăţarea supervizată şi cea nesupervizată cu implicaţii pentru analiza clasificării şi cea a clusteringului. Se prezintă metricile utilizate de către algoritmi pentru calcularea similarităţii sau disimilarităţii dintre documente. Pentru evaluarea rezultatului algoritmilor de clasificare/clustering sunt prezentate şi măsurile de evaluare. În finalul capitolului se prezintă seturile de date Reuters utilizate pentru algoritmii de clasificare precum şi seturile de date RSS utilizate în algoritmii de clustering.

Partea a doua conţine capitolele trei şi patru şi reprezintă cercetarea autorului în domeniul clusteringului. Astfel, capitolul trei conţine o prezentare originală a unei posibile taxonomii a algoritmilor de clustering. Sunt prezentate diferite tipuri de algoritmi de clustering pentru fiecare

categorie în parte, fiind analizaţi detaliat algoritmii cei mai reprezentativi. Algoritmii HAC (Hierarchical Aglomerative Clustering) şi K-Medoids sunt prezentaţi mai amănunţit, cei doi fiind aleşi pentru efectuarea experimentelor în clusteringul documentelor text.

În capitolul patru sunt prezentate cercetările autorului în domeniul clusteringului. Utilizarea modelului de reprezentare a documentelor ca arbori de sufixe (STDM) în algoritmi de clustering este analizat şi comparat cu modelul clasic vectorial (VSM) de reprezentare a documentelor.

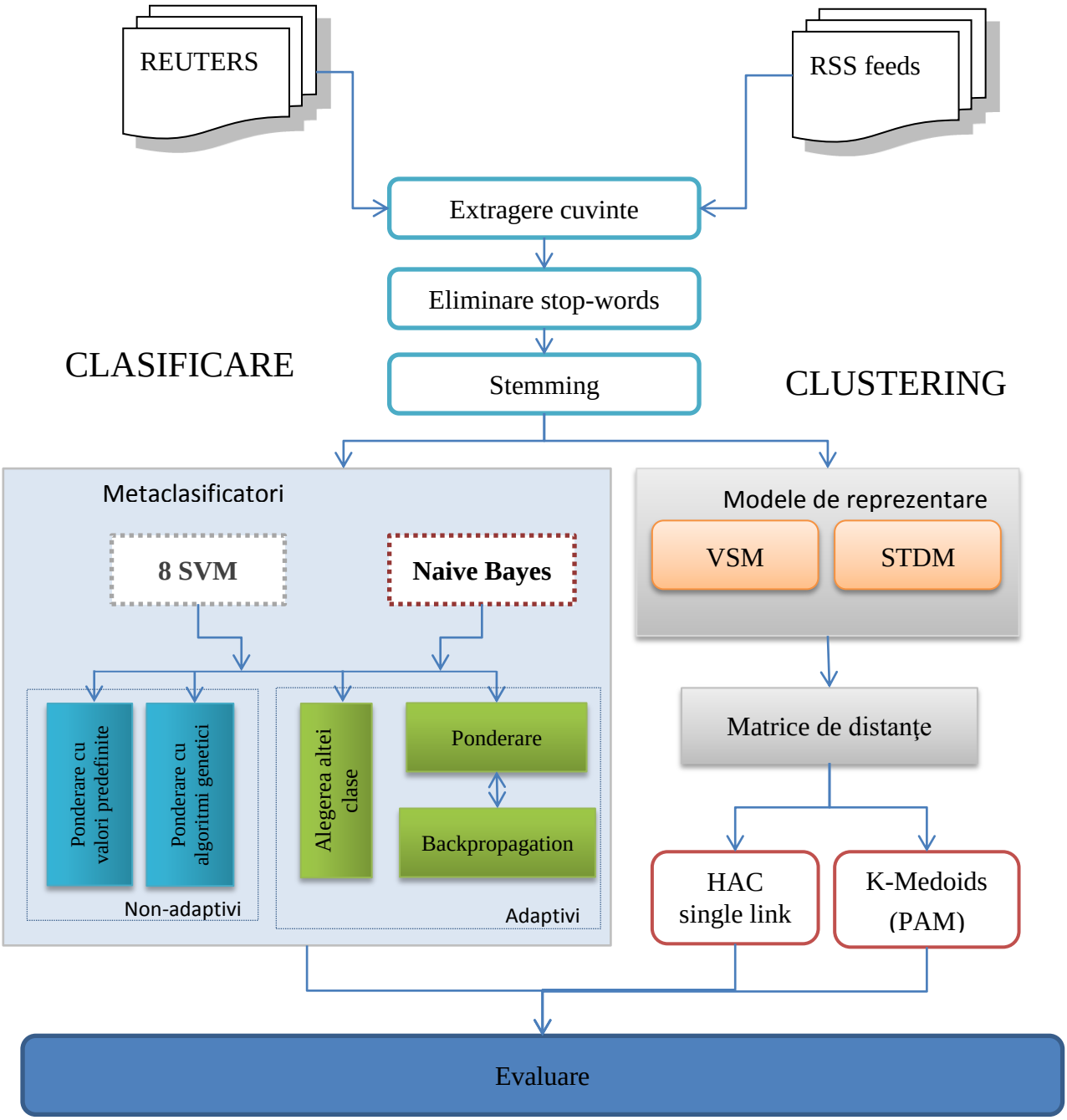


Fig. 1.1 Procesul de extragere a informaţiei din documente text. Abordarea din teză

În acest capitol sunt prezentate metodologia de lucru aplicată la algoritmii de clustering aleşi precum şi rezultatele obţinute de aceştia. Se prezintă metrici pentru calcularea similarităţii

între două documente reprezentate prin modelul STDM, una fiind originală şi cea care obţine rezultatele cele mai bune. În finalul capitolului sunt prezentate şi analizate rezultatele obţinute de algoritmii de clustering utilizând diferite metrici şi cele două modele de reprezentare a documentelor – STDM şi VSM.

În partea a treia, care conţine capitolele cinci şi şase se prezintă cercetarea efectuată în domeniul clasificării documentelor text. În capitolul cinci se face o prezentare a unei posibile taxonomii pentru algoritmii de clasificare. Sunt prezentaţi algoritmi de clasificare caracteristici pentru fiecare categorie şi unele rezultate din faza de testare a acestora. De asemenea, la finalul capitolului cinci sunt prezentate noţiuni esenţiale legate de metaclasificatoare.

În capitolul şase se prezintă cercetări privind proiectarea metaclasificatoarelor. Sunt propuse unele soluţii de îmbunătăţire a unor metaclasificatori existenţi şi se prezintă 3 metaclasificatori originali. Utilizarea unor algoritmi genetici şi a unei reţele neuronale au dus la îmbunătăţirea rezultatelor clasificării documentelor text care sunt prezentate în cadrul acestui capitol.

Prezenta teză se încheie cu capitolul şapte care este dedicat prezentării ideilor principale care se desprind din aspectele teoretice şi practice ale cercetărilor efectuate şi care sintetizează contribuţiile personale aduse în această lucrare precum şi perspectivele de cercetare.

Mulţumiri

În primul rând cele mai sincere mulţumiri doresc să le adresez conducătorului meu de doctorat dl. prof. univ. dr. ing. Lucian N. VINŢAN, pentru coordonarea ştiinţifică riguroasă, pentru încrederea acordată şi pentru discuţiile profesionale extrem de stimulative şi interesante pe care le-am avut precum, pentru exigenţa manifestată faţă de lucrare.

De asemenea, doresc să mulţumesc şi pe această cale colegilor din Catedra de Calculatoare şi Automatizări din cadrul Universităţii „Lucian Blaga” din Sibiu, în special domnului şl. dr. ing. Daniel Morariu, pentru faptul că a acceptat să-i continui munca şi pentru atmosfera plăcută creată, care a contribuit şi ea semnificativ la motivarea mea.

Doresc să mulţumesc referenţilor acestei teze de doctorat, pentru acceptul şi bunăvoinţa de a recenza această lucrare şi pentru efortul competent depus.

Totodată mulţumirile mele se îndreaptă spre familia şi prietenii mei pentru susţinerea morală şi pentru faptul că nu au renunţat să creadă în mine, cu toate că timpul care l-am alocat lor a fost redus considerabil.

Este interesant faptul că, această lucrare a început o dată cu căsătoria mea, a asistat la naşterea fiicei mele Ana, a însoţit-o în prima zi de grădiniţă când nu vorbea, a fost alături când Ana a început să vorbească limba germană, să schieze şi să înoate. Sper că prima zi de şcoală a Anei va reprezenta actul final pentru această teză şi totodată ieşirea mea ca cercetător din rândurile „grădiniţei” şi intrarea în „şcoală”.

Information is not knowledge.
Albert Einstein

2 Stadiul actual în procesarea automată a documentelor de tip text

2.1 *Data mining în baze de date*

Data mining [Han01] reprezintă extragerea cunoştinţelor din masive de date. Este o prescurtare pentru „mineritul cunoştinţelor din date" (knowledge mining from data). Data mining este un simplu pas în procesul complex de descoperire a cunoştinţelor din baze de date. Acest pas este folosit în special pentru a analiza datele conţinute în baze de date. Această analiză are ca rezultat descoperirea caracteristicilor ce pot fi folosite ulterior la organizarea datelor. Astfel, data mining poate fi privit ca procesul de descoperire a pattern-urilor şi a legăturilor (corelaţiilor) dintre date (tehnici de acest fel sunt prezentate în [Ian00]). Acest proces trebuie să fie automat sau semi-automat. Deci, data mining este procesul de descoperire a cunoştinţelor relevante în cantităţi mari de date organizate în baze de date, depozite de date sau alte magazii de informaţii (data warehouse). Un sistem de data mining trebuie să aibă următoarele componente majore [Han01]:

1. *Baze de date, depozite de date sau orice alt depozit de informaţii* – cuprinde curăţirea datelor (eliminarea zgomotului şi corectarea inconsistenţelor din date) şi tehnicile de integrare (agregare) efectuate pe date.
2. *Baze de date sau servere pentru depozitarea datelor* – sunt responsabile pentru aducerea datelor relevante bazate pe cererea utilizatorului.
3. *Cunoştinţele de bază* – pentru ghidarea căutării sau evaluarea pattern-urilor rezultate.
4. *Motorul data mining* – este esenţial pentru sistemul de data mining şi ideal ar trebui să cuprindă module funcţionale pentru sarcini cum ar fi, asocierile, clasificările, analiza clusterelor şi analiza evoluţiei şi derivării.
5. *Module de evaluare a pattern-urilor* – sunt folosite pentru a interacţiona cu modulul de data mining pentru a realiza o căutare a pattern-urilor relevante.
6. *Interfaţa grafică a utilizatorului* – comunicarea între utilizator şi sistemul de data mining pentru specificarea interogărilor de data mining şi afişarea informaţiilor obţinute.

Aşadar, data mining este considerată a fi unul dintre cele mai importante domenii în sistemul de baze de date şi unul în care se prevăd dezvoltări interesante şi promiţătoare în industria informatică.

2.1.1 Preprocesarea datelor

Bazele de date din lumea reală sunt astăzi foarte sensibile la zgomot, la lipsuri şi inconsistenţă a datelor. Pentru a putea obţine rezultate folositoare din date se folosesc următoarele tehnici de preprocesare:

- *Curăţirea datelor* – pentru eliminarea zgomotului şi corectarea inconsistenţelor din date;

- *Agregarea datelor* – contopirea datelor din surse multiple într-o formă corespunzătore mineritului. Combină datele din mai multe surse într-un depozit de date coerent: depozit de date sau cub de date;
- *Selectarea datelor* – pentru a selecta informaţiile relevante pentru procesul curent de analiză cu scopul simplificării muncii în etapa propriu-zisă de extragere de cunoştinţe;
- *Transformarea datelor* – pregătirea datelor pentru analiză printr-o reprezentare cât mai adecvată. Constă în operaţii cum ar fi normalizarea datelor. Normalizarea poate îmbunătăţi acurateţa şi eficienţa algoritmilor de data mining. Normalizarea datelor reprezintă scalarea tuturor datelor la un domeniu prestabilit;
- *Reducerea datelor* – poate reduce dimensiunea datelor prin agregare, eliminarea trăsăturilor caracteristice redundante sau gruparea trăsăturilor comune.

Aceste tehnici sunt aplicate apriori procesului de data mining şi pot îmbunătăţi calitatea procesului pentru găsirea şabloanelor şi/sau reducerea timpului necesar pentru mineritul efectiv.

2.1.2 Curăţirea datelor

Deoarece datele pentru procesul de data mining sunt preluate din diverse surse, acestea au deseori diverse structuri şi valori eronate sau lipsă. Algoritmii de curăţire a datelor încearcă să „umple” *valorile lipsă*, să netezească valorile considerate *zgomot* prin identificarea extremelor şi să corecteze *inconsistenţele în date*.

2.1.2.1 Valorile lipsă

Atunci când se aduc toate datele în aceeaşi structură pot apărea valori lipsă în valorile corespunzătoare unui document. Pentru a umple valorile lipsă există câteva metode clasice:

- *Ignorarea tuplelor* (valorile atributelor corespunzătoare unui document) – de obicei este făcută când clasa de etichete sau o mare parte din atributele tuplului lipsesc. Această metodă nu este eficientă mai ales în cazul în care tuplele conţin câteva atribute cu valori lipsă. Este în special slabă când procentajul valorilor lipsă per atribute variază considerabil;
- *Umplerea valorilor lipsă manual* – în general această cale este consumatoare de timp şi nu este fezabilă la multe valori lipsă;
- *Utilizarea unei constante globale pentru umplerea valorilor lipsă* – înlocuieşte toate atributele lipsă cu aceeaşi constantă (ex. „Unknown” sau $-\infty$). Această metodă este simplă dar nu este recomandată deoarece poate duce la valori eronate ale tuplului;
- *Utilizarea unui atribut de mijloc pentru înlocuirea valorilor lipsă* – de exemplu se umple cu media tuturor valorilor din câmpul respectiv;
- *Utilizarea unui atribut de mijloc pentru toate eşantioanele aparţinând aceleiaşi clase din care provine tuplul* – Aceasta se aplică în cazul în care avem la dispoziţie şi un set mic de date preetichetate. Se vor înlocui valorile lipsă în setul de datete neetichetat cu media valorilor pe tuplul respectiv pentru toate documentele aparținând aceleiași clase din setul de date etichetat;
- *Utilizarea valorii cele mai probabile pentru înlocuirea valorilor lipsă* – determinarea prin regresie a valorilor, are ca rezultat componente bazate pe concluzii utilizând teoria Bayesiană sau bazat pe arbori de decizie.

Ultima metodă este cea mai des folosită pentru că utilizează datele prezente pentru a prezice valorile lipsă.

2.1.2.2 Zgomotul

Zgomotul este o eroare aleatoare sau variabilă în valorile măsurate. Acesta poate duce la erori în evaluarea pattern-urilor. Câteva tehnici de netezire a datelor (eliminare a zgomotului) sunt:

1. ***Filtrarea*** – reprezintă netezirea valorilor sortate prin consultarea vecinătăţii. Valorile sortate sunt distribuite într-un număr de „grupuri" sau *bin (container).* Deoarece metoda consultă valoarea vecinului, ea va face o netezire locală. Există mai multe metode de netezirea a datelor:
   - *Netezire prin filtru median* **–** fiecare element din container este înlocuit cu media elementelor din acel grup.
   - *Netezirea prin mediană* **–** fiecare element este înlocuit cu mediana elementelor din container.
   - *Netezirea prin margini* – minimul şi maximul sunt date de *container* şi reprezintă graniţele, fiecare element este înlocuit de valoarea graniţei celei mai apropiate.
2. ***Clusterarea*** – extremităţile pot fi găsite prin grupare, unde valorile similare sunt organizate într-un grup sau „cluster". Valorile care rămân afară după grupare sunt considerate zgomot şi sunt eliminate.
3. ***Combinarea inspecţiei umane cu cea a calculatorului*** – aplicaţia este utilizată pentru identificarea pattern-urilor extreme care reflectă un conţinut „surpriză" şi respectă o etichetă cunoscută. Pattern-urile găsite pot fi bune sau pot fi „gunoi". Omul poate sorta aceste pattern-uri pentru a identifica care dintre acestea reprezintă gunoi.
4. ***Revenirea la starea anterioară*** – datele pot fi netezite prin ajustarea datelor. Regresia liniară implică găsirea celei mai bune linii pentru a uni două variabile astfel încât variabilele cunoscute pot fi utilizate să predicţioneze altele. Regresia multiplă liniară este o extensie a regresiei liniare unde mai mult de două variabile sunt implicate şi datele sunt unite pe o suprafaţă multidimensională. Se utilizează regresia pentru a găsi ecuaţia matematică care ajută la netezirea datelor şi elimină zgomotul.

2.1.3 Transformarea şi integrarea datelor

Data mining oferă posibilitatea de *integrare a datelor* – concatenarea datelor din surse multiple de date. Aceste date trebuie *transformate* într-o formă convenabilă pentru căutare.

2.1.3.1 Integrarea datelor

Este inevitabil ca în etapa de analiză a datelor, să se apeleze şi la *integrarea datelor* din surse multiple într-o sursă de date coerentă – aceasta reprezintă depozitarea datelor. Există un număr de probleme de luat în considerare pe durata integrării datelor:

- *Identificarea entităţilor* – cum pot şti că *customer_id* dintr-o bază de date şi *cust_number* din cealaltă fac referire la aceleaşi entităţi. Bazele de date şi depozitele de date conţin de obicei metadate. Fiecare metadată poate fi folosită pentru a evita erori în schema de integrare.
- *Redundanţa* – un atribut poate fi redundant dacă el este „derivat" dintr-o altă tabelă. Inconsistenţa în atribute sau dimensiune pot de altfel cauza redundanţă în setul de date. Redundanţa poate fi eliminată prin *analiza corelaţiilor*. Analiza măsoară cât de puternic un atribut implică pe celălalt, bazându-se pe datele disponibile. Corelaţia dintre atributele ***A*** şi ***B*** poate fi măsurată prin [Han01]**:**

$$r_{A,B} = \frac{\sum (A-\overline{A})(B-\overline{B})}{(n-1)\sigma_A \sigma_B} \qquad (2.1)$$

unde n reprezintă numărul de tuple, $\overline{A}$ şi $\overline{B}$ sunt media valorilor ***A*** şi ***B*** iar σ_A şi σ_B reprezintă deviaţia standard:

$$\overline{A} = \frac{\sum A}{n}, \overline{B} = \frac{\sum B}{n}, iar: \sigma_A = \sqrt{\frac{\sum (A-\overline{A})^2}{n-1}}, \sigma_B = \sqrt{\frac{\sum (B-\overline{B})^2}{n-1}} \qquad (2.2)$$

Dacă rezultatul relaţiei (2.1) este mai mare decât zero atunci ***A*** şi ***B*** sunt posibil corelate, în sensul că dacă valoarea lui ***A*** creşte va creşte şi valoarea lui ***B***. De aici, o valoare mare va indica că ***A*** sau ***B*** pot fi şterse deoarece sunt redundante. Dacă rezultatul este zero ***A*** şi ***B*** sunt independente şi nu există corelaţie între ele. Dacă valoarea rezultată este mai mică ca 0, atunci ***A*** şi ***B*** au o corelaţie negativă (anticorelaţie) unde dacă valoarea unui atribut creşte valoarea celuilalt atribut va descreşte.

- Detecţia şi rezoluţia valorii datelor de conflict – valorile pot diferi din cauza scării de reprezentare sau codării. De exemplu înălţimea poate fi memorată în diferite unităţi de măsura. Preţul pentru diferite hoteluri poate implica nu doar diferenţe de preţ ci şi diferenţe de servicii.

Integrarea cu grijă a datelor din surse multiple poate ajuta la reducerea şi evitarea redundanţelor şi inconsistenţelor în setul de date rezultat şi poate duce la o îmbunătăţire a procesului de extragere de cunoştinţe.

2.1.3.2 Transformarea datelor

Transformarea datelor constă în reprezentarea sau consolidarea datelor într-o formă convenabilă pentru mineritul datelor. Ea implică următoarele:

- *Netezirea* – se foloseşte pentru eliminare zgomotului din date. Asemenea tehnici includ filtrare, clusterare şi regresie;
- *Agregarea* – operaţii de sumarizare sau agregare sunt aplicate pe date. De exemplu vânzările zilnice pot fi agregate ca şi calcule lunare sau anuale;
- *Generalizarea* – unde primitivele de date de nivel jos sunt înlocuite cu primitive de nivel înalt. De exemplu atributul *stradă* poate fi generalizat cu *oraş* sau *ţară*. La fel *vârsta* poate fi generalizată în *tânăr, mediu şi bătrân*;
- *Normalizarea* – se aplică o scalare într-un anumit domeniu pentru atribute. De exemplu – datele sunt scalate în intervalul [-1.0, 1.0] sau în intervalul [0.0, 1.0].
- *Construirea atributelor* (construirea articolelor) – unde noi atribute sunt create şi adăugate în setul de atribute dat pentru procesul de mining.

În continuare, nu voi detalia tehnicile de netezire, agregare şi generalizare, ci mă voi axa pe normalizare pentru că a fost utilizată cu precădere în experimentele prezentate în această teză.

Normalizarea – reprezintă scalarea unei valori a atributului într-un domeniu specificat şi se realizează printr-o funcţie matematică. Dintre metodele de normalizare mai des utilizate, se menţionează următoarele:

- *Normalizarea min-max*: transformare liniară de la datele originale. Presupunem min_A şi max_A sunt valorile minime şi maxime a atributului. Normalizarea valorii v a lui ***A*** în v' în noul domeniu [new_min_A, new_max_A] se face după formula:

$$v` = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A . \qquad (2.3)$$

Normalizarea min-max păstrează relaţiile dintre valorile datelor originale.

- *Normalizarea z-mediu* – valoarea A este normalizată în baza unei medii şi deviaţii standard a lui A: $v` = \frac{v - \bar{A}}{\sigma_A}$, unde v este valoarea lui A, v' este valoarea calculată, $\bar{A}$ este media iar σ_A este deviaţia standard. Această metodă de normalizare este utilizată când minimul şi maximul sunt necunoscute sau A iese din domeniul min-max.
- *Normalizare prin scalare decimală* – normalizare prin mutarea punctului decimal, punct care se mută ţinând cont de valoarea maximă absolută a lui A: $v' = \frac{v}{10^j}$ unde j este un întreg dat de $\max(v') < 1$. De exemplu pentru A în domeniul [–986; 917], j ia valoarea 3.

Normalizarea schimbă datele originale în totalitate, în special ultimele două metode. Este necesar să se salveze parametrii de normalizare pentru ca noile date să se normalizeze în aceiaşi măsură.

Construirea atributelor – noile atribute sunt construite şi adăugate plecând de la atributele deja existente pentru a ajuta la o mai bună reprezentare a setului de date. De exemplu din atributele „lungime" şi „lăţime" s-ar putea construi atributul „arie".

2.2 *Text mining*

Cele mai multe colecţii de date din lumea reală sunt în format text şi constau din colecţii mari de documente din diferite surse, cum ar fi articole de ştiri, lucrări de cercetare, cărţi, biblioteci digitale, mesaje e-mail, pagini web şi fluxuri RSS etc. Aceste tipuri de date sunt semistructurate şi nu au o structură complet deterministă. Au fost dezvoltate tehnici de regăsire a informaţiilor, cum ar fi metodele de indexare a textului, care au fost dezvoltate pentru a manevra documente nestructurate.

Există mai multe abordări în text mining, care pot fi clasificate din perspective diferite, unele bazate pe intrări din cadrul sistemului de text mining şi unele bazate pe sarcinile care urmează să fie efectuate de sistemul de minerit. În general, abordările majore, sunt [Han01]: (1) abordarea bazate pe cuvinte cheie, în cazul în care ca şi intrare este un set de cuvinte cheie sau termeni din documente, (2) abordarea bazată pe etichetare, în cazul în care datele de intrare sunt reprezentate printr-un set de etichete şi (3) abordări bazate pe extragerea de informaţii, abordare care are ca intrare informaţii semantice, cum ar fi evenimente, fapte, sau alte entităţii descoperite prin extragerea de informaţii. O simplă abordare bazată pe cuvinte cheie ar putea să descopere doar relaţii la un nivel relativ superficial, cum ar fi redescoperirea expresiilor simple (de exemplu, "baza de date" şi "sisteme") sau apariţii împreună în pattern-uri cu semnificaţie puţin importantă (de exemplu, "pescar" şi "peşte"), dar care nu pot duce la o înţelegere profunda a textului. Metoda etichetării documentelor se bazează pe denumiri de clase, obţinute prin etichetare manuală (care este costisitoare şi este imposibil de realizat pentru colecţii mari de documente) sau de către unii algoritmi de clasificare automată (care pot prelucra un set relativ mic de clase dar au nevoie în prealabil de definirea etichetelor). Cea de a treia abordare şi anume, extragerea de informaţii este mult mai avansată şi poate duce la descoperirea unor cunoştinţe profunde, dar ea necesită o analiză semantică a textului prin înţelegerea limbajului natural utilizând atât metode de învăţare automată cât şi lingvistica computaţională.

Cele trei abordări prezentate sunt rezolvate prin diferite metode. Acestea includ clusteringul documentelor, clasificarea, extragerea de informaţii şi analiza asocierilor. În prezenta lucrare voi prezenta câteva aspecte şi cercetări proprii privind clusteringul şi clasificarea de documente text.

2.2.1 Analiza datelor text şi regăsirea informaţiei

Un domeniu care a fost dezvoltat o dată cu sistemele de baze de date este regăsirea informaţiei (Information Retrieval - IR). Faţă de un sistem de baze de date care se axează pe interogări şi procesarea tranzacţiilor în date structurate, regăsirea informaţiei este concentrată pe organizarea şi recunoaşterea de informaţii din documente text. Pentru sistemele de regăsire a informaţiei, o problemă importantă o reprezintă alegerea documentelor relevante pentru utilizator.

Deoarece sistemele de regăsire a informaţiilor şi sistemele de baze de date utilizează fiecare diferite tipuri de date, problemele care apar la aceste două tipuri de sisteme sunt diferite. Problemele majore ale regăsirii informaţiei din documente text sunt legate de aproximarea căutării bazându-ne pe cuvinte cheie şi pe noţiunea de relevanţă.

2.2.1.1 Măsuri de bază pentru regăsirea informaţiei

Să presupunem că un sistem de regăsire a informaţiei a returnat un număr de documente pe baza unei interogări formulate. Problema care apare este cum se poate aprecia dacă setul de documente regăsit pe baza interogării este satisfăcător.

Prezentăm în continuare două notaţii importante pentru sistemele de regăsire a informaţiei [Han01]. Fie $\{Relevant\}$ mulţimea tuturor documentele relevante faţă de interogare şi $\{Regasit\}$ mulţimea tuturor documentelor găsite în urma interogării. Setul de documente care le include pe ambele este notat $\{Relevant\} \cap \{Regasit\}$. Există 3 măsuri de bază pentru aprecierea calităţii textului regăsit:

- **Precizie** – reprezintă probabilitatea ca un document clasificat corect să fie util:

$$precision = \frac{\{Relevant\} \cap \{Regasit\}}{\{Regasit\}} \tag{2.4}$$

- **Recall** – reprezintă probabilitatea ca un document util să fie clasificat corect

$$recall = \frac{\{\text{Relevant}\} \cap \{\text{Regasit}\}}{\{\text{Relevant}\}} \tag{2.5}$$

- **F-measure** care este definit ca medie armonică între Precizie şi Recall

$$F - measure = \frac{recall \times precision}{(recall + precision)/2} \tag{2.6}$$

Este ştiut faptul că media armonică penalizează mai bine decât alte măsuri (ex. media aritmetica) un sistem care sacrifică (supralicitează) o măsură în favoarea celeilalte.

2.2.2 Metode de regăsire a informaţiei

În general, metodele de regăsire se împart în două categorii: metode care văd regăsirea de informaţie ca o **problemă de selecţie** a unor documente şi metode care oferă o **ierarhie de documente**. Această ierarhie se bazează pe un scor obţinut de fiecare document în funcţie de cât de similare sunt acestea faţă de interogarea formulată.

În metodele de selecţie a documentelor, interogarea stabileşte anumite limite pentru selectarea documentelor relevante. O metodă tipică din această categorie este modelul Boolean de regăsire a informaţiei în care un document este reprezentat ca un set de cuvinte cheie şi oferă unui utilizator o expresie booleană de cuvinte cheie, cum ar fi "bed AND breakfast", "tea OR coffee", sau "browser web NOT Internet Explorer". Un astfel de sistem de regăsire a informaţiei

va executa o astfel de interogare şi va returna doar documentele care satisfac expresia booleană. Totuşi, metoda de regăsire booleană, în general, funcţionează bine doar atunci când utilizatorul cunoaşte setul de date din care vrea să selecteze informaţii şi poate formula o interogare booleană corectă în acest sens.

Metodele care oferă o ierarhie a documentelor, ordonează documentele în funcţie de relevanţa acestora în raport cu o interogare formulată. Pentru utilizatorii obişnuiţi, aceste metode sunt mai adecvate decât metodele de selecţie a documentelor. Cele mai multe sisteme moderne de regăsire a informaţiilor oferă o listă de documente ierarhizate, ca răspuns la interogarea bazată pe cuvinte sau expresii cheie a unui utilizator. Scopul acestei metode este de a aproxima gradul de relevanţă a unui document cu interogarea formulată, cu ajutorul unui scor calculat pe baza informaţiilor, cum ar fi frecvenţa de cuvinte din document şi cuvintele din întregul set. Este evident faptul că este dificil să găsim o măsură precisă a gradului de relevanţă pentru un set de cuvinte cheie.

Cea mai des utilizată abordare pentru reprezentarea documentului este utilizarea modelului spaţiului vectorial (Vector Space Model - VSM). Ideea de bază a modelului VSM este următoarea: reprezentăm un document şi o interogare, ca vectori într-un spaţiu *n*-dimensional corespunzător pentru toate cuvintele cheie şi vom folosi o măsură de similaritate specifică pentru a calcula similaritatea dintre vectorul de interogare şi vectorul documentului. Valorile similarităţii pot fi apoi utilizate pentru ierarhizarea documentelor. Pornind de la un set *D* de documente şi *t* termeni (cuvinte), putem modela fiecare document ca fiind un vector *v* într-un spaţiu *n* dimensional ortogonal R^n. Coordonata *j* a lui *v* este un număr (frecvenţă de termeni – care se poate şi pondera) care măsoară importanţa termenului *j* pentru documentul dat: este de obicei 0 dacă documentul nu conţine termenul şi diferit de zero în rest. Presupunând acum că la documente similare se aşteaptă să aibă o frecvenţă similară a termenilor putem măsura similaritatea prin compararea frecvenţelor cuvintelor de bază. De exemplu folosind formula cosinusului dintre doi vectori n-dimensionali:

$$sim(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\| \|v_2\|} \qquad (2.7)$$

unde $v_1 \cdot v_2$ reprezintă produsul scalar a doi vectori şi se calculează ca fiind $\sum_{i=1}^{n} v_{1i} v_{2i}$ (cosinusul între versorii ortogonali este evident 0) iar $\|v_1\|$ este norma (modulul, măsura) vectorului şi se calculează ca fiind $\|v_1\| = \sqrt{v_1 \cdot v_1}$. Evident ca *sim (v1,v2)=1* semnifică o similaritate maximă între cei doi vectori.

2.2.3 **Asocierea între cuvinte cheie şi clasificarea documentelor**

O astfel de analiză colectează seturi de cuvinte cheie sau termeni care apar frecvent împreună şi apoi stabileşte relaţii de asociere sau de corespondenţă între ele. Metoda analizei asocierilor în baze de date text efectuează câteva operaţii de preprocesare a datelor, prin extragerea rădăcinii cuvintelor şi prin eliminarea cuvintelor de „legătură” din text obţinând astfel un set „redus”, apoi apelează algoritmi specifici mineritului pe bază de asocieri. Într-o bază de date cuprinzând documente, fiecare document poate fi privit ca o tranzacţie, în timp ce un set de cuvinte cheie din document poate fi considerat ca un set de elemente din acea tranzacţie. Baza de date este în formatul:

{*id_document, lista_cuvinte_cheie*} (2.8)

Problema mineritului asocierii cuvintelor cheie din documente este, în consecinţă, mineritul asocierii înregistrărilor din bazele de date tranzacţionale, unde au fost dezvoltate mai multe metode [Agra93], [Hols94]. Setul de cuvinte cheie care în mod consecutiv apar frecvent

pot forma un termen sau o frază. Procesul de minerit al asocierilor poate ajuta la detectarea *componentelor asocierii*, adică termeni sau fraze dependente de domeniu (ex. Stanford, University), sau *asocierilor necombinate* (RON, curs valutar). Mineritul bazat pe aceste asocieri este numit „mineritul asocierilor de termeni" care este diferit faţă de mineritul cuvintelor simple. Mineritul asocierilor la nivelului termenului se bucură de două avantaje în analiza textului: (1) termenii şi frazele sunt automat etichetate eliminându-se factorul uman şi (2) numărul de rezultate fără sens este substanţial redus în timpul execuţiei algoritmului de minerit.

2.2.4 **Alte tehnici de indexare pentru regăsirea textului**

Alte tehnici de regăsire pe care le-am analizat se referă la *index invers* şi *fişiere de semnături.*

Un *index invers* este o structură de două tabele numite *document_table* şi *term_table* indexate cu tabele HASH sau cu ajutorul unui arbore B+:

- ***document_table*** – constă dintr-un set de înregistrări, fiecare conţinând două câmpuri: *doc_id* şi *posting_list*, unde *posting_list* este o listă de termeni care apar în document, sortată în raport cu metrica utilizată;
- ***term_table*** – constă dintr-un set de termeni înregistraţi, fiecare conţinând: *term_id* şi *posting_list*, unde *posting_list* specifică o listă de identificatori de documente în care apar termenii.

Fişierele de semnătură sunt fişiere care memorează semnături ale înregistrărilor pentru fiecare document din baza de date. Fiecare semnătură are o dimensiune fixă de *b* biţi care reprezintă termenii. Fiecare bit al documentului de semnături este setat la 0 la început. Un bit este trecut pe 1 dacă termenul reprezentat prin semnătură apare în document. O semnătură S1 se potriveşte cu o altă semnătură S2 dacă fiecare bit care este setat în S1 este setat şi în S2.

2.3 *WWW mining*

WWW (World Wide Web)este un sistem hipermedia distribuit, oferind o gamă largă de informaţii şi servicii. Informaţiile conţinute în paginile WEB precum şi legăturile (linkurile) între ele oferă surse importante pentru data mining. În continuare prezint unele probleme cheie care vizează utilizarea WWW ca sursă pentru data mining.

- Web-ul pare a fi prea vast pentru realizarea unui data warehousing sau data mining eficient. Dimensiunea actuală a WEB-ului este de ordinul sutelor sau miilor de terabytes şi este în continuă creştere rapidă. La sfârşitul lunii iunie 2011 compania Netcraft [NETK] – o companie care monitorizează severe de web – a raportat răspunsuri de la un număr de **346.004.403** de site-uri.
- Complexitatea paginilor WEB este de departe mai mare decât orice altă colecţie de documente text tradiţionale. În paginile web lipseşte standardizarea. În ultimii 2-3 ani de când ideea de web semantic prinde tot mai mult contur se încearcă introducerea unor standarde la nivel de limbaj şi structură a paginii. În momentul actual, paginile de web conţin structuri şi stiluri diferite. Chiar dacă WEB-ul este considerat a fi o mare bibliotecă digitală, majoritatea documentelor din această bibliotecă nu sunt ordonate. Nu există indexare după categorii, titlu sau autor. Căutarea şi regăsirea de informaţii într-o asemenea bibliotecă este o sarcină importantă şi dificilă.
- Web-ul este o sursă de informaţii foarte dinamică. Web-ul creşte continuu şi informaţiile conţinute sunt constant actualizate. Ştirile, bursa de valori, reclame şi reclame pentru servicii web modifică paginile web zilnic.

- Web-ul deserveşte o diversitate de utilizatori. Comunitatea utilizatorilor care folosesc internetul este într-o continuă creştere. Utilizatorii pot avea cunoştinţe diferite, interese şi scopuri de utilizare diferite. Majoritatea utilizatorilor nu au cunoştinţe legate de structura reţelei Internet, se poate să nu fie conştienţi de costul ridicat a unei căutări, se pot uşor pierde în „întunecimea" reţelei sau se pot uşor plictisi în aşteptarea unui rezultat.
- Doar o mică parte din informaţiile de pe Web sunt într-adevăr relevante sau valoroase pentru un utilizator la un moment dat. Se pare că aproape 99% din informaţiile obţinute de un utilizator sunt neinteresante sau neimportante pentru el. Cum pot fi determinate porţiunile din web care sunt într-adevăr relevante? Cum putem găsi o pagină web de înaltă calitate cu un topic specific? Iată doar câteva întrebări care mai aşteaptă un răspuns.

Există multe **motoare de căutare** bazate pe indecşi care caută în web, indexează pagini şi construiesc şi memorează indecşi imenşi bazaţi pe cuvinte cheie, care ajută la localizarea seturilor de pagini care conţin acele cuvintele cheie. Un utilizator experimentat poate să localizeze rapid un document prin furnizarea unui set de constrângeri de cuvinte cheie şi fraze. Oricum, căutarea bazată pe cuvintele cheie are două probleme majore: (1) fiecare topic conţine sute sau mii de documente. Aceasta face ca motorul de căutare să returneze numeroase pagini, multe dintre ele fiind doar parţial relevante pentru interogarea formulată sau conţin informaţii de o calitate slabă. (2) multe documente care ar fi relevante pentru o interogare pot să nu conţină cuvinte cheie definite în interogare. Aceasta este problema *polisemiei*. De exemplu cuvântul „leu" poate însemna un animal sălbatic care trăieşte în Africa sau o unitate monetară.

Mineritul pe web reprezintă o sarcină mult mai provocatoare, înseamnă ordonarea conţinutului, identificarea de structuri în Web, identificarea de regularităţi şi conţinuturi dinamice, mineritul pattern-urilor de acces la Web. În general, sarcina de minerit pe web poate fi clasificată în 3 categorii: *mineritul conţinutului web, mineritul structurii web* şi *mineritul utilizării web*.

2.3.1 **Mineritul structurii paginilor web**

Comparativ cu textul simplu dintr-un document, o pagină web conţine pe lângă text şi imagini şi o structură. Din acest motiv paginile web sunt considerate a fi date semistructurate. Structura de bază a unei pagini web este data de DOM (Document Object Model). DOM este o interfaţă independentă de platformă şi limbaj, care permite programelor şi script-urilor să acceseze dinamic şi să actualizeze conţinutul, structura şi stilul documentului web [W3C].

Structura DOM a unei pagini web este o structură arborescentă, în care fiecare tag (etichetă) HTML în pagina corespunde unui nod în arborele DOM. Pagina web poate fi delimitată (marcată) de taguri predefinite structural. Tag-uri utile includ <p> (paragraf), <table> (tabel), <li> (element de lista), <h1>... <h6> (titlu-heading) etc.

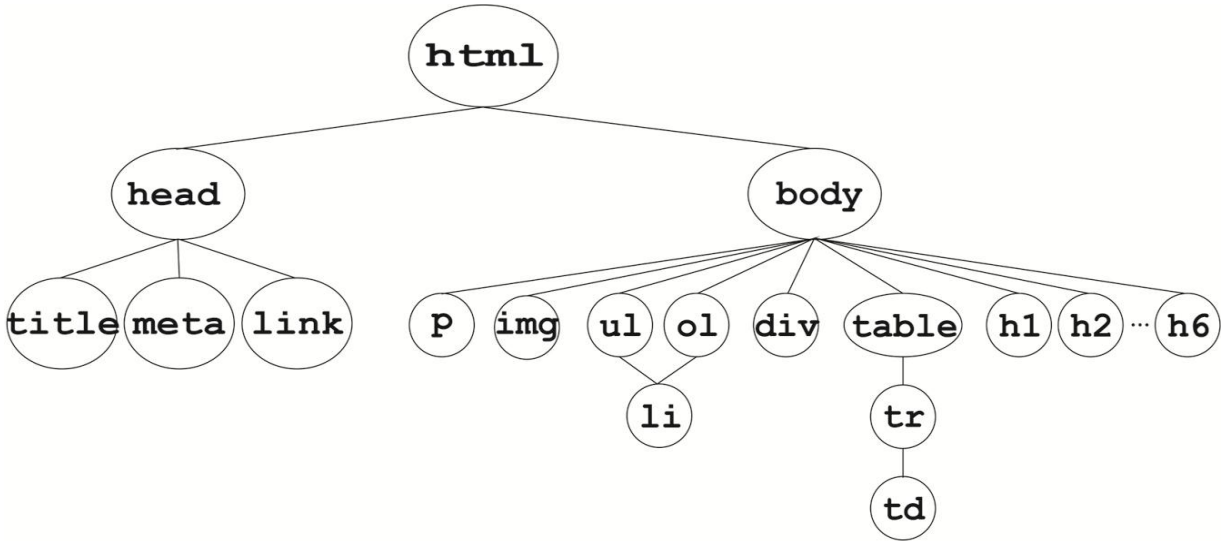


Fig. 2.1 Structura şi taguri conţinute într-o pagină web

Astfel, structura DOM poate fi folosită pentru a facilita extragerea de informaţii. Din păcate, datorită flexibilităţii de sintaxă HTML, multe pagini web nu respectă specificaţiile W3C privind structura HTML. Cel mai „nociv" standard utilizat în web a fost standardul HTML 3.2 care a permis amestecarea „formei" cu cea a „conţinutului" fapt ce a dus la îngreunarea indexării paginilor web. În prezent, standardul XHTML1 şi mai noul HTML5 încearcă o separare clară a conţinutului de forma de afişare prin utilizarea unor stiluri declarate în antetele paginilor sau în fişiere separate cu ajutorul sintaxei CSS **(**Cascade Style Sheets**).** Mai mult decât atât, arborele DOM a fost iniţial introdus pentru afişarea documentului în browser şi nu ca şi o descriere a structurii semantice a paginii Web. De exemplu, chiar dacă două noduri în arbore DOM au acelaşi părinte, cele două noduri ar putea fi semantic mai legate de alte noduri.

2.3.2 **Mineritul link-urilor pentru identificarea paginilor web autoritare**

Presupunem că un utilizator execută o interogare pentru a obţine pagini relative la un subiect dat, cum ar fi „curs valutar". Paginile relevante şi de înaltă calitate din punct de vedere al subiectului sunt considerate a fi pagini autoritare. Problema pentru motoarele de căutare este să găsească o modalitate pentru a identifica automat paginile web autoritare pentru un subiect dat. Interesant este faptul că „secretul" autorităţii este ascuns în legătura (link-ul) paginii. Web-ul conţine nu numai pagini dar şi hiperlink-uri care pointează de la o pagină la alta. Aceste hiperlink-uri conţin o cantitate enormă de notaţii umane latente care pot ajuta la regăsirea automată a noţiunii de autoritate. Când un autor al unei pagini creează un hiperlink către o altă pagină, aceasta înseamnă că autorul acordă încredere paginii spre care a făcut linkul. Colecţia de linkuri venite pentru o pagină dată de la autori diferiţi de pe web poate indica importanţa paginii şi se poate să conducă natural la descoperirea paginilor web autoritare. Aşadar, (meta)informaţiile web de legătură furnizează informaţii bogate despre relevanţă, calitate şi structura conţinutului web-ului şi astfel este o sursă bogată pentru mineritul web-ului.

Structura legăturilor web are nişte caracteristici unice. În primul rând, nu fiecare hiperlink reprezintă o „avizare" pentru o pagină, unele linkuri sunt create pentru alte scopuri cum ar fi navigarea sau reclame. Totuşi dacă majoritatea hiperlink-urilor către o pagină sunt linkuri de „avizare" atunci opinia colectivă va domina. În al doilea rând, din interese comerciale evidente o pagină autoritară nu va avea pe pagina proprie un link către rivali în acelaşi domeniu. În al treilea rând, paginile autoritare sunt rareori foarte descriptive.

Aceste proprietăţi ale structurii linkurilor conduce cercetarea la o altă categorie importantă a paginilor web numită *hub.* Un hub este o pagină sau un set de pagini care

furnizează colecţii de link-uri autoritare. Paginile hub pot fi chiar ele „neimportante" existând poate chiar puţine link-uri care pointează către ele. Oricum, ele furnizează link-uri către site-uri autoritare, având un subiect comun. Astfel de pagini pot fi liste de link-uri recomandate pe pagini personale, liste de referinţe ale unui curs sau site-uri comerciale. În general, un hub bun este o pagină cu link-uri către pagini autoritare de calitate.

2.3.3 **Mineritul utilizării web**

Pe lângă mineritul conţinutului şi structurii de legături, un alt task important este mineritul utilizări web-ului, care prelucrează fişierele log pentru descoperirea pattern-urilor de acces ale utilizatorilor la paginile web. Analizarea şi exploatarea regularităţilor în înregistrările de web log pot identifica potenţiali clienţi pentru comerţul electronic, mărirea calităţii şi livrarea serviciilor de informaţii pe internet la utilizatorul final şi îmbunătăţirea performanţei sistemelor de servere de pe web.

Serverele web de obicei înregistrează (în weblog) fiecare logare pentru fiecare acces la o pagină web. Aceasta include URL-ul, adresa IP de la care a fost făcută cererea iniţială şi „ştampila de timp" a cererii. Bazele de date weblog furnizează informaţii amănunţite despre dinamica web-ului. Pentru dezvoltarea tehnicilor de minerit pe weblog-uri acest aspect devine foarte important.

În dezvoltarea tehnicilor pentru utilizarea mineritului pe weblog-uri, putem considera următoarele: în primul rând, este important ca datele din aceste fişiere de weblog să fie valide şi de încredere. Adesea, datele din fişiere weblog trebuie să fie curăţate, condensate şi transformate într-o ordine pentru regăsirea şi analiza informaţiilor semnificative şi valoroase.

În al doilea rând, având URL-urile disponibile, timpul, adresa IP şi informaţii despre conţinutul pagini web se poate realiza o selecţie multidimensională iar cu ajutorul unei analize de tip OLAP (OnLine Analytical Processing – specifice masivelor de date) multidimensională se pot determina *N* useri de top, numărul celor mai accesate pagini web, perioade de timp cu cele mai multe accese şi altele care pot ajuta de exemplu în descoperirea clienţilor potenţiali, utilizatorilor, pieţelor comerciale.

În al treilea rând, data mining poate fi efectuat pe înregistrările weblog pentru găsirea pattern-urilor de asociere sau secvenţiale şi tendinţe pentru accesul la web.

Datele weblog furnizează informaţii despre acele grupuri de utilizatori accesează anumite grupuri de pagini, informaţiile weblog pot fi integrate cu conţinutul web şi structura de legături web pentru ajutorul mineritului paginilor, clasificarea documentelor web şi construcţia pe mai multe niveluri a informaţiilor de bază.

2.3.4 **Construirea informaţiilor de bază pe mai multe niveluri web**

Deoarece o pagină de web conţine totuşi, în afară de textul propriu-zis şi metainformaţii despre pagina respectivă, ar fi interesant să se structureze informaţiile de bază pe mai multe niveluri pentru a vedea dacă sunt realiste sau folositoare. Cel mai de jos nivel (cu cele mai multe detalii) poate fi web-ul însuşi. El nu poate fi separat în depozite. Mă voi referi la acest nivel ca *layer-0*. Totuşi este nerealist să creezi depozite web conţinând copii a fiecărei pagini de pe web, deoarece ar fi un duplicat al WWW.

Al doilea nivel pe care îl voi numi în continuare *layer-1* este nivelul de descriere al pagini web, conţinând informaţiile care descriu pagina web. Acesta ar trebui să fie mult mai mic decât *layer-0* dar suficient de bogat în prezentarea celor mai interesante informaţii generale pentru furnizarea cuvintelor cheie de bază, căutarea multidimensională sau minerit. Bazându-ne pe

varietatea de pagini web, *layer-1* poate fi organizat în diferite clase semistructurate cum ar fi: documente, persoane, organizaţii, vânzări etc.

Al treilea nivel, *layer-2*, reprezintă nivelul de servicii WEB. Acestea pot fi cataloage web construite deasupra nivelului, servicii specifice de aplicaţie care pot utiliza limbajul WSDL (Web Service Discovery Language) sau care pot utiliza protocoale mai simple cum ar fi SOAP (Simple Object Access Protocol - este un protocol pentru schimbul de informaţie structurat între servicii WEB).

Utilizând rangurile paginilor web date de motoare de căutare şi algoritmi de clasificare a paginii sau documentului putem obţine informaţii relevante de înaltă calitate şi pagini relevante de pe web din construcţia nivelului 1.

Standardizarea cu ajutorul XML-ului (eXtended Markup Language) va facilita schimbul de informaţii şi extragerea informaţiilor pentru construirea informaţiilor de bază pe mai multe nivele. În afară de asta, căutarea informaţiilor de bază pe web şi limbajele de descoperire a cunoştinţelor pot fi create şi implementate pentru astfel de scopuri. Web-ul semantic va avea ca bază această standardizare la care se vor adăuga diferite modele şi limbaje de descriere a datelor cum ar fi RDF (Resource Description Framework). RDF este un model standard pentru schimbul de date pe Web. RDF are caracteristici care să faciliteze fuzionarea informaţiilor, chiar dacă schemele care stau la baza acestor informaţii sunt diferite şi sprijină în mod specific evoluţia schemelor de-a lungul timpului fără a necesita la un moment data ca toate datele consumatorilor să fie schimbate.

2.3.5 Clasificarea automată a documentelor web

Clasificarea automată a unui document web constă în alocarea unei etichete de clasă care există într-un set pre-clasificat De exemplu colecţia DMOZ şi documentele web asociate anumitor categorii poate fi utilizată ca seturi de antrenament pentru obţinerea schemei de clasificare. Apoi după ce s-a obţinut schema de clasificare ea poate fi aplicată unor documente web noi pentru a le clasifica corespunzător.

Metodele de clasificare a documentelor pe baza cuvintelor cheie au fost prezentate în secţiunea 2.2, aceste metode pot fi utilizate pentru clasificarea documentelor web. Astfel, scheme de clasificare bazate pe termeni au dat rezultate bune pentru clasificarea documentelor web. Cu toate acestea, deoarece o pagină web poate conţine mai multe teme, publicitate şi informaţii de navigare, analiza de conţinut bazată pe blocuri de pagină poate juca un rol important în construcţia de modele de clasificare de înaltă calitate. De vreme ce hiperlink-urile conţin informaţii de înaltă calitate din punct de vedere semantic în comparaţie cu topicul paginii, este indicată utilizarea şi a unor astfel de informaţii semantice pentru a obţine o acurateţe şi mai bună decât simplele clasificări bazate pe cuvinte cheie. Totuşi, link-urile care „înconjoară” un document pot provoca zgomot iar folosirea simplă a termenilor din „vecinătatea de link-uri” a unui document poate chiar degrada acurateţea.

În ultimii ani s-a intensificat activitatea de cercetare privind construcţia şi utilizarea web-ului semantic; astfel informaţia despre infrastructura web va structura la un nivel superior web-ul care se va baza şi pe sensul conţinutului paginilor web. Clasificarea documentelor web prin web mining va ajuta la extragerea automata a sensului semantic a paginilor web şi la construirea ontologiilor pentru web-ul semantic. Iar dacă web-ul semantic se va construi cu succes, acesta va ajuta foarte mult automatizarea clasificării documentelor web.

2.4 *Clustering vs. clasificare*

În ultimii ani creşterea semnificativă a utilizării web-ului şi îmbunătăţirea calităţii şi vitezei internetului au transformat societatea noastră într-una care depinde puternic de informaţii. Cantitatea uriaşă de date care este generată de acest proces de comunicare conţine informaţii importante care se acumulează zilnic în bazele de date şi nu sunt uşor de regăsit. Domeniul mineritului datelor (data mining) s-a dezvoltat ca un mijloc de extragere de informaţii şi cunoştinţe din baze de date pentru a descoperi modele sau concepte care nu sunt evidente.

După cum se menţionează în [Berk06, Han01], învăţarea automată oferă tehnici de bază pentru data mining prin extragerea informaţiilor din datele brute conţinute în bazele de date. Procesul de obicei este format din următoarele etape:

- transformarea datelor într-un format adecvat;
- curăţarea datelor;
- deducerea sau extragerea concluziilor cu privire la date.

Învăţarea automată este împărţit în două subdomenii principale: învăţarea supervizată şi învăţarea nesupervizată. În cadrul categoriei de învăţare nesupervizată unul dintre instrumentele principale este clusteringul datelor. Clasificarea datelor face parte din cel de al doilea subdomeniu şi anume învăţarea supervizată.

2.4.1 **Învăţare nesupervizată şi supervizată**

În scopul de a clarifica diferenţele dintre clasificare şi clustering voi face o scurtă prezentare a învăţării supervizate şi nesupervizate. În învăţarea supervizată algoritmul primeşte atât datele (vectorizate) cât şi etichetele care reprezintă conceptul de învăţat, pentru fiecare caz. Scopul învăţării supervizate este ca algoritmul să înveţe conceptul, în sensul că, atunci când se prezintă un exemplu nou, pentru a fi clasificat, algoritmul ar trebui să prezică o etichetă pentru acest caz. Măsura calităţii clasificării este dată de acurateţea clasificării.

În conformitate cu această paradigmă, pe seturi relativ reduse de date există posibilitatea de „overfitting" sau „păcălire" prin memorarea tuturor etichetelor pentru fiecare caz de către algoritm, în detrimentul învăţării de către acesta a relaţiilor generale predictive dintre valorile atributelor şi etichete.

Rezultatele învăţării supervizate sunt de obicei evaluate pe un set de exemple de test, disjunct faţă de setul de exemple de antrenare. Metodele de clasificare utilizate sunt foarte variate, aria lor cuprinzând de la abordările tradiţionale statistice, reţelele neuronale şi până la algoritmi bazaţi pe nuclee precum SVM [Burg98].

În învăţarea nesupervizată algoritmul primeşte doar date neetichetate iar sarcina algoritmului este de a găsi o reprezentare adecvată a distribuţiei datelor (gruparea în clusteri relevanţi).

Unii cercetători au combinat învăţarea supervizată cu cea nesupervizată şi a apărut conceptul de învăţare semi-supervizată [Benn89]. De obicei, în acest tip de abordare învăţarea nesupervizată este aplicată iniţial setului de date necunoscut în scopul de a face unele presupuneri cu privire la distribuţia datelor şi apoi această ipoteză este confirmată sau infirmată printr-o abordare supervizată.

2.4.2 **Clasificare şi analiza clasificării**

Clasificarea automată de documente text este un aspect important în mineritul textului deoarece, datorită existenţei unui număr imens de documente on-line, este esenţială gruparea în mod automat a acestor documente în clase pentru a facilita regăsirea de documente şi analize ulterioare.

Pentru a efectua clasificarea automată de documente text se parcurg câteva etape. În primul rând, un set de documente preclasificate este considerat ca fiind setul de antrenare. Setul de antrenament este analizat pentru a obţine schema de clasificare. Schema de clasificare apoi trebuie să fie rafinată printr-un proces de testare. Schema de clasificare obţinută astfel poate fi aplicată la clasificarea celorlalte documente.

Acest proces pare oarecum similar cu clasificarea datelor din bazele de date relaţionale. Oricum există o diferenţă fundamentală. Datele relaţionale sunt structurate: fiecare tuplu este definit de un set de perechi *atribut-valoare*. Analiza clasificării decide care seturi de perechi de astfel de atribute are puterea de a determina dacă sau nu „se întâmplă ceva”. Pe de altă parte, seturile de documente nu sunt structurate pe perechi de tipul atribut-valoare. Un set de cuvinte-cheie asociate cu un set de documente nu este organizat într-un set fix de atribute sau dimensiuni. Dacă considerăm fiecare cuvânt-cheie distinct sau un atribut în document ca reprezentând o dimensiune, pot exista mii de dimensiuni într-un set de documente. Prin urmare, metodele de clasificare frecvent utilizate în bazele de date de date relaţionale, cum ar fi arbori de decizie, nu pot fi eficace pentru clasificarea seturilor de documente text.

Voi aminti câteva metode tipice de clasificare care au fost folosite cu succes în clasificarea documentelor text. Acestea includ clasificatorul k-NN (k-Nearest-Neighbor), metode de selecţie a trăsăturilor, clasificarea bayesiană, clasificatori SVM (Support Vector Machine) şi clasificatori care utilizează clasificarea bazată pe asociere.

Conform modelului vectorial de reprezentare a documentelor VSM, două documente sunt similare în cazul în care vectorii care reprezintă documentele sunt similari sau conţin părţi similare. Clasificatorul k-Nearest-Neighbor face parte din categoria algoritmilor de clasificare prin analogie şi pleacă de la premisa că documente similare vor primi aceeaşi etichetă de clasă. Atunci când un document de testare este prezentat algoritmul îl va trata ca pe o interogare la sistemul recunoaştere a informaţiilor şi va prelua din setul de antrenare cele *k* documente reprezentative pentru cele *k* categorii existente, *k* fiind o constantă dată. Eticheta de clasă a documentului de test poate fi determinată pe baza similarităţii acestuia fată de unul din cele *k* documente reprezentative şi va fi etichetat cu eticheta documentului reprezentativ cel mai similar. De aici provine şi numele algoritmului **k**-**N**earest-**N**eighbor. Această metodă are nevoie de resurse de memorie importante pentru a salva informaţiile legate de antrenare în comparaţie cu alte tipuri de clasificatori.

Clasificarea bayesiană este una din cele mai frecvent utilizate tehnici care pot fi utilizate eficient pentru clasificare documentelor text. Deoarece clasificarea documentelor poate fi privită ca o calculare a distribuţiei statistice a documentelor în anumite clase, un clasificator Bayesian calculează pentru setul de antrenament distribuţia $P(d|C)$ a documentului *d* pentru fiecare clasă *C* şi apoi în etapa de testare va genera pentru un document de test clasa cea mai probabilă pe baza distribuţiei stabilite în etapa de antrenare. Deoarece această metodă se poate aplica la seturi de date n-dimensionale, ea poate fi folosită pentru clasificarea documentelor text.

Support Vector Machine (SVM) este un alt exemplu de clasificator care lucrează eficient cu seturi de date mari, neseparabile liniar, deci şi cu documente text. Astfel, algoritmul construieşte o funcţie de mapare directă între mulţimea termenilor şi variabilele de clasă în etapa de antrenare. Prin determinarea unui hiperplan de separaţie a acestor puncte utilizate se realizează găsirea claselor pentru documentele din setul de testare. Când nu e posibila găsirea

unui hiperplan de separare se trece la o dimensiune superioara de reprezentare etc. Ideea esenţială, care diferenţiază SVM-ul de celelalte metode bazate pe nuclee, este aceea de a mapa, atunci când este necesar, datele de antrenament, reprezentate într-un anumit spaţiu, într-un alt spaţiu, de ordin superior prin intermediul unei funcţii Φ şi de a construi un hiperplan optimal de separare în acest nou spaţiu. Această operaţie conduce evident la o funcţie de separare neliniară în spaţiul iniţial de date. Deoarece toţi vectorii apar în produse scalare utilizând funcţia nucleu $\langle \phi(\mathbf{w}), \phi(x) \rangle$, hiperplanul de separare poate fi determinat fără a proiecta explicit datele în noul spaţiu de trăsături. Am notat cu **w** vectorul pondere iar cu $\langle\ ,\ \rangle$ produsul scalar a doi vectori.

Calculele sunt reduse semnificativ prin introducerea unui nucleu pozitiv definit *k*, astfel ca $k(x,x') := \langle x, x' \rangle$. Această substituire, care este referită uneori ca trucul nucleu. Deoarece toţi vectorii de trăsături apar doar în produse scalare poate fi aplicat trucul nucleu. Utilizând funcţia Φ reprezentăm vectorii de intrare în noul spaţiu. Pentru nuclee sunt folosiţi în principal nucleul polinomial şi nucleul Gaussian. Conform [Mora08] pentru nucleul polinomial singurul parametru care trebuie modificat este gradul iar pentru nucleul Gaussian rămâne de modificat parametru *C* după următoarele formule:

- Polinomial:

$$k(x,x') = \left(2 \cdot d + \langle x \cdot x' \rangle\right)^d \tag{2.9}$$

- *d* fiind sigurul parametrul care trebuie modificat.

- Gaussian (radial basis function RBF):

$$k(x,x') = \exp\left(-\frac{\|x - x'\|^2}{n \cdot C}\right) \tag{2.10}$$

- *n* este numărul de elemente distincte care au ponderea mai mare ca 0 în vectorii de intrare.

În formulele 2.9 şi 2.10 *x* şi *x'* reprezintă vectorii de intrare. Clasificatorul de tip SVM este mai amănunţit prezentat în secţiunea 5.5.

Metode efective pentru clasificarea documentelor sunt şi cele care utilizează clasificarea bazată pe asociere, care clasifică documente bazându-se pe un set de asocieri pe bază de pattern-uri de text apărute frecvent. Deoarece termenii foarte frecvenţi nu diferenţiază semnificativ documentele în acest tip de clasificare se vor folosi numai acei termeni care nu sunt foarte frecvenţi şi care diferenţiază puternic documentele.

Metodele de clasificare pe bază de asocieri parcurg următorii paşi: (1) se extrag cuvintele cheie şi termeni prin sisteme de regăsire a informaţiei şi tehnici de analiză a asocierilor; (2) se obţine o ierarhie de cuvinte-cheie şi termeni utilizând clasele de termeni disponibile (a se vedea WordNet sau o ontologie de domeniu) sau pe baza unor sisteme de clasificare bazate pe cuvinte cheie. Documentele din setul de antrenament pot fi astfel clasificate într-o ierarhie de clase. Prin metodele de minerit al asocierii termenilor se descoperiră seturile de termeni asociaţi care pot fi utilizaţi pentru diferenţierea maximă a unei clase de documente faţă de alta. Se obţine astfel un set de reguli de asociere asociate cu fiecare clasă de documente. Astfel de reguli de clasificare pot fi ordonate pe baza puterii de diferenţiere a frecvenţei de apariţie şi folosite pentru a clasifica noi documente.

2.4.3 Clustering şi analiza clusterilor

Analiza clusterilor include două aspecte majore: clustering şi validarea clusterilor găsiţi [Cret07].

Clusteringul se referă la gruparea unor obiecte în funcţie de anumite criterii. Pentru a atinge acest scop au fost dezvoltaţi un număr mare de algoritmi [Jain88, Kauf90, Jain90, Berk06]. Deoarece nu există algoritmi generali care se pot aplica la toate tipurile de aplicaţii, devine necesară aplicarea unui mecanism de evaluare astfel încât utilizatorul să poată găsi un algoritm potrivit pentru aplicaţia sa particulară.

Analiza clusterilor este un proces iterativ de clustering şi validare a clusterilor de către utilizator, facilitat prin algoritmi de clustering şi metode de validare a clusterilor. Analiza clusterilor este un proces de descoperire prin explorare. Poate fi utilizat pentru a descoperi structuri fără a fi nevoie de interpretări [Jain88].

Validarea clusterilor devine astfel procesul de evaluare a calităţii unui cluster.

2.4.3.1 Definiţii

Definiţia 1. Clusteringul este procesul de grupare a unor obiecte fizice sau abstracte în clase de obiecte similare [Jain90].

Un cluster este o colecţie de obiecte care sunt similare unul faţă de celălalt şi sunt nesimilare (diferite) faţă de obiectele care nu fac parte din acel cluster.

Clusteringul şi analiza clusterilor este o activitate deosebit de importantă în aplicaţiile pentru recunoaşterea de patern-uri, procesarea de imagini, cercetări de piaţă etc. Clusteringul este important pentru clasificarea documentelor WEB în vederea extragerii de cunoştinţe. În mineritul datelor, clusteringul şi analiza clusterilor se poate utiliza ca aplicaţie de sine stătătoare pentru analizarea distribuţiei datelor, pentru a observa anumite caracteristici ale fiecărui cluster. De asemenea, analiza clusterilor se poate utiliza ca etapă de preprocesare în cadrul unor algoritmi de clasificare.

Din cauza cantităţilor impresionante de date acumulate în diferite baze de date, clusteringul a devenit o ramură de cercetare importantă în cadrul mineritului datelor. În principal, direcţiile de cercetare s-au axat mai mult pe clustering bazat pe distanţă. Astfel au fost dezvoltaţi algoritmii *k-Means* şi *k-Medoids*.

Cercetări ulterioare au demonstrat utilitatea şi altor abordări cum ar fi: utilizarea unor algoritmi bazaţi pe arbori de sufixe [Meye05], [Zami98], pe ontologii [Hoth03], [Bate08] şi pe algoritmi bio-inspirati, exploatând o inteligenţă colectivă, precum cei bazaţi pe comportamentul furnicilor în căutarea hranei [Abra03], [Labr03], [Dori00] sau pe roiuri de particule (particle swarm optimization) având un comportament similar cu al unui stol de pasări în căutarea hranei [Merw03].

Definiţia 2. Clusteringul conceptual - obiectele dintr-un grup vor forma o clasă doar dacă aceasta poate fi descrisă printr-un concept. Această abordare diferă faţă de clusteringul convenţional unde măsura disimilarităţii se bazează pe distanţa geometrică. Clusteringul conceptual are două componente:

1. Descoperă clasele apropiate;
2. Creează descrieri pentru fiecare clasă ca şi la clasificare.

Ideea de bază în clustering, care este punctul de pornire pentru ambele abordări, este găsirea acelor clusteri cu similaritate intraclasă foarte mare şi similaritate interclasă foarte mică.

2.4.4 Cerinţe cheie pentru algoritmii de clustering

Cerinţe tipice pentru algoritmii de clustering în mineritul datelor şi a documentelor WEB după [Han01] sunt:

1. **Scalabilitatea algoritmilor**: Mulţi algoritmi de clustering lucrează foarte bine cu seturi relativ mici de date. Totuşi, bazele de date mari conţin milioane de obiecte iar utilizarea unui eşantion din aceste seturi mari ar duce la rezultate neconcludente.
2. **Abilitatea de a utiliza tipuri diferite de atribute**: Mulţi algoritmi de clustering utilizează date numerice ca date de intrare. În unele cazuri este necesar să se aplice algoritmi de clustering pe date binare, pe date ordinale sau pe o combinaţie dintre acestea.
3. **Recunoaşterea clusterilor de formă arbitrară**: Mulţi algoritmi determină clusterii utilizând distanţe geometrice cum ar fi distanţa euclidiană sau distanţa Manhattan. Aceşti algoritmi tind însă să găsească clusteri sferici cu dimensiune şi densitate asemănătoare. În realitate, clusterii pot avea orice formă.
4. **Stabilirea parametrilor de intrare bazată pe cunoaşterea minimă a domeniului**: Mulţi algoritmi de clustering în analiza clusterilor cer anumiţi parametri cum ar fi numărul clusterilor de determinat (ex. k-Means). Rezultatul clusteringului poate fi sensibil diferit în funcţie de parametrii de intrare stabiliţi. Parametrii de intrare, în special pentru seturi de date care conţin obiecte cu dimensionalitate mare, sunt greu de determinat.
5. **Abilitatea de a utiliza date care conţin zgomot**: Multe seturi de date reale conţin date incomplete, necunoscute sau outlineri (date care sunt complet diferite faţă de marea majoritate a datelor) Unii algoritmi sunt sensibili la astfel de date şi calitatea clusteringului devine slabă.
6. **Insensibilitate la ordinea de prelucrare a datelor**: Unii algoritmi de clustering pot produce rezultate diferite la schimbarea ordinii prelucrării datelor de intrare (ex. Single Pass, BIRCH). Este importantă dezvoltarea unor algoritmi care nu sunt sensibili la ordinea datelor de intrare.
7. **Dimensionalitate mare**: Datele pot conţine o mulţime de dimensiuni şi de atribute. Mulţi algoritmi de clustering reuşesc să producă rezultate bune în cazul datelor cu două sau trei dimensiuni. Ochiul uman reuşeşte să evalueze calitatea unui cluster cu până la trei dimensiuni.
8. **Clustering care ţine cont de anumite constrângeri:** Dacă ar trebui identificate zone urbane, atunci anumite graniţe naturale cum sunt râuri, şosele importante sunt constrângeri semnificative pentru unele aplicaţii şi de care algoritmii de clustering trebuie să ţină cont.
9. **Interpretabilitate şi utilitate**: Utilizatorii doresc ca rezultatele clusteringului să fie utile, interpretabile şi inteligibile.

2.5 *Metrici de similaritate a documentelor text*

2.5.1 Structurarea datelor utilizate în clustering

În această secţiune voi prezenta principalele metrici şi tipuri de date utilizate în clustering şi clasificare şi cum se preprocesează pentru algoritmii care le vor utiliza.

2.5.1.1 Matricea de date

Această matrice va reprezenta, de exemplu în clustering, *n* obiecte care fiecare are *m* atribute. Astfel se va crea o matrice de *n* x *m* atribute.

$$\begin{pmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1m} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{ij} & \dots & x_{im} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & \dots & x_{nj} & \dots & x_{nm} \end{pmatrix} \tag{2.11}$$

2.5.1.2 Matricea de disimilaritate

Această matrice va fi o matrice pătratică *n* x *n* şi va conţine măsurile disimilarităţilor dintre toate perechile de obiecte supuse clusteringului. Deoarece *d(i,j)=d(j,i)* şi *d(i,i)*=0 obţinem matricea de disimilaritate de forma:

$$\begin{pmatrix} 0 & . & . & . & . \\ d(2,1) & 0 & . & . & . \\ d(3,1) & d(3,2) & 0 & . & . \\ \dots & \dots & \dots & 0 & . \\ d(n,1) & d(n,2) & \dots & d(n,n-1) & 0 \end{pmatrix} \tag{2.12}$$

unde *d(i,j)* reprezintă disimilaritatea măsurată dintre două obiecte.

Obiectele sunt grupate în funcţie de similaritatea dintre ele sau în funcţie de disimilaritatea dintre ele.

În continuare, voi prezenta unele aspecte legate de modalităţi de evaluare a calităţii clusteringului bazate pe coeficienţi de corelaţie care pot fi convertiţi la coeficienţi de disimilaritate sau de similaritate. Voi prezenta cum poate fi exprimată disimilaritatea dintre obiecte care sunt fie descrise cu ajutorul variabilelor scalate într-un anumit interval, fie descrise de variabile binare, fie de variabile nominale sau combinaţii ale acestora.

2.5.2 **Disimilaritate şi similaritate: măsuri ale calităţii clusteringului**

În general, disimilaritatea *d(i,j)* este un număr pozitiv apropiat de 0 când *i* şi *j* se află aproape unul de celălalt şi creşte proporţional cu distanţa între *i* şi *j*. Disimilaritatea dintre două obiecte se poate obţine prin evaluări subiective efectuate de către experţi pe baza observaţiei directe sau poate fi calculată pe baza unor coeficienţi de corelare.

2.5.3 **Distanţe uzuale**

Pentru a putea calcula disimilaritatea dintre obiecte vom calcula distanţa între fiecare două obiecte. Distanţa se defineşte în spaţiul R^n astfel [Fre1906]:

Fie mulţimea $X \neq \emptyset$. Se numeşte distanţă pe X, o funcţie $d: X \times X \to R$, cu proprietăţile:

1. $\forall\ x,y \in X$ $\quad d(x,y) \geq 0$ şi $d(x,y)=0 \Leftrightarrow x=y$;
2. $\forall\ x,y \in X$ $\quad d(x,y) = d(y,x)$;

3. $\forall$ x,y,z$\in$X d(x,z)$\leq$ d(x,y) + d(y,z) (regula triunghiului).

Perechea (X,d), cu X $\neq\emptyset$ şi d metrică pe X se numeşte spaţiu metric.

Pe aceeaşi mulţime X se pot defini diverse metrici, deci mai multe structuri de spaţiu metric.

În R^n distanţa dintre două puncte $x=(x_1,x_2,\ldots,x_n)$ *şi* $y=(y_1,y_2,..,y_n)$ se poate defini ca fiind numărul real *d(x,y)* şi se poate calcula utilizând diferite formule prezentate mai jos.

Distanţa euclidiană: este distanţa dintre două puncte din spaţiul *n*-dimensional.

Distanţa euclidiană între două obiecte de *n* dimensiuni are valorile în intervalul [0, ∞) şi se calculează după formula:

$$d_E(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2} \tag{2.13}$$

Distanţa Manhattan: această distanţă diferă de distanţa euclidiană clasică prin faptul că ea se măsoară ca şi când drumul s-ar parcurge pe axe perpendiculare (analogie cu străzile din Manhattan), are valorile în intervalul [0,∞).

$$d_{Ma}(x_i, x_j) = \sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right| \tag{2.14}$$

Distanţa Minkowski: este o generalizarea a distanţei euclidiene şi a distanţei Manhattan. În cazul în care p→∞ se obţine distanţa Cebîşev. Are valorile în intervalul [0, ∞).

$$d_{Mi}(x_i, x_j) = \left(\sum_{k=1}^{n} (x_{ik} - x_{jk})^p \right)^{1/p}, \qquad p \geq 1 \tag{2.15}$$

Distanţa cosinus: este o măsură de similaritate şi calculează „unghiul" dintre doi vectori din spaţiul *n*-dimensional. Are valorile în intervalul [0, 1].

$$d_{\cos} = \frac{\sqrt{\sum_{k=1}^{n} x_{ik}^2} \cdot \sqrt{\sum_{k=1}^{n} x_{jk}^2}}{\sum_{k=1}^{n} x_{ik} \cdot x_{jk}} \tag{2.16}$$

Distanţa Canberra:

$$d_{CAN}(i, j) = \sum_{k=1}^{n} \frac{\left| x_{ik} - x_{jk} \right|}{\left| x_{ik} \right| + \left| x_{jk} \right|} \tag{2.17}$$

Distanţa Canberra se modifică semnificativ pentru vectori care conţin multe valori 0, cum este în cazul vectorilor de reprezentare a documentelor şi întoarce valori în intervalul [0, ∞).

Distanţa Bray-Curtis:

$$d_{BC}(i, j) = \frac{\sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right|}{\sum_{k=1}^{n} x_{ik} + \sum_{k=1}^{n} x_{jk}} \tag{2.18}$$

Această distanţă este asemănătoare distanţei Canberra iar când toate coordonatele sunt pozitive distanţa este în intervalul [0,1]

2.5.4 Tipuri de variabile utilizate în clustering/clasificare

2.5.4.1 Variabile scalate într-un anumit interval

Variabilele scalate într-un anumit interval sunt cele obţinute de exemplu în urma operaţiei de normalizare descrisă în par. (2.1.3.2). Aceste variabile vor fi folosite apoi la descrierea măsurilor distanţelor (de ex. distanţa euclidiană, distanţa Manhattan ş.a).

2.5.4.2 Variabile standardizate

Pentru standardizarea valorilor o variantă este de a converti valorile originale la valori fără unitate de măsură. Astfel, pentru o variabilă *x* care are *n* valori x_n se parcurg următorii paşi:

Pas 1: Calculăm abaterea medie absolută

Abaterea medie absolută $\overline{d}$ reprezintă media aritmetică simplă sau ponderată a abaterilor "absolute" ale valorilor variabilei de la tendinţa lor centrală, caracterizată cu ajutorul mediei sau al medianei.

În cazul în care abaterea valorilor individuale sunt calculate şi analizate faţă de medie, atunci abaterea medie absolută $\overline{d}_{\bar{x}}$ pentru variabila *x* se determină astfel:

$$\overline{d}_{\bar{x}} = \frac{1}{n}\sum_{i=1}^{n}|x_i - m| \tag{2.19}$$

unde x_i este valoarea variabilei, şi

$$m = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.20}$$

unde *n* este numărul de valori.

Exemplu:

Fie pentru o variabilă următoarele valori: {1, 2, 3, 5, 4, 10, 0}

Atunci $m = \frac{1+2+3+5+4+10+10}{7} = 3{,}57$

iar abaterile medii

x	**1**	**2**	**3**	**5**	**4**	**10**	**0**	$\overline{d}_{\bar{x}} = \frac{1}{n}\sum_{i=1}^{n}\lvert x_i - m\rvert$
$\lvert x - 3{,}57\rvert$	2,571429	1,571429	0,571429	1,428571	0,428571	6,428571	3,571429	2,367347

Tabel 2.1 Exemplu de calcul al abaterilor medii

Pas 2 Calcularea scorului standard (z-score) care reprezintă valoarea standardizată a variabilei.

Calculăm scorul după formula:

$$z_i = \frac{x_i - m}{\overline{d}_{\bar{x}}}, i = \overline{1,n} \tag{2.21}$$

Astfel valorile standardizate pentru variabila x dată devin:

x	**1**	**2**	**3**	**5**	**4**	**10**	**0**
z_i	-1,08621	-0,66379	-0,24138	0,603448	0,181034	2,715517	-1,50862

Tabel 2.2 Exemplu de calcul al z-score

Standardizarea datelor poate fi utilă în anumite aplicaţii de clustering sau clasificare, mai ales când se doreşte compararea rezultatelor unor algoritmi diferiţi şi care au ieşiri în domenii diferite.

2.5.4.3 Variabile binare (dihotomice)

O variabilă binară are doar două stări: 0 sau 1, zero însemnând că variabila nu exista şi 1 când ea exista. De exemplu, pentru variabila "bolnav" care descrie o persoană, valoarea 1 însemnă că persoana este bolnavă iar valoarea 0 indică faptul că persoana respectivă nu este bolnavă. Tratarea acestor variabile ca şi când ele ar fi variabile scalate într-un anumit interval ar duce la interpretări greşite.

2.5.4.3.1 Matricea de disimilaritate pentru variabile binare

Presupunând că toate variabilele au aceeaşi pondere putem construi un tabel de contingenţă de 2x2 (Tabel 2.3) unde *a* reprezintă numărul de variabile care au valoarea 1 atât în obiectul *i* cât şi în *j*, b este numărul de variabile care sunt 1 pentru *i* dar 0 pentru *j*, c este numărul de variabile care sunt 0 pentru *i* dar 1 pentru *j* iar d este numărul de variabile a căror valoare este 0 atât pentru *i* cât şi pentru *j*.

Numărul total de variabile este $p=a+b+c+d$

		obiectul *j*		
		1	0	
obiectul *i*	1	*a*	*b*	*a+b*
	0	*c*	*d*	*c+d*
		a+c	*b+d*	*p*

Tabel 2.3 Tabel de contingenţă

Calculul disimilarităţii invariante: o variabilă este simetrică dacă ambele stări au aceeaşi pondere şi nu contează care din ele este codată ca fiind 1 şi care va fi codată cu valoare 0. Calculul disimilarităţii bazat pe variabile simetrice se numeşte **disimilaritatea invariantă**.

Pentru calculul disimilarităţii se va utiliza coeficientul dat de formula:

$$d(i,j)=\frac{b+c}{a+b+c+d} \qquad (2.22)$$

Coeficientul de la formula (2.20) se află în intervalul [0,1].

Exemplu:

Fie următoarele 3 documente:

d_1: Peştele este în tigaia mea.

d_2: Tigaia mea este roşie.

d_3: Mark este student.

Vocabular={peştele, este, în, tigaia, mea, roşie, Mark, student}

	Vocabular							
Document	*peştele*	*este*	*în*	*tigaia*	*mea*	*roşie*	*Mark*	*student*
d_1	1	1	1	1	1	0	0	0
d_2	0	1	0	1	1	1	0	0
d_3	0	1	0	0	0	0	1	1

Calculăm disimilaritatea dintre d_1 şi d_2;

		d_2		
		1	0	
d_1	1	*3*	*2*	*a+b*
	0	*1*	*2*	*c+d*
		a+c	*b+d*	*p*

$$d(d_1, d_2) = \frac{3}{8} = 0,375 \tag{2.23}$$

		d_3		
		1	0	
d_1	1	*1*	*4*	*a+b*
	0	*2*	*1*	*c+d*
		a+c	*b+d*	*p*

$$d(d_1, d_3) = \frac{6}{8} = 0,75 \tag{2.24}$$

Din (2.23) şi (2.24) se poate observa faptul că documentele d_1 şi d_2 au o valoare a coeficientului de disimilaritate mai mică.

Observaţie: Când coeficientul de disimilaritate atinge valoarea 1 atunci obiectele nu sunt similare iar când atinge valoarea 0 atunci sunt computaţional identice.

O variabilă binară este **asimetrică** dacă codarea ieşirilor are importanţă diferită pentru ieşiri cum ar fi, de exemplu, codarea ieşirilor pentru un test de boală. Astfel, similaritatea a două ieşiri codate cu 1 devine mai importantă decât similaritatea a două ieşiri codate cu 0.

Acest tip de similaritate se numeşte similaritate noninvariantă. Cel mai cunoscut coeficient pentru acest tip de similaritatea este coeficientul lui **Jaccard** care se determină după formula:

$$d(i, j) = \frac{b + c}{a + b + c} \tag{2.25}$$

Exemplul

Utilizăm din nou datele din exemplul precedent şi presupunem că prezenţa comună a cuvintelor în propoziţii este mai importantă decât lipsa lor. Astfel codate variabilele noastre devin asimetrice.

Calculăm coeficientul de disimilaritate Jaccard pentru documentele noaste:

$$d(d_1, d_2) = \frac{b + c}{a + b + c} = \frac{2 + 1}{3 + 2 + 1} = \frac{3}{6} = 0,5 \tag{2.26}$$

$$d(d_1, d_3) = \frac{b + c}{a + b + c} = \frac{4 + 2}{1 + 4 + 2} = \frac{6}{7} = 0,85 \tag{2.27}$$

$$d(d_2, d_3) = \frac{b + c}{a + b + c} = \frac{3 + 2}{1 + 3 + 2} = \frac{5}{6} = 0,83 \tag{2.28}$$

Din (2.26), (2.27), (2.28) rezultă că documentele d_1 cu d_2 sunt mai similare decât d_1 cu d_3 sau d_2 cu d_3.

2.5.4.4 Variabile nominale

Variabilele nominale sunt o generalizare a variabilelor binare, doar că variabilele nominale pot să aibă mai mult de două stări.

Fie M numărul de stări pe care le poate lua o variabilă nominală. Stările variabilei pot fi notate prin numere întregi ca 1,2,3,..M sau litere sau alte simboluri care nu au o ordine specifică.

Pentru a calcula disimilaritatea dintre două obiecte i şi j se poate utiliza formula de potrivire simplă:

$$d(i, j) = \frac{p - m}{p} \tag{2.29}$$

m reprezintă numărul de potriviri (numărul variabilelor pentru care i şi j sunt în aceeaşi stare)

p este numărul total de variabile.

2.6 *Evaluarea algoritmilor de clasificare/ clustering*

Fiecare algoritm de clustering aplicat pe acelaşi set de date va grupa datele respective din diferite puncte de vedere, în funcţie de metrica de similaritate folosită. Aceasta face analiza eficienţei algoritmilor de clustering foarte dificilă.

Pentru a evalua performanţa sau calitatea algoritmilor de clustering trebuie să fie stabilite măsuri obiective. Există 3 tipuri de măsuri de calitate:

a. externe, atunci când există o cunoaştere a priori despre clusteri (avem date pre-etichetate);
b. interne, care nu au nici o informaţie despre clusteri;
c. relative, care evaluează diferenţele dintre soluţiile de grup diferite.

Măsurile externe sunt aplicate atât la algoritmii de clasificare cât şi la algoritmii de clustering, în timp ce măsurile interne şi relative sunt aplicate numai la clustering.

2.6.1 **Măsuri externe de validare a clusteringului şi a clasificării**

Măsurile externe de validare, care permit analiza rezultatelor algoritmilor, necesită existenţa unor seturi pre-etichetate de date în faza de testare. Deoarece gruparea datelor se poate face din multe puncte de vedere iar etichetele pot varia în funcţie de aceste puncte de vedere, la datele pre-etichetate se renunţă la categorie iar comparaţia se face faţă de grupul care conţine cele mai multe date dintr-o anumită categorie.

Voi prezenta în continuare câteva din măsurile de evaluare externă:

Fie C_i o categorie de documente şi S_j o clasă cunoscută:

Precizia: este probabilitatea ca un document clasificat corect să fie util. Precizia ia valori in domeniul [0,1], cu 1 cel mai bun rezultat.

$$precizia(C_i, S_j) = \frac{|C_i \cap S_j|}{|C_i|} \tag{2.30}$$

Recall: este probabilitatea ca un document util să fie şi clasificat corect. Recall ia valori în domeniu [0,1], cu 1 cel mai bun rezultat.

$$recall(C_i, S_j) = \frac{|C_i \cap S_j|}{|S_i|} \tag{2.31}$$

Acurateţea: reprezintă procentajul de documente care sunt grupate corect în categorii în funcţie de etichetele documentelor.

F-measure: este o măsură care combină precizia şi recall-ul prin media armonică şi se calculează conform formulei:

$$F-measure(C_i,S_j)=\frac{2\cdot precizia(C_i,S_j)\cdot recall(C_i,S_j)}{precizia(C_i,S_j)+recall(C_i,S_j)} \quad (2.32)$$

Pentru fiecare clasă se va selecta doar clusterul cu cea mai mare F-measure. În final, măsura generală F-measure pentru soluţia unui clustering este ponderată prin dimensiunea fiecărui cluster.

$$F(S)=\sum_i \frac{n_i}{n}\max(F(C_i,S_j)) \quad (2.33)$$

*Entropia***.** Entropia indică omogenitatea unui cluster: o entropie scăzută indică o omogenitate mare (caz dorit) şi invers. Un cluster cu un singur element va avea valoarea entropiei zero. Valoarea entropiei unui cluster devine 1 în cazul în care conţine un număr egal de elemente din fiecare clasă din soluţia iniţială. Formula pentru calculul entropiei pentru un cluster C_i este:

$$E(C_i)=-\sum_{i=0}^{k-1} pr_{ij}\cdot\log(pr_{ij}) \quad (2.34)$$

unde pr_{ij} reprezintă raportul dintre numărul de elemente din clasa *j* conţinute în clusterul *i*. Calculul entropiei pentru soluţia de clustering a unui set de date cu k clase este dată de formula:

$$E(S)=\sum_i \frac{n_i}{n}E(C_i) \quad (2.35)$$

2.6.2 **Măsuri de validare internă a clusterilor**

În [M01] sunt prezentate patru astfel de metrici care sunt specifice algoritmilor de clustering:

1.Compactness – măsoară cât de compacte sunt datele în cadrul unui cluster:

$$compactness(c)=\sum_{i=1}^{n_C}(\vec{c}-\vec{x}_i)^2 \quad (2.36)$$

unde *c* este clusterul curent, n_C numărul de elemente din cadrul clusterului, $\vec{c}$ este centroidul clusterului şi x_i un element din cluster**.** Cu alte cuvinte, această metrică măsoară cât de „apropiate" sunt documentele din cadrul unui cluster. Valoarea este în domeniul $[0,\infty)$ cu cât mai mică, cu atât mai bună.

2. Separability – măsoară disimilaritatea (separabilitatea, disjuncţia) între clusterii creaţi:

$$separability=\sum_{i,j=1}^{C} dist(\vec{c}_i,\vec{c}_j) \quad (2.37)$$

Astfel, pentru fiecare cluster se determină clusterul cel mai apropiat. Pe baza acestei metrici căutăm clusteri care maximizează această valoare, deci căutăm clusteri care sunt cât mai distanţaţi.

3. Balance – măsoară cât de echilibraţi sunt clusterii formaţi:

$$balance = \frac{n / k}{\max_{i=\overline{1,k}}(n_i)} \tag{2.38}$$

unde *n* este numărul total de documente supuse clusteringului, *k* este numărul de clusteri formaţi ($k \leq n$), n_i numărul de documente din clusterul *i* iar numitorul reprezintă numărul de documente din clusterul cel mai numeros.

Această măsură poate lua valori în domeniul [0,1], valoarea maximă 1 se obţine atunci când toţi clusterii generaţi au acelaşi număr de documente. O valoare apropiată de 0 se obţine atunci când numărul de documente conţinute de clusteri diferă foarte mult.

4. Davies-Bouldin-Index [Davi79] – combină primele două metrici:

Fie:

$$scatter(c) = \frac{compactness(c)}{|X_c|} \tag{2.39}$$

unde $|X_c|$ este numărul de elemente din clusterul c.

Indexul Davies-Bouldin pentru clusterul *i* devine:

$$r_i = \max_{j, j \neq i} \left(\frac{scatter(c_i) + scatter(c_j)}{dist(c_i, c_j)} \right) \tag{2.40}$$

Prin intermediul acestei măsuri căutăm clusteri care minimizează parametrul *r*, caz în care distanţele între clusteri sunt mari şi gradul de împrăştiere a clusterilor este mic. Astfel, s-a găsit un echilibru agregat între separability (văzută ca distanţă - să fie mare) şi compactness (văzută ca împrăştiere - să fie mică).

2.7 *Seturi de date utilizate*

2.7.1 Setul de date Reuters

Deoarece continui munca de cercetare a domnului dr. Daniel Morariu prezentată în [Mora07] pe partea de preprocesare a setului de date Reuters nu apare nici o modificare.

În experimente efectuate am folosit colecţia de date Reuters-2000 [Reut00], care conţine 984Mb de articole de tip ştiri. Colecţia Reuters 2000 este folosită în clasificare de mulţi cercetători, aceasta conţinând 806.791 ştiri publicate de agenţia Reuters între anii 1996-1997.. Documentele sunt preclasificate în trei categorii mari astfel: în funcţie de ramura industrială la care face referire articolul, în funcţie de zona geografică şi, după anumite categorii specifice propuse de Reuters, în funcţie de conţinutul ştirii astfel existând 126 de categorii diferite. În experimente efectuate am considerat clasele propuse de Reuters în funcţie de conţinut pentru documente ca fiind „perfecte”.

În partea de preprocesare am parcurs etapele de extragere a cuvintelor, de eliminare a cuvintelor de legătură şi de extragere a rădăcinii cuvintelor. Am folosit lista de cuvinte de legătură pentru documente în limba engleză oferită de Universitatea din Texas [IR]. Pentru extragerea rădăcinii cuvintelor am folosit algoritmul „Porter” [Rijs80]. Numărul de apariţii a rădăcinii cuvântului dintr-un document a fost apoi contorizată obţinând un vector de frecvenţe pentru fiecare document.

Aceste cuvinte le vom numi în continuare trăsături caracteristice - *features*. Acest vector îl vom considera ca fiind reprezentarea vectorială a documentului în spaţiul trăsăturilor caracteristice. Deoarece nu toate cuvintele apar în fiecare document, a mai fost creat un vector

care conţine toate cuvintele ce apar în toate documentele din setul de date. Acest vector caracterizează întreg setul de documente iar dimensiunea lui reprezintă dimensiunea spaţiului de reprezentare a tuturor documentelor din setul respectiv.

Există mai multe posibilităţi de reprezentare a trăsăturilor din vectori pe care le-am detaliat în secţiunea 4.1.1.2.

Fiecare vector iniţial creat pentru un document a fost modificat astfel încât să devină de lungimea vectorului care conţine toate cuvintele, specificându-se valoarea 0 pe poziţiile cuvintelor ce nu apar în documentul respectiv, pe celelalte poziţii specificându-se frecvenţa termenilor.

După acest pas toţi vectorii de reprezentare a documentelor din setul de date au devenit de aceeaşi dimensiune şi putem considera că fiecare vector reprezintă semnătura unui document în spaţiul de reprezentare a setului de date.

Deoarece memoria necesară pentru a stoca aceşti vectori este destul de mare, în [Mora07] s-a ales varianta de a memora doar valorile din vector pentru care numărul de apariţii al cuvintelor este diferit de zero. Pentru fiecare vector în parte, la sfârşit se păstrează categoriile (clasele) propuse de Reuters pentru documentul respectiv.

2.7.1.1 Alegerea documentelor pentru antrenare - testare

Pentru a putea compara rezultatele experimentale obţinute cu cele prezentate în [Mora07] – care sunt considerate rezultate de bază, în scopul îmbunătăţirii lor prin cercetările din această lucrare - am ales documentele în aceeaşi manieră. Din cele peste 800000 de documente, am ales acelea clasificate de Reuters în categoria „System Software”. Făcând această selecţie au rezultat 7.083 documente, care conţin 19.038 trăsături caracteristice fiind clasificate în 68 de clase. Am eliminat clasele slab reprezentate (sub 1% din documente) sau excesiv reprezentate (în mai mult de 99% din documente)

În final am obţinut doar 24 clase distincte cu un număr de 7.053 documente. Având în vedere faptul că 19038 de trăsături este un număr mare utilizând metoda câştigului informaţional „Information Gain”. Astfel, s-a calculat pentru fiecare atribut (trăsătură) valoarea care reprezintă câştigul obţinut în clasificare dacă păstrăm acel atribut. Valorile din vectorii de reprezentare a documentelor au fost normalizate folosind reprezentarea binară prezentată în secţiunea 2.5.4. Valoarea maximă obtenabilă pentru câştigul informaţional este 1. Pentru selectarea doar a atributelor considerate relevante din punct de vedere al câştigului informaţional am impus un prag de 0.01. Astfel au fost selectate un număr de 1309 trăsături din cele 19038 existente, selecţie realizată pe baza valorii descrescătoare a câştigului informaţional.

Pentru antrenarea şi testarea clasificatorilor implementaţi am utilizat următoarele seturi de date:

2.7.1.2 Setul *A1*

Acest set de date conţine 4.702 exemple, 1.309 de atribute şi 24 de clase (topic-uri) şi este de forma [Mora07]:

```
#Samples 4702
#Attributes 1309
#Topics 24

@attribute 1.0
@attribute 2.0
@attribute 3.0
@attribute 4.0
@attribute 5.0
```

```
@attribute 6.0
@attribute 7.0
...
@attribute 1309.0

@topic c18  747
@topic c181  722
@topic c15  3645
@topic c152  2096
@topic c11  626
@topic c14  179
@topic c22  580
@topic gcat  451
@topic c33  529
@topic c31  456
@topic c13  275
@topic c17  448
@topic c171  385
@topic c12  249
@topic gcrim  258
@topic c21  270
@topic c23  152
@topic c41  395
@topic c411  369
@topic ecat  107
@topic m11  131
@topic mcat  137
@topic c151  1630
@topic c1511  410

@data
//urmează primul document în care avem perechi atribut:frecvenţă urmate după # de eticheta clasei
0:1 1:8 6:1 8:5 10:1 11:1 13:1 16:2 30:3 35:19 40:1 42:1 57:5 62:2 63:11 64:1
68:3 71:1 77:3 86:1 95:1 111:2 117:1 118:3 120:1 129:1 136:2 147:1 152:1 159:1 168:1
174:1 177:1 181:1 190:1 191:2 194:1 203:2 220:1 226:1 227:1 232:2 234:1 284:1 288:1
293:1 313:1 317:1 332:1 337:4 340:1 342:1 351:1 352:1 353:1 363:1 375:1 442:1 475:1
476:2 481:1 486:1 488:1 492:2 537:2 541:7 555:1 568:1 570:1 641:2 706:1 725:1 743:1
745:1 872:1 877:1 912:1 949:1 979:3 1029:1 1033:1 1051:1 1150:1 1160:1 # c31
//urmează al doilea document
0:1 1:1 8:5 16:1 18:2 23:1 39:1 41:1 43:1 46:9 48:1 62:1 71:1 73:1 79:1 82:1
93:1 95:1 114:2 122:1 136:1 150:1 154:1 160:1 175:1 184:1 201:1 213:1 217:1 232:1
240:1 242:1 251:1 256:2 266:1 284:2 285:1 338:4 351:1 355:4 359:2 372:2 374:5 382:1
386:1 387:2 424:2 450:2 461:1 467:1 469:1 478:1 481:1 511:3 521:3 532:1 552:1 554:1
574:1 579:4 609:4 612:1 619:1 626:1 655:1 663:1 674:1 689:1 701:1 702:2 705:1 718:1
725:1 748:1 776:1 796:1 879:1 896:2 924:1 979:1 1074:1 1131:2 1162:1 1202:1 1227:1
1263:6 # c14 c17 c171
...
```

2.7.1.3 Setul T1

Setul acesta de date conţine 2.351 de exemple cu 1.309 atribute şi 24 de clase. Este folosit pentru evaluarea (testarea) după antrenare a clasificatorilor [Mora07].

```
#Samples 2351
#Attributes 1309
#Topics 24

@attribute 1.0
@attribute 2.0
...
@attribute 1309.0
@topic c18  747
@topic c181  722
@topic c15  3645
@topic c152  2096
@topic c11  626
@topic c14  179
@topic c22  580
@topic gcat  451
```

```
@topic c33  529
@topic c31  456
@topic c13  275
@topic c17  448
@topic c171  385
@topic c12  249
@topic gcrim  258
@topic c21  270
@topic c23  152
@topic c41  395
@topic c411  369
@topic ecat  107
@topic m11  131
@topic mcat  137
@topic c151  1630
@topic c1511  410

@data

0:1 1:15 2:1 3:12 4:3 5:2 6:2 7:3 8:9 9:2 10:2 11:2 12:3 13:1 14:2 15:3 16:2
17:2 18:1 19:4 20:1 21:2 22:1 23:1 24:2 25:1 26:1 27:2 28:1 29:1 30:1 31:4 32:1 33:1
34:2 35:15 36:1 37:1 38:1 39:1 40:1 41:1 42:1 43:1 44:1 45:2 46:1 47:1 48:2 49:1 50:1
51:2 52:1 53:1 54:1 55:1 56:2 57:1 58:2 59:4 60:2 61:2 62:1 63:4 64:1 65:3 66:2 67:3
68:1 69:1 70:1 71:2 72:1 73:1 74:1 75:1 76:4 77:1 78:1 79:1 80:1 81:1 82:1 83:1 84:1
85:1 86:2 87:1 88:1 89:1 90:1 91:1 92:1 93:1 94:1 95:1 96:1 97:1 98:1 99:1 100:1 101:1
102:2 103:1 104:1 105:1 106:1 107:1 108:1 109:1 110:1 111:1 112:1 113:1 114:1 115:1
116:1 117:1 118:1 119:1 120:1 121:1 122:1 123:1 124:1 125:1 126:1 127:1 # c18 c181
0:1 2:1 18:1 31:1 115:1 128:1 129:2 130:1 131:1 132:1 133:1 134:1 135:1 136:1
137:1 138:1 139:1 # c15 c152

...
```

2.7.1.4 Setul T2

Setul acesta de date conţine 136 de exemple cu 1.309 atribute şi 24 de clase. Acest set conţine acele documentele din setul T1 care nu au putut fi clasificate corect de nici un clasificator selectat în metaclasificatorul prezentat în [Mora07].

```
#Samples 136
#Attributes 1309
#Topics 24

@attribute 1.0
@attribute 2.0
...
@attribute 1309.0
@topic c18  747
@topic c181  722
@topic c15  3645
@topic c152  2096
@topic c11  626
@topic c14  179
@topic c22  580
@topic gcat  451
@topic c33  529
@topic c31  456
@topic c13  275
@topic c17  448
@topic c171  385
@topic c12  249
@topic gcrim  258
@topic c21  270
@topic c23  152
@topic c41  395
@topic c411  369
@topic ecat  107
@topic m11  131
@topic mcat  137
```

```
@topic c151  1630
@topic c1511  410

@data

0:1 1:15 2:1 3:12 4:3 5:2 6:2 7:3 8:9 9:2 10:2 11:2 12:3 13:1 14:2 15:3 16:2
17:2 18:1 19:4 20:1 21:2 22:1 23:1 24:2 25:1 26:1 27:2 28:1 29:1 30:1 31:4 32:1 33:1
34:2 35:15 36:1 37:1 38:1 39:1 40:1 41:1 42:1 43:1 44:1 45:2 46:1 47:1 48:2 49:1 50:1
51:2 52:1 53:1 54:1 55:1 56:2 57:1 58:2 59:4 60:2 61:2 62:1 63:4 64:1 65:3 66:2 67:3
68:1 69:1 70:1 71:2 72:1 73:1 74:1 75:1 76:4 77:1 78:1 79:1 80:1 81:1 82:1 83:1 84:1
85:1 86:2 87:1 88:1 89:1 90:1 91:1 92:1 93:1 94:1 95:1 96:1 97:1 98:1 99:1 100:1 101:1
102:2 103:1 104:1 105:1 106:1 107:1 108:1 109:1 110:1 111:1 112:1 113:1 114:1 115:1
116:1 117:1 118:1 119:1 120:1 121:1 122:1 123:1 124:1 125:1 126:1 127:1 # c18 c181
```

2.7.2 **Setul de date RSS –Web**

Acest set de date a fost utilizat în testele efectuate cu algoritmii de clustering. Deoarece algoritmii de clustering vor funcţiona pentru gruparea unor ştiri provenite din fluxuri de date RSS am creat mai multe seturi de date care conţin astfel de informaţii. Am utilizat pentru crearea seturilor de date ştiri din fluxuri RSS obținute de pe site-ul agenției Reuters și al agenției BBC. Știrile au fost alese din mai multe fluxuri rss (fișiere xml) preclasificate de către agenții din 3 zile consecutive. Am ales ştiri din categoriile: *Lybia, Motorsport, Japan, Israel, Syria, Yemen, FotbalEuropean.* Din fișierele xml am extras doar conținutul nodului <*title*> care conține titlul știrii și conținutul nodului <*description*> în care se găsește textul știrii.

Fiecare ştire a fost salvată într-un fișier text separat, iar ştirile dintr-o categorie au fost toate salvate în același subdirector. În prima fază de preprocesare s-au eliminat caracterele speciale care apar în ştire din motive de interpretor web și în final în fișierele text pe prima linie a rămas titlul ştirii iar pe liniile următoare conținutul – fraze separate prin punct. Am păstrat categoria stabilită deja de agențiile de ştiri din care face parte ştirea pe care o vom folosi în final în etapa de evaluare a rezultatelor clusteringului.

În primele experienţe efectuate cu algoritmii HAC şi *k*-Medoids dar şi cu alte tipuri de algoritmi de clustering care nu au fost prezentaţi special în această lucrare (Expectation Maximisation - EM şi COWEB din pachetul WEKA 3.7) am observat că tendinţa tuturor algoritmilor a fost de a crea un cluster de dimensiune foarte mare şi câţiva clusteri care conţineau câte un document. Astfel, am analizat fiecare categorie în parte furnizată de fluxurile de ştiri reprezentând-o ca o dendogramă cu ajutorul algoritmului HAC şi am constatat că în fiecare categorie existau documente (puţine) care erau unite de algoritm la o distanţă foarte mare în raport cu celelalte documente. Acele documente au fost eliminate din setul de date. Analizându-le am sesizat că ele erau corect grupate de către agenţia de ştiri dar utilizau multe sinonime ale cuvintelor folosite de celelalte documente din acea categorie. În viitor ne gândim şi la o abordare care să ia în considerare şi sinonime de cuvinte în momentul în care se construieşte arborele de sufixe.

De pe fluxurile de ştiri am extras ştirile din 3 zile consecutive obţinând 212 de documente care au fost grupate iniţial în 7 categorii. Am creat din documentele astfel extrase mai multe seturi de date care diferă prin numărul de categorii şi numărul de documente care vor fi utilizate pentru evaluarea algoritmilor de clustering. În seturile create am păstrat categoriile furnizate de către agenţiile de ştiri. Astfel s-au creat 7 seturi de date distincte prezentate în continuare:

Denumirea setului	Nr. categorii	Nr. total documente	Structura	
			Denumire categorie	Nr. documente
S01	3	151	FotbalEuropean	47
			Lybia	62
			Motorsport	42
S02	4	172	FotbalEuropean	38
			Japan	37
			Lybia	57
			Motorsport	40
S03	4	156	FotbalEuropean	47
			Israel	8
			Lybia	59
			Motorsport	42
S04	5	177	FotbalEuropean	38
			Japan	37
			Lybia	56
			Motorsport	40
			Israel	6
S05	5	193	FotbalEuropean	47
			Japan	37
			Lybia	59
			Motorsport	42
			Israel	8
S06	6	55	Japan	9
			Lybia	12
			Motorsport	20
			Israel	9
			Syria	3
			Yemen	2
S07	6	179	FotbalEuropean	38
			Japan	37
			Lybia	56
			Motorsport	40
			Israel	6
			Yemen	2

Tabel 2.4 Seturi de date pentru clustering

Partea a II-a. Clustering

... Cum nu vii tu, Ţepeş doamne, ca punând mâna pe ei,
Să-i împarţi în două cete: ...
Mihai Eminescu, Scrisoarea III

3 Algoritmi de clustering. Paradigma actuală

În continuare voi prezenta o taxonomie a celor mai importanţi algoritmi utilizaţi pentru clustering. Pentru fiecare algoritm am selectat versiunea comună, care să reprezinte întreaga familie.

3.1 *O posibilă taxonomie*

Aranjarea datelor în diferite grupuri se poate face utilizând diverse strategii. O posibilă strategie, conform [Berk06], [Zhao04] împarte algoritmii de clustering în două categorii mari: tehnici de clustering bazate pe partiţionarea datelor şi tehnici de clustering bazate pe metode ierarhice. Pe lângă aceste două mari categorii „clasice" algoritmi mai noi se pot grupa şi în algoritmi bazaţi pe densitate, algoritmi bazaţi pe reţele şi algoritmi bazaţi pe modele [Han01], [Cret11_b].

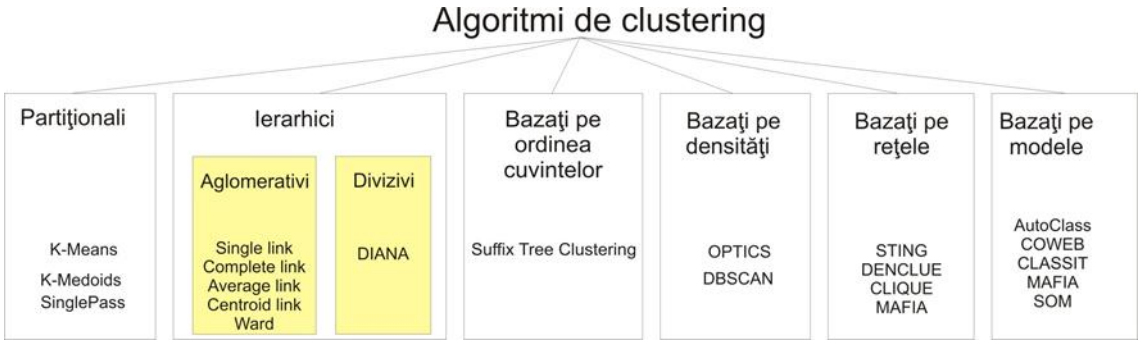


Fig. 3.1 Posibilă taxonomie a algoritmilor de clustering

3.1.1 Algoritmi partiţionali (sau metode partiţionale)

Daca se consideră n obiecte care trebuie grupate în *k* grupe atunci o metodă de partiţionare construieşte *k* partiţii din cele *n* obiecte fiecare partiţie reprezentând un cluster cu $k \leq n$. Clusterii se formează ţinând cont de optimizarea unui criteriu (funcţii) obiectiv denumit şi funcţie de disimilariate, ca de exemplu distanţa, astfel încât obiectele dintr-un cluster sunt similare iar obiectele din clusteri diferiţi sunt disimilare. Această grupare trebuie să satisfacă două condiţii:

- Fiecare cluster trebuie să conţină cel puţin un obiect.
- Fiecare obiect trebuie să fie inclus într-un singur cluster.

Ideea de bază utilizată de acest tip de metode este următoarea: iniţial se dă numărul *k* care reprezintă numărul de partiţii (clusteri) care se doresc a fi obţinute iar apoi se aplică o metodă de partiţionare care creează *k* clusteri.

În cadrul acestei metode se mai utilizează şi tehnica relocării iterative prin care se încearcă îmbunătăţirea partiţionării prin mutarea obiectelor dintr-un grup în altul. Criteriul de mutare este ca în clusterii rezultaţi obiectele din acelaşi cluster să fie asemănătoare (apropiate). Există diferite criterii pentru a evalua calitatea clusterilor. Aceste criterii au fost prezentate în secţiunea 2.6.

În cadrul metodelor de partiţionare putem identifica trei tipuri de algoritmi:

- algoritmi de clustering de tip k-means;
  - la acest tip de algoritmi fiecare cluster este reprezentat de valoarea medie a obiectelor (distanţei dintre obiectele clusterului) din cadrul clusterului respectiv;
- algoritmi de clustering de tip *k*-Medoids;
  - la acest tip de algoritmi fiecare cluster este reprezentat de obiectul care se află cel mai aproape de centrul de greutate al clusterului;
- algoritmi pentru clustering probabilistic;
  - metodele probabilistice pleacă de la premisa că datele provin dintr-o mixtură de populaţii a căror distribuţii şi probabilităţi trebuiesc determinate.

Aceşti algoritmi pot fi utilizaţi în colecţii de date mici până la mijlocii şi găsesc clusteri de tip sferici.

Voi prezenta în secţiunea următoare câteva metode de partiţionare relevante şi/sau consacrate de clustering.

3.1.1.1 Metoda k-Means

Algoritmul K-Means [MacQ67], [Hart75], [Hart79] este cel mai popular şi mai utilizat algoritm în aplicaţii ştiinţifice şi industriale. Numele acestui algoritm provine de la reprezentarea a *k* clusteri C_j a căror valoare medie (mean) este c_j calculată ca şi centrul de greutate al punctelor din C_j (al centroidului). Valoarea parametrului *k* se stabileşte la începutul parcurgerii algoritmului. Similaritatea elementelor dintr-un cluster C_j se calculează raportat la centrul de greutate al centroidului. Criteriul utilizat de obicei pentru evaluarea clusteringului este eroarea medie pătratică definită prin formula:

$$E(C) = \sum_{j=1}^{k} \sum_{x_i \in C_j} \left\| x_i - c_j \right\|^2 \qquad (3.1)$$

unde *x* este punctul din spaţiu care reprezintă obiectul dat iar c_j centrul de greutate al lui C_j.

Scopul este de a obţine o similaritate intra-cluster mare simultan cu o similaritate inter-cluster foarte mică astfel încât să obţinem *k* clusteri cât mai compacţi şi, totodată, cât mai separaţi posibil.

Este evident că acest tip de algoritm funcţionează cu variabile numerice. Paşii pe care algoritmul îi parcurge ar putea fi:

1. Se generează aleator *k* centre în spaţiul *n*-dimensional, care reprezintă iniţial centroizii (numărul de clusteri care se doresc a fi obţinuţi).
2. Pentru fiecare document se calculează distanţa faţă de centroizi iar documentul curent se introduce în clusterul corespunzător centroidului faţă de care este obţinută distanţa minimă. În ceea ce priveşte distanţa există mai multe abordări; în funcţie de abordarea utilizată se particularizează algoritmul folosit. De exemplu, foarte des folosite sunt: distanţa euclidiană, distanţa Manhattan sau distanţa Minkovski.

3. După ce toate documentele au fost asignate corespunzător clusterilor, se recalculează poziţia celor *k* centroizi ca fiind centrul de greutate al tuturor eşantioanelor atribuite fiecărui cluster în parte.
4. Se repetă paşii 2 şi 3 până când poziţia centroizilor nu se mai modifică semnificativ (practic până în momentul în care documentele grupate într-o iteraţie nu se mai redistribuie).
5. Documentele asignate centroizilor formează documentele celor *k* clusteri construiţi.

Există şi variante în care în primul pas se aleg aleator *k* documente din cele *N* disponibile în setul de date unde N>>k.

Algoritmul în pseudocod poate fi reprezentat astfel:

```
Intrări:
X = {x1, ...,xn} (obiecte de clusterizat)
k (numărul clusterilor de obţinut)

Ieşiri:
C = {c1, ...,ck} (centroizii clusterilor)
m:X→C (apartenenţa la un cluster)

algoritmul KMeans
        Iniţializează C (aleator)
        for each xj∈ X
                m(x_j) = arg min_{n=1,k} distanta(x_j, c_n)
        end
        while m se modifică
                for each i ∈ {1,k}
                        recalculează ci ca centroid {c|m(x) = i}
                end
                for each xj∈ X
                m(x_j) = arg min_{n=1,k} distanta(x_j, c_n)
                end
        end
        return C
```

Fig. 3.2 Pseudocod pentru algoritmul K-Means

În figurile următoare sunt prezentaţi paşii parcurşi de algoritmul k-Means folosind un set de 50 de puncte generate aleator utilizând o distribuţie gaussiană [CLU]. Pentru calcularea distanţei în Fig. 3.3 s-a utilizat distanţa euclidiană, în Fig. 3.4 s-a utilizat distanţa Manhattan iar în Fig. 3.5 s-a utilizat distanţa Minkovski.

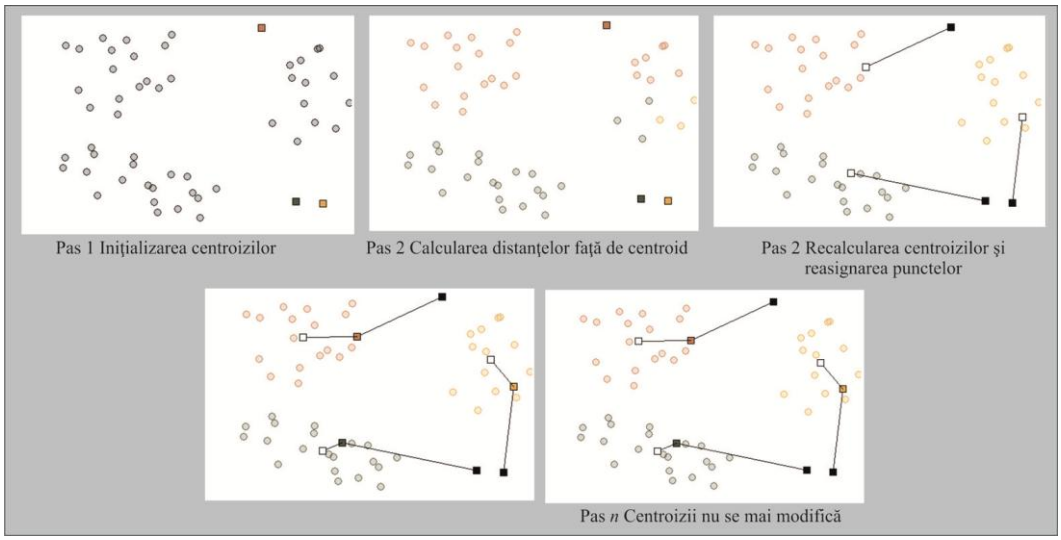


Fig. 3.3 K-Means,- distanţă euclidiană

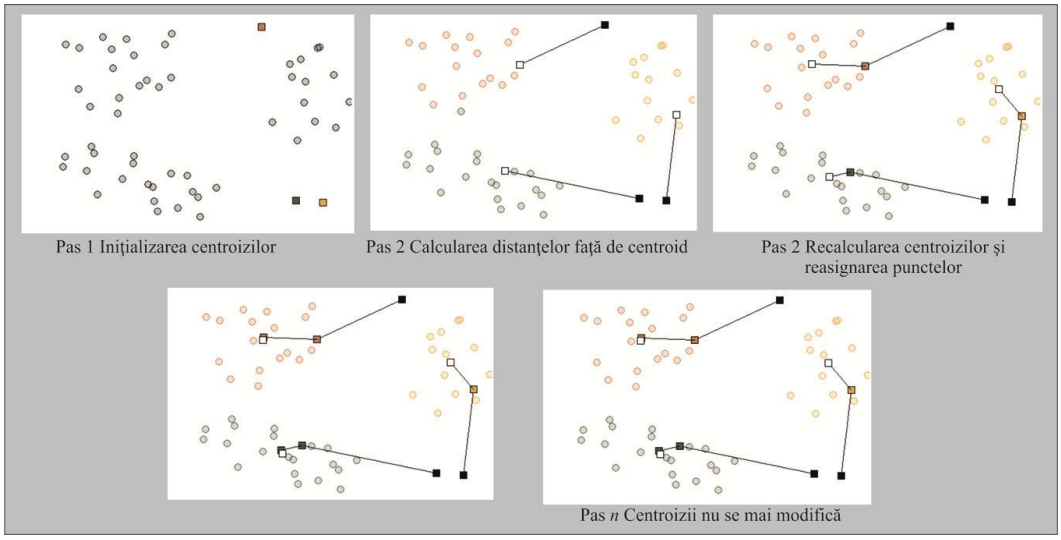


Fig. 3.4 K-Means,- distanţă Manhattan

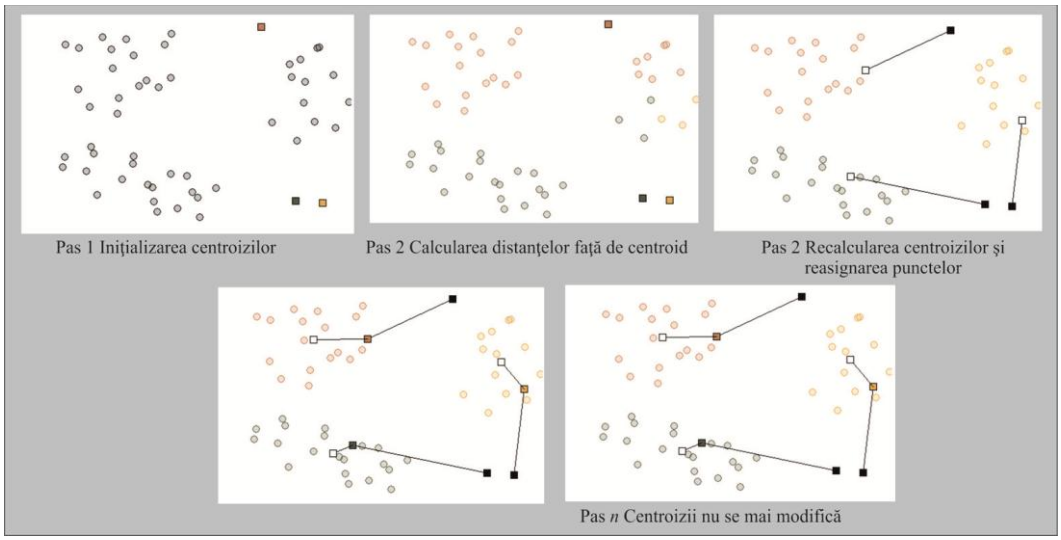


Fig. 3.5 K-Means, - distanţă Minkovski

Algoritmul prezentat mai sus este scalabil şi destul de eficient deoarece complexitatea computaţională este *O(nkt)* unde *n* este numărul total de obiecte, *k* este numărul clusterilor iar *t* este numărul iteraţiilor. Având în vedere faptul că algoritmul utilizează centrul de greutate al centroidului, evident că acesta poate fi aplicat doar datelor numerice.

Algoritmul K-Means are totuşi şi unele dezavantaje:

- Rezultatul depinde de alegerea iniţială a numărului de clusteri (*k*);
- Optimul local calculat poate diferi faţă de optimul global în unele cazuri;
- Nu este evident care ar fi valoarea optimă pentru *k*;
- Algoritmul este sensibil la zgomot;
- Acceptă doar variabile numerice;
- Clusterii rezultaţi pot fi „dezechilibraţi” chiar în unele cazuri fără să conţină ceva.

În [Brad98] se propune o variantă de a diminua efectele produse de iniţializarea algoritmului prin alegerea numărului *k*. Astfel, algoritmul k-Means se aplică pe eşantioane mici din mulţimea datelor alegând centroizii în jurul maximelor funcţiei de densitate a probabilităţilor. Apoi această iniţializare se aplică în clusteringul tuturor datelor. Zhang [Zhan01] a propus o altă variantă de a optimiza algoritmul prin faptul că punctele sunt asignate „soft” clusterilor cu „ponderi” apropiate.

O soluţie simplă ar fi să comparăm rezultatele obţinute în urma rulării multiple a algoritmului cu valori diferite pentru *k* şi alegerea celei mai bune soluţii conform cu un criteriu dat. Totuşi, când cantitatea de date este mare, această soluţie ar fi mare consumatoare de timp pentru a rulări multiple cu k diferit.

3.1.1.2 Metoda *k*-Medoids

Algoritmul k-Medoids reprezintă o adaptare a algoritmului k-Means. În loc să se calculeze centrul de greutate din fiecare grup, se alege un element reprezentativ, sau medoid, pentru fiecare cluster la fiecare iteraţie. Medoidul pentru fiecare grup se calculează prin găsirea unui element *i* din cluster care minimizează costul:

$$\sum_{j \in C_k} d(i, j) \qquad (3.2)$$

unde C_k este clusterul care conţine obiectul *i* iar *d(i,j)* distanţa dintre obiectul *i* şi obiectul *j*.

Există două avantaje prin utilizarea obiectelor existente ca centre de clusteri. În primul rând, un medoid poate să descrie în mod util clusterul. În al doilea rând, nu este necesară calcularea repetată a distanţelor la fiecare iteraţie, deoarece se pot calcula o dată şi se salvează apoi într-o matrice de distanţe.

Paşii din cadrul algoritmului k-Medoids pot fi rezumaţi după cum urmează:

1. Alege *k* obiecte la întâmplare să fie medoizii iniţiali ai clusterilor.
2. Atribuie fiecare obiect la clusterul asociat medoidului cel mai apropiat şi calculează costul conform formulei 3.2 (unde costul reprezintă distanţa faţă de acel medoid).
3. Recalculează poziţiile medoizilor *k*.

Se repetă paşii 2 şi 3 până când medoizii nu se mai modifică semnificativ sau costul nu mai scade

Algoritmul *k*-Medoids este eficient pe seturi de date mici (100 documente), în schimb pe seturi de date mari, datorită costului computaţional mare, devine destul de lent. Pentru a putea aplica algoritmul PAM (Partitioning Around Medoids) la seturi de date mari Kaufman şi Rousseew [Kauf90] au dezvoltat algoritmul CLARA (Clustering Large Applications) care se bazează pe aplicarea algoritmului PAM pe un eşantion de date dintr-un set dat. CLARA va descoperi astfel k medoizi din eşantionul dat. Dacă eşantionul dat şi ales aleator va reprezenta corect întregul set de date atunci medoizii descoperiţi vor fi identici cu cei din întregul set de date. Complexitatea fiecărei iteraţii devine $O(kS^2+k(k-n))$ unde *S* este dimensiunea eşantionului ales, *k* numărul clusterilor iar *n* numărul total de puncte. În [Kauf90, Ng94, Wei03] se propune alegerea valorii $S=40+2k$.

Pseudocodul pentru algoritmul CLARA este următorul:

Algoritm CLARA

```
Input: S eşantion din setul D, S = {O_i}_{i=1,m} unde m este dimensiunea lui S
k număr de clusteri
iniţializează mincost cu MAXINT
Repeat m ori
      selectează k obiecte arbitrar din S
      generează setul de medoizi M din S aplicând algoritmul PAM
            if Cost(M, D) <mincost
            then
                  mincost = Cost(M, D);
                  bestset = C;
            End-if;
End-repeat;
Return bestset;
```

unde,

$$Cost(M,D) = \frac{\sum_{i=1}^{n} d(O_i, ret(M,O_i))}{n} \quad (3.3)$$

cu M setul de medoizi, $d(O_i, O_j)$ este distanţa dintre obiectele O_i, şi O_j şi $ret(M,O_i)$ returnează un medoid din M care este cel mai apropiat de O_i

Pentru a îmbunătăţi calitatea şi scalabilitatea algortimului CLARA în [Ng94] se propune algoritmul CLARANS (Clustering Large Applications based upon RANdomized Search). În algoritmul CLARANS găsirea a k medoizi este văzută ca şi căutarea într-un graf. În acest graf, un nod este reprezentat de un set de k obiecte $\{O_1, ..., O_k\}$, indicând faptul că $O_1, ..., O_k$ sunt medoizii aleşi. Două noduri sunt vecine (de exemplu, conectate printr-un arc) în cazul în care seturile lor diferă cu un singur obiect. Este evident că fiecare nod în graf are $k(n-k)$ vecini. Din moment ce un nod reprezintă o colecţie de k medoizi, fiecare nod corespunde unui posibil rezultat al clusteringului a cărui calitate este măsurată prin funcţia de cost definită în (3.3).

În locul folosirii unei strategii de căutare exhaustive, CLARANS adoptă o strategie de căutare euristică pe baze de selecţii randomizate. CLARANS porneşte de la un nod arbitrar în graf şi selectează aleatoriu unul dintre vecinii săi. În cazul în care costul obţinut prin selectarea vecinului este mai mic decât cea a nodului curent, CLARANS reporneşte de la acest vecin şi continuă selecţia altor vecini. În caz contrar, CLARANS examinează la întâmplare un alt vecin până când se găseşte un vecin mai „bun" sau este atins numărul prestabilit de vecini. În acest ultim caz, nodul curent este declarat a fi un minim local. Pentru a evita să se oprească la o soluţie neoptimă, CLARANS repeta procesul de căutare dintr-un nod iniţial diferit pentru un număr predeterminat de ori. Ulterior, nodul cu costul minim este selectat ca grupare finală.

Fie *maxneighbor* numărul maxim de vecini pentru a examina şi *numlocal* numărul de minimele locale obţinute. Algoritmul CLARANS este reprezentat în figura de mai jos.

Algoritm CLARANS

```
iniţializează minCost cu un număr mare (9999);
For i = 1 to numLocal do
      selectează aleator un nod ca nod curent C în graf;
      iniţializează j = 1;
      Repeat
            alege aleator un vecin N al lui C;
            If Cost(N, D) < Cost(C, D)
                  Then
                        N devine nodul curent C;
                        Reset j to 1;
            Else
                  j=j+1;
            End-if;
      Until( j > maxNeighbor);

      If Cost(C, D) < minCost
            Then
                  minCost = Cost(C, D);
                  bestNode = C;
      End-if;
End-for;
Return bestNode;
```

În [Ng02] se recomandă ca parametrul *numLocal* să fie iniţializat cu 2 şi parametrul maxNeighbor cu valoarea: $\frac{1{,}25 \cdot k(n-k)}{100}$

Faţă de algoritmul CLARA care la un anumit stadiu al căutării foloseşte un eşantion fix, algoritmul CLARANS extrage un eşantion cu un grad aleator mai mare. Procesul de clustering poate fi privit ca şi căutarea într-un graf unde fiecare nod este o soluţie potenţială (un set de medoizi).

3.1.2 Metode ierarhice

Metodele ierarhice realizează o structurare ierarhică a setului de obiecte. Există două tipuri de abordări:

- aglomerative: au o abordare „*bottom-up*”; la început fiecare obiect este asignat unui cluster, apoi clusterii se unesc pas cu pas pe baza unor măsuri de similaritate până când toţi sunt uniţi într-unul singur sau până la o condiţie de oprire dată, creându-se o structură ierarhică.
- divizive: au o abordare „*top-down*”, la început toate obiectele sunt considerate a fi conţinute într-un singur cluster, apoi prin iteraţii succesive se divide fiecare cluster în clusteri mai mici până când fiecare obiect devine un cluster sau până la o condiţie de oprire dată.

O problemă majoră a algoritmilor de clustering ierarhic constă în faptul că o dată ce pasul de divizare sau de unire s-a efectuat acesta nu mai poate fi anulat. Această problemă reprezintă totodată un avantaj major al acestui tip de algoritmi datorită reducerii numărului de calcule evitându-se calcularea diferitelor combinaţii de posibilităţi.

Algoritmii BIRCH (Balanced Iterative Reducing and Clustering) propus în [Zhan96] şi CURE (Clustering Using REpresentatives) propus în [Guha98] combină metodele de partiţionare cu cele de tip ierarhic.

3.1.2.1 Algoritmi aglomerativi ierarhici (HAC)

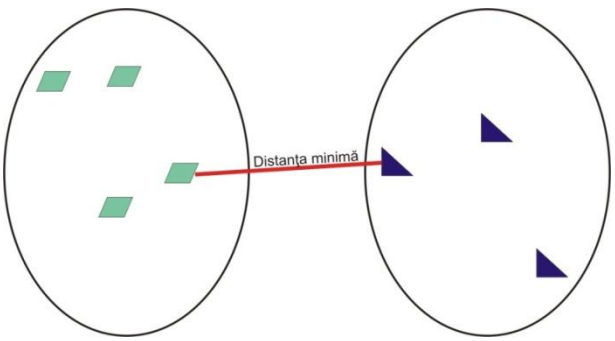
În algoritmul de clustering aglomerativ ierarhic (Hierarchical Clustering Algorithm - HAC) fiecare document este considerat la început, ca formând un cluster apoi pe următorul nivel (un nivel superior mai general) câte doi clusteri sunt uniţi pe baza similarităţii lor. Algoritmul se repetă până când **toţi clusterii** sunt uniţi într-unul singur. Disimilaritatea clusterilor se exprimă ca şi distanţa între doi clusteri. Distanţa se poate exprima pe baza distanţei euclidiene sau pe baza cosinusului unghiului între cei doi vectori care modelează clusterii.

3.1.2.1.1 Disimilaritatea dintre doi clusteri în algoritmi ierarhici aglomerativi

Deşi în cele mai multe ori este utilizată distanţa între clusteri pentru algoritmii ierarhici aglomerativi voi prezenta cazul general bazat pe disimilaritate. În funcţie de modalitatea de calcul a disimilarităţii se disting mai multe modele ale acestei categorii de algoritmi de clustering [Mann09], [Mann08]:

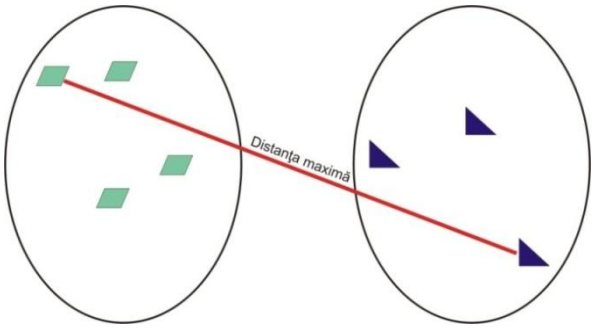
3.1.2.1.1.1 Single link

Disimilaritatea între doi clusteri este dată de disimilaritatea celor mai similare două documente din cei doi clusteri (distanţa minimă dintre două documente conţinute în acei clusteri). Adică, dacă pentru doi clusteri A şi B cu documentul $a \in A$ şi documentul $b \in B$ atunci:



$$disim(A,B) = \min_{a \in A, b \in B} disim(a,b) \qquad (3.4)$$

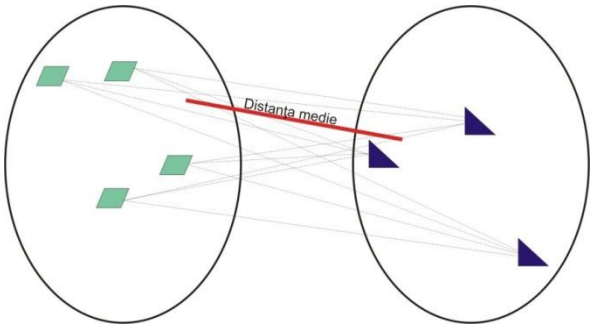
3.1.2.1.1.2 Complete link



Disimilaritatea a doi clusteri este dată de disimilaritatea a celor mai disimilare documente din cei doi clusteri (distanţa maximă dintre două documente conţinute în acei clusteri). Adică, dacă pentru doi clusteri A şi B cu a∈A şi b∈B atunci:

$$disim(A,B) = \max_{a\in A, b\in B} disim(a,b) \qquad (3.5)$$

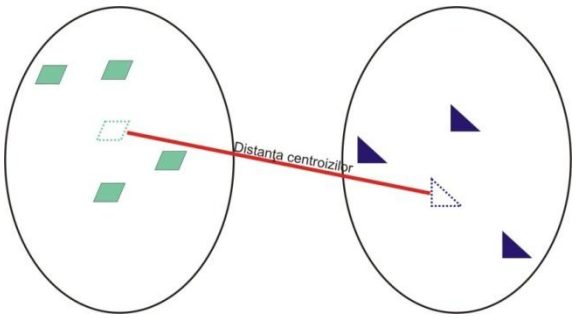
3.1.2.1.1.3 Average link



Disimilaritatea dintre doi clusteri este calculată ca media disimilarităţii documentelor din cele doi clusteri (media distanţelor dintre fiecare element dintr-un cluster şi toate elementele din celălalt cluster).

$$disim(A,B) = \frac{1}{|A||B|}\sum_{a\in A, b\in B} disim(a,b) \qquad (3.6)$$

3.1.2.1.1.4 Centroid link



Disimilaritatea dintre doi clusteri este calculată ca disimilariatea între centroizii clusterilor (distanţa dintre centroizii clusterilor).

$$disim(A,B) = \|C_A - C_B\| \qquad (3.7)$$

3.1.2.1.1.5 Metoda lui Ward

Metoda lui Ward [Ward63] determină distanţa dintre doi clusteri ca fiind valoarea creşterii sumei pătratelor distanţelor când se unesc cei doi clusteri. Creşterea sumei pătratelor distanţelor $\Delta(A,B)$ este dată de formula (3.8)

$$\Delta(A,B) = \sum_{i\in A\cup B}\|x_i - m_{A\cup B}\|^2 - \sum_{i\in A}\|x_i - m_A\|^2 - \sum_{i\in B}\|x_i - m_B\|^2 = \frac{n_a\cdot n_b}{n_a+n_b}\|m_A - m_B\|^2 \qquad (3.8)$$

unde m_j este centroidul clusterului j şi n_j este numărul punctelor din clusterul j.

La începutul algoritmului, utilizând această metodă, $\Delta(A,B)$ este zero deoarece fiecare cluster este format dintr-un singur obiect. Pe măsură ce se unesc clusteri, $\Delta(A,B)$ creşte iar metoda lui Ward descrisă prin ecuaţia (3.8) încearcă să minimizeze această valoare. În metoda lui Ward în cazul în care două perechi de clusteri au centrele la distanţe identice se vor uni clusterii mai mici.

3.1.2.1.1.6 SAHN (Sequential, Agglomerative, Hierarchical and Nonoverlapping)

În [Jain88] Jain şi Dubes prezintă o formulă de calcul generalizată prezentată de Lance şi Williams în [Lanc67] a distanţei unui cluster faţă de un cluster nou unit. Astfel, calcularea distanţei $d_{A\to(B,C)}$ unui cluster *A* cu n_k puncte faţă de un cluster nou unit *(B,C)* se calculează după formula:

$$d_{A\to(B,C)} = \alpha_B \cdot d_{A\to B} + \alpha_C \cdot d_{A\to C} + \beta \cdot d_{B\to C} + \gamma \cdot \left| d_{A\to B} - d_{A\to C} \right| \qquad (3.9)$$

unde coeficienţii α, β, γ sunt daţi în tabelul (Tabel 3.1)

Metoda de clustering	α_B	α_C	β	γ
Single link	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
Complete link	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
Average link – neponderat (Unweighted pair group method average)	$\frac{n_B}{n_B + n_C}$	$\frac{n_C}{n_B + n_C}$	0	0
Average link –ponderat (Weighted pair group method average)	$\frac{1}{2}$	$\frac{1}{2}$	0	0
Centroid link - neponderat (Unweighted pair group method centroid)	$\frac{n_B}{n_B + n_C}$	$\frac{n_C}{n_B + n_C}$	$\frac{-n_B \cdot n_C}{(n_B + n_C)^2}$	0
Centroid link - ponderat (Weighted pair group method centroid)	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	0
Metoda lui Ward	$\frac{n_B + n_A}{n_B + n_C + n_A}$	$\frac{n_A + n_A}{n_B + n_C + n_A}$	$\frac{-n_A}{n_B + n_C + n_A}$	0

Tabel 3.1 - Coeficienţii formulei generalizate pentru calculul distanţei dintre un cluster şi un alt cluster nou format

O variantă ar fi ca să utilizăm metoda lui Ward pentru a calcula „costul" unirii clusterilor. Micşorăm k – numărul clusterilor (k iniţial este egal cu numărul de documente) până când costul $\Delta(A,B)$ înregistrează o creştere mare. Atunci, se presupune că prin unire s-ar pierde prea multă diferenţă specifică şi atunci s-ar putea opri algoritmul la numărul de clusteri obţinuţi înainte de creşterea costului. Această metodă poate fi combinată cu utilizarea unui algoritm partiţional - k-Means de exemplu - iar iniţializarea acestui algoritm făcându-se cu numărul de clusteri *k* obţinuţi prin metoda lui Ward. În cercetările efectuate am utilizat algoritmul HAC cu metoda single link deoarece am dorit să obţin clusteri cât mai mici şi diferiţi.

3.1.2.2 Algoritmul BIRCH

Algoritmul BIRCH (Balanced Iterative Reducing and Clustering) [Zhan96] este un algoritm hibrid (incremental şi ierarhic) pentru seturi de documente mari şi utilizează o structură care se va salva în memoria principală.

Structura are la bază două concepte: Clustering Feature şi CF-Tree.

Clustering Feature (CF) este un triplet care sumarizează informaţia unui subcluster de puncte. Astfel, fiind date N puncte într-un spaţiu n-dimensional, $\{X_i\}$este un subcluster, şi pentru acest subcluster CF se defineşte ca:

$$CF = (N, \vec{LS}, SS) \tag{3.10}$$

unde

N este numărul de puncte din subcluster

$\vec{LS}$ este suma liniară a N puncte adică $\sum_{i=1}^{N} \vec{X}_i$

SS este suma pătratelor punctelor adică $\sum_{i=1}^{N} \vec{X}_i^{\,2}$

Clustering Feature Tree este un arbore identic cu un arbore de tip B+. În acest arbore există *frunze* - care conţin mai multe intrări şi care reprezintă un cluster care la rândul său poate fi constituit din subclustere ale acestor intrări şi *noduri* care reprezintă un cluster format din clusterii din nodurile copil. Introducerea datelor în arbore se efectuează dinamic în funcţie de doi parametri: factorul de ramificare B şi pragul T pentru dimensiunea unei intrări (pragul pentru diametrul maxim al unei intrări).

Nodurile din arborele CF pot avea maxim B copii şi un maxim de intrări L pentru frunze.

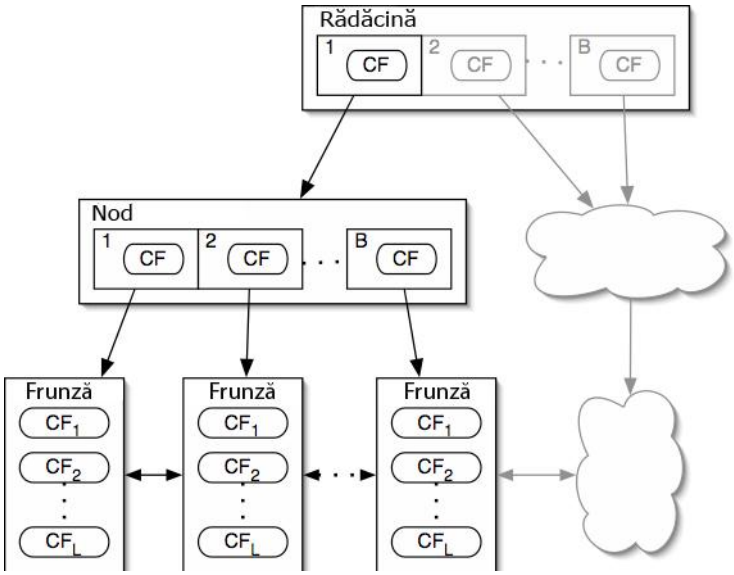


Fig. 3.6 - CF-Tree

Introducerea unui nou element în arbore se face într-un nod frunză, în cel mai apropiat subcluster pentru elementul respectiv atâta timp cât se respectă pragul T (diametrul subclusterului). În cazul în care se depăşeşte acest prag, se creează un CF nou şi se introduce punctul nou. Apoi, se actualizează toate nodurile CF de la rădăcină la frunze care sunt afectate de schimbarea produsă în arbore.

Având în vedere faptul că arborele se salvează direct în memorie, se poate întâmpla ca pentru un prag T să nu fie suficientă memorie şi atunci mărind pragul T se obţine o micşorare a arborelui.

Algoritmul BIRCH se efectuează în două etape:

- Construirea arborelui CF, care poate fi privită ca o compresie a datelor dar care păstrează structurarea datelor;
- Aplicarea unui algoritm de clustering (ierarhic) pentru nodurile frunză din arborele CF.

BIRCH este un algoritm de clustering scalabil şi obţine rezultate bune când clusterii sunt sferici. Dacă clusterii au forme arbitrare, atunci datorită faptului că se utilizează centroizii în calcularea CF, rezultatele algoritmului BIRCH sunt mai slabe.

3.1.2.3 Algoritmul CURE

Majoritatea algoritmilor de clustering favorizează clusterii de formă sferică sau de dimensiuni asemănătoare dar sunt sensibili la prezenţa zgomotului. Un algoritm interesant CURE (Clustering Using REpresentatives) prezentat în [Guha98] integrează metode de clustering ierarhic cu cele partiţionale şi elimină problema favorizării clusterilor de formă sferică sau dimensiune asemănătoare.

CURE utilizează un număr fix de **puncte reprezentative** pentru un cluster în locul centroizilor clasici.

3.1.2.4 Algoritmi divizivi

Se porneşte de la un cluster care este format din toate documentele, apoi printr-o metodă de divizare aplicată recursiv se va ajunge până la clusteri formaţi dintr-un singur document.

Clustering-ul ierarhic diviziv este din punct de vedere conceptual mai complex. Dacă în prima etapă a unei metode aglomerative, considerăm toate unirile posibilele a două obiecte ajungem la un număr de $\frac{n \cdot (n-1)}{2}$ combinaţii iar în cazul utilizării metodei ierarhic divizive există $2^{n-1}-1$ posibilităţi de a împărţi datele în două grupuri. Acest număr este considerabil mai mare decât în cazul unei metode aglomerative.

Kaufmann şi Rousseeuw în [Kauf90] au introdus un algoritm de clustering diviziv DIANA (Divisiv Analysis). Acest algoritm utilizează acelaşi principiu ca şi algoritmul AGNES (AGglomerative NESting - prezentat amănunţit în secţiunea 3.2) doar că porneşte de la un cluster care conţine toate obiectele şi împarte clusterul iniţial în doi subclusteri şi se repetă până când toate obiectele ajung să fie în câte un cluster sau până la o condiţie de terminare dată.

3.1.3 Metode bazate pe ordinea cuvintelor

3.1.3.1 Suffix Tree Clustering (STC)

Acest algoritm, prezentat în [Zami98], nu necesită o reprezentare vectorială a datelor. Se va utiliza pentru reprezentarea datelor într-un arbore de sufixe. Astfel, algoritmul va ţine cont de ordinea atributelor din date şi va crea clusteri în funcţie de atribute sau grupuri de atribute consecutive care apar în setul de date. Astfel, acesta poate reprezenta un avantaj major deoarece ţine cont de ordinea atributelor.

Un arbore de sufixe al unui document *d* este un arbore compact care conţine toate sufixele din acel document. În cazul nostru, un sufix reprezintă un şir format din unul sau mai multe atribute.

3.1.3.1.1 Pas1: Construcţia arborelui de sufixe

Pentru a construi arborele de sufixe se parcurg următoarele etape:

a. Se delimitează propoziţiile (frazele) din document prin adăugarea unui terminator (de exemplu caracterul $);
b. Se creează un nod rădăcină – notat „NULL”;

c. Se extrag de la coadă la cap câte un sufix (unul sau mai multe cuvinte) din propoziţie (frază);
d. Se verifică dacă primul sau primele cuvinte din sufix au deja arc care porneşte din nodul rădăcină;
    i. Dacă DA, atunci se foloseşte arcul existent şi se adaugă un nou nod din nodul curent pentru partea necomună, în cazul în care există. Dacă nu există parte necomună, atunci se introduce în frunza în care am ajuns şi identificatorul pentru documentul curent;
    ii. Dacă NU, se introduce un nou arc etichetat cu sufixul curent;
e. Dacă mai sunt sufixe neexplorate se sare la pasul c.

3.1.3.1.2 Pas 2: Selectarea nodurilor de bază

Selectarea nodurilor de bază care vor deveni clusteri de bază:

a. Fiecare nod din arborele de sufixe care are cel puţin doi copii devine cluster de bază şi primeşte un scor S(B) conform ecuaţiei (3.11);
b. Se ordonează în ordine descrescătoare clusterii de bază pe baza scorului obţinut şi se reţin primii *k* (de obicei k=500 sau un alt prag). Aceşti clusteri de bază vor fi utilizaţi în continuare în Pasul 3;
c. Nodurile care obţin un scor mai mic decât un prag prestabilit sunt eliminate;

Formule utilizate:

Fie $|B|$ numărul de documente conţinute în nodul *B şi* fie $|P|$ numărul de cuvinte din propoziţia (fraza) *P*.

Scorul unui nod se calculează conform:

$$S(B) = |B| \cdot f(|P|) \tag{3.11}$$

unde funcţia de ponderare pentru lungimea unei propoziţii este:

$$\mathrm{f}(|P|) = \begin{cases} 0.5, & |P| = 1 \\ |P|, & 2 \le |P| \le 6 \\ 6, & |P| > 6 \end{cases} \tag{3.12}$$

Această funcţie penalizează propoziţiile (frazele) formate dintr-un singur cuvânt, este liniar crescătoare pentru propoziţiile formate din două până la şase cuvinte şi devine constantă pentru propoziţii mai lungi de şase cuvinte.

Similaritatea dintre doi clusteri se calculează:

$$SIM(B_i, B_j) = \begin{cases} 1, & \left(\frac{|B_i \cap B_j|}{|B_i|} > \alpha\right) \wedge \left(\frac{|B_i \cap B_j|}{|B_j|} > \alpha\right) \\ 0, & \text{altfel} \end{cases} \tag{3.13}$$

unde α reprezintă un prag ales.

În [Zami98] pragul pentru α este 0.5 iar în [Wen09] acesta este 0.8. În ceea ce priveşte formula de calcul a similarităţii, fiind vorba de mulţimi, am putea alege şi o variantă bazată pe coeficientul lui Jaccard care ne dă gradul de similaritate între două mulţimi. Cu cât valoarea e mai mare cu atât mulţimile sunt mai similare (valoarea e în intervalul [0,1] – cu 1 indicând mulţimi identice):

$$J(B_i, B_j) = \frac{|B_i \cap B_j|}{|B_i \cup B_j|} \tag{3.14}$$

3.1.3.1.3 Pas 3: unirea clusterilor de bază similari

Combinarea clusterilor de bază.

a. Similaritatea dintre doi clusteri de bază se calculează conform formulei (3.13).
b. Dacă similaritatea dintre doi clusteri de bază depăşeşte un anumit prag atunci se vor uni.

Rezultatul clusteringului cu ajutorul algoritmului STC este un set de clusteri care pot conţine documente comune.

Exemplu:

Fie următorul set de documente care conţine fiecare câte o propoziţie:

D1: document classification is hard.

D2: document clustering is hard too.

D3: document classification and clustering is hard.

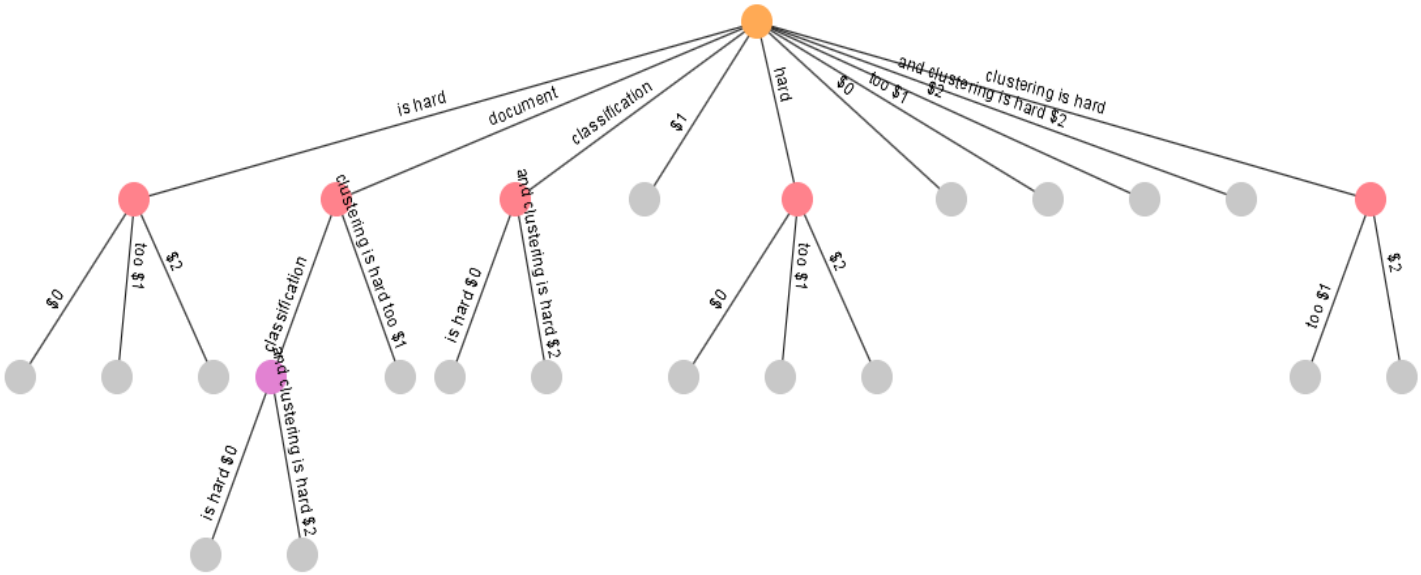


Fig. 3.7 Un exemplu de arbore de sufixe

Pas 1: Construim arborele de sufixe;

Pas 2: Alegerea nodurilor de bază şi calcularea scorului fiecărui cluster de bază;

Nodurile de bază posibile sunt acele noduri care au cel puţin 2 copii. În **Fig. 3.**8 identificăm nodurile care pot deveni clusteri de bază. Acestea devin clusteri de bază.

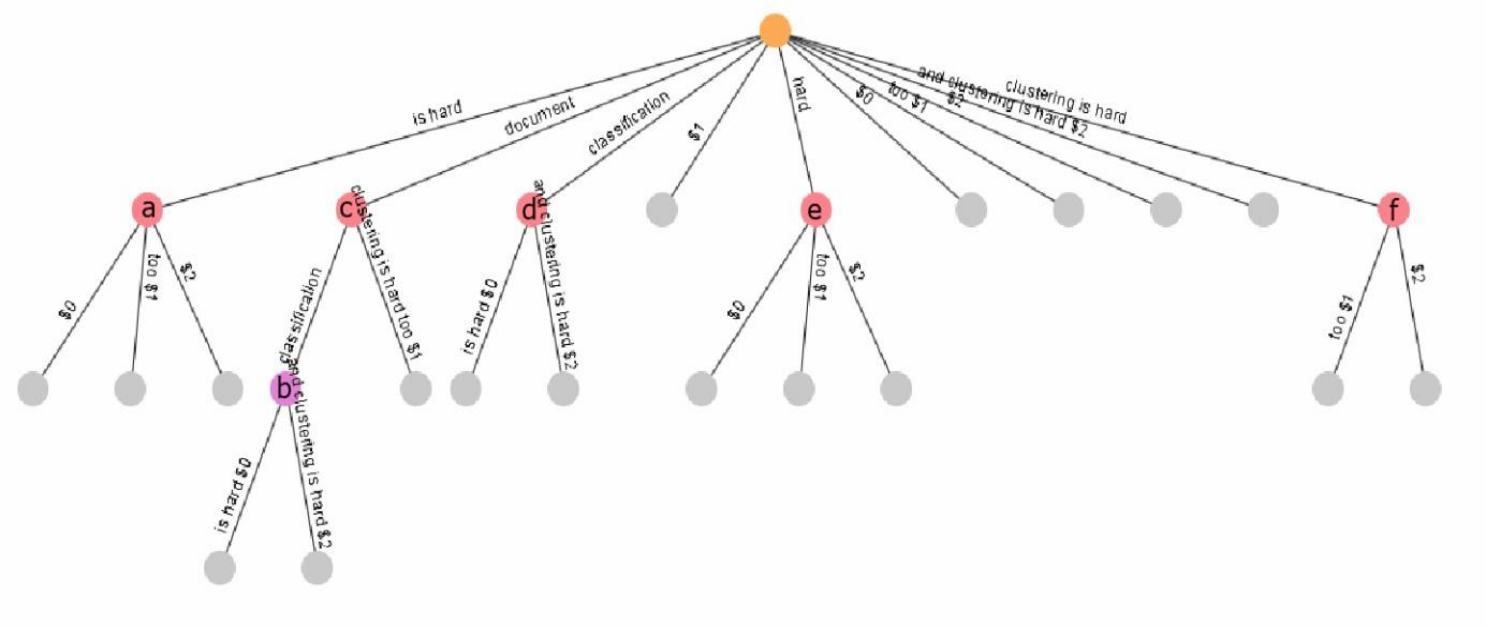


Fig. 3.8 Identificarea nodurilor de bază din arborele de sufixe

Nodul	Propoziţia din nod	Documente conţinute în nod
a	`is hard`	1, 2, 3
b	`document`	1, 2, 3
c	`document classification`	1, 3
d	`classification`	1, 3
e	`hard`	1, 2, 3
f	`clustering is hard`	2, 3

Tabel 3.2 Nodurile din arborele de sufix şi documentele conţinute în noduri

Se calculează pentru fiecare cluster de bază un scor pe baza formulei (3.11).

$S(a) = nr\_docu \cdot f(P_a) = 3 \cdot 2 = 6$

$S(b) = 3 \cdot 0{,}5 = 1{,}5$

$S(c) = 2 \cdot 2 = 4$

$S(d) = 2 \cdot 0{,}5 = 1$

$S(e) = 3 \cdot 0{,}5 = 1{,}5$

$S(f) = 2 \cdot 3 = 6$

Clusterii de bază se ordonează descrescător în funcţie de scor. În practică se păstrează doar primii zece clusteri (în literatura de specialitate se specifică un k=500). Deoarece algoritmul va crea foarte mulţi clusteri asemănători datorită suprapunerilor, în pasul următor clusterii rămaşi se vor uni dacă satisfac condiţia de similaritate din (3.13)

Ordinea clusterilor de bază în funcţie de scorul obţinut este: a, f, c, b, e, d.

Vom aplica formula (3.13) pentru clusterii *a* şi *f* alegând α=0.6:

astfel $\frac{|a \cap f|}{|a|} = \frac{2}{2} = 1$ şi $\frac{|a \cap f|}{|f|} = \frac{2}{3} = 0{,}66$, ambele valori fiind mai mari decât α, clusterii de bază *a* şi *f* se vor uni. Clusterul rezultat va conţine documentele 1, 2 şi 3. În cazul nostru, este chiar unirea tuturor exemplelor, ceea ce este şi normal având în vedere conţinutul textului.

Dacă am fi ales un prag mai mare atunci aceste noduri nu s-ar fi unit. Se procedează identic cu toate perechile de clusteri de bază.

3.1.3.1.4 Pas 4. Etichetarea clusterilor

Clusterii obţinuţi prin unire se etichetează cu etichetele conţinute în noduri sau cu altă metodă semantică.

3.1.4 Metode bazate pe densităţi

O mulţime deschisă în spaţiul (topologic) euclidian poate fi descompusă în componentele sale conexe. Plecând de la această idee, un cluster este definit ca fiind o componentă conexă densă care creşte în direcţia în care densitatea este mai mare (după gradientul densităţii). Din această cauză, algoritmii care se bazează pe densitate sunt capabili să descopere clusteri de forme arbitrare. Totodată, deplasarea în direcţia creşterii densităţii protejează algoritmul de zgomot (outlineri). Totuşi, aceste metode au dezavantajul că, dacă un cluster se compune din două zone adiacente de densitate diferită, dar mai mare ca un prag dat, algoritmii nu reuşesc să-l descopere corect. [Han01].

Dintre cei mai importanţi algoritmi ai acestei metode amintim: DBSCAN·(Density-Based Spatial Clustering Algortithm with Noise) [Anke99] care generează clusteri având un prag pentru densitatea acestora; OPTICS (Ordering Points To Identify the Clustering Structure) [Anke99] care extinde algoritmul DBSCAN la calcularea unei infinităţi de distanţe ε_i mai mici decât un ε dat $0 \leq \varepsilon_i \leq \varepsilon$. Singura diferenţă faţă de DBSCAN este faptul că algoritmul OPTICS nu asignează apartenenţa la un cluster ci memorează ordinea în care obiectele au fost procesate.

3.1.5 Metode de tip grid-based

O metodă bazată pe reţele cuantifică spaţiul de reprezentare a obiectului într-un număr finit de celule care formează o structură de tip reţea. După realizarea acestei reprezentări se aplică algoritmi de clustering. Marele avantaj este timpul mic de procesare care este independent de numărul de obiecte fiind dependent doar de numărul de celule din fiecare dimensiune a spaţiului transformat. Algoritmi tipici pentru acest tip de metodă sunt: STING (STatistical Information Grid) [Wang97], DENCLUE (DENsity CLUstering), CLIQUE (CLustering In QUEst) [Ilan10], MAFIA (Merging of Adaptive Intervals Approach to Spatial Data Mining).

3.1.6 Metode bazate pe modele

Metodele bazate pe modele pleacă de la ipoteza că există un model pentru fiecare cluster şi încearcă să găsească date care se potrivesc cel mai bine cu modelul respectiv. Un algoritm bazat pe modele poate descoperi clusteri prin construirea unei funcţii de densitate care reflectă distribuţia spaţială a datelor. Totodată, aceşti algoritmi pot conduce la o descoperire automată a numărului de clusteri când se bazează pe metode statistice standard ţinând cont de zgomot. Algoritmi care fac parte din această categorie sunt: AutoClass (Automatic Classification) [Chee96], COBWEB (COnceptual clustering) [Fish87]. Tot din această categorie fac parte şi algoritmii care utilizează ca model reţelele neuronale cum ar fi SOM (Self Organizing Maps).

De asemenea, pot fi incluşi în această categorie şi algoritmii bio-inspiraţi care se bazează pe comportamentul colectiv al furnicilor (ant colonies) respectiv al stolurilor de păsări, în căutarea hranei etc. O colonie de furnici are multe caracteristici care sunt considerate utile. O colonie poate fi privită ca un sistem format din mulţi agenţi care, deşi la nivel de individ sunt simpli, pot efectua sarcini destul de complexe la nivel de grup, dar fără o coordonare centrală. Câteva exemple sunt: construirea unui cuib de furnici, cuib pentru puiet şi cimitire de furnici

[Cama01]. În general există două tipuri algoritmi de clustering bazaţi pe furnici. Primul tip de metode imită direct comportamentele de grupare observate în coloniile de furnici reale. Al doilea tip este mai puţin inspirat din natură şi sarcina grupării este reformulată ca o sarcină de optimizare care utilizează tehnici euristice bazate pe comportamentul furnicilor pentru a găsi un clustering acceptabil sau aproape optim [Hand07].

3.2 *Algoritmi ierarhici. HAC – implementarea AGNES*

Algoritmul AGNES (AGglomerative NESting) porneşte cum am prezentat în par. (3.1.2) de la a poziţiona fiecare obiect în propriul său cluster - astfel, la început, numărul clusterilor este egal cu numărul obiectelor) - după care se unesc clusteri pe rând până la o condiţie de terminare sau până când se va obţine un singur cluster care conţine toate obiectele.

Algoritmul AGNES are o abordare de tip bottom-up în ceea ce priveşte construcţia clusterilor.

Pseudocodul pentru un astfel de algoritm este:

```
Algoritmul AGNES

Input: n obiecte
Output: k Clusteri
Pas 1.   Numerotează documentele de la 1 la n. Fiecare document este un
           cluster;
Pas 2.   Calculează distanţa d(r,s) ca fiind distanţa între obiectele r şi s
           cu r, s = 1, 2, ..., n; Fie D = (d(r,s)) matricea de similaritate
Pas 3.   Determină perechea de clusteri cei mai similari r, s, astfel încât
           d(r,s) este minimă în D;
Pas 4.   Uneşte r şi s într-un cluster nou t
           Daca d(r,s)>prag
                 Exit
                 Altfel. Calculează d(t,k) pentru toţi k ≠ r,s. Şterge
           rândurile şi coloanele corespunzătoare pentru r şi s din D şi
           adaugă o coloană şi un rând corespunzător pentru t;
Pas 5.   Repetă pas 3 de n-1 ori până când rămâne un singur cluster sau cand
           o altă condiţie de terminare e îndeplinită.
```

Structura returnată de algoritm oferă mai multe informaţii decât un set de clusteri returnaţi de un algoritm plan (algortimi care nu dezvoltă o structură ierarhică ci doar realizează o simplă grupare, care la rândul ei poate să fie suprapusă sau nu) Rezultatul clusteringului AGNES este o structură ierarhică în care, cu cât suntem pe un nivel mai înalt, cu atât avem clase mai mari şi mai generale – vizualizarea este o dendrogramă. Algoritmul de clustering AGNES nu necesită predefinirea unui număr de clusteri. Dezavantajul acestui tip de clustering îl reprezintă eficienţa mai scăzută având în vedere complexitatea cel puţin pătratică după numărul de documente faţă de complexitatea liniară a unui algoritm plan.

Exemplu

În exemplul de calcul prezentat pentru AGNES am realizat o aplicaţie simplă scrisă în cod php şi care utilizează calculul similarităţii bazat pe metoda single link.

Setul de date folosit pentru exemplu este un set de şase puncte reprezentate în plan având două dimensiuni.

Pas 1

Iniţializarea algoritmului: fiecare punct este un cluster.

Obiectul	x1	x2
d1	3	1
d2	1,5	5
d3	3,5	8
d4	4	7
d5	1	9
d6	5	1,5

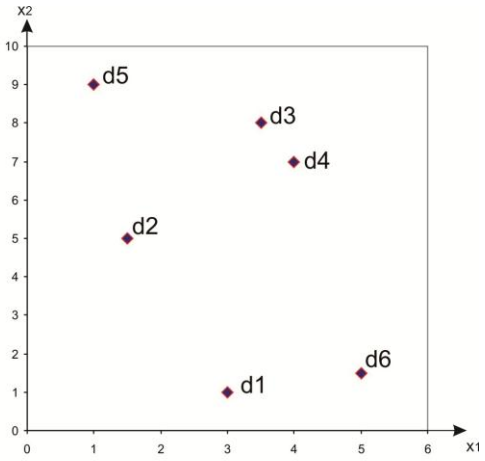


Fig. 3.9 - Iniţializarea algoritmului AGNES

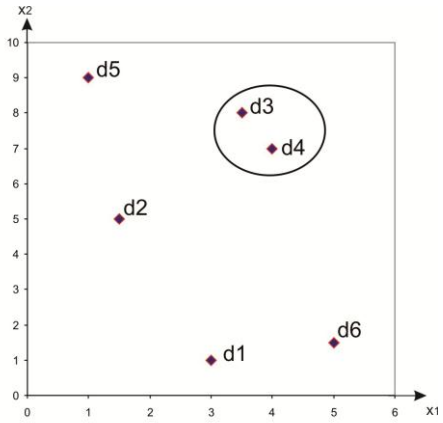
Pas 2

Calculăm matricea de distanţe dintre toate cele 6 puncte utilizând distanţa euclidiană.

Distanţa	d1	d2	d3	d4	d5	d6
d1	0,0	4,3	7,0	6,1	8,2	2,1
d2	4,3	0,0	3,6	3,2	4,0	4,9
d3	7,0	3,6	0,0	1,1	2,7	6,7
d4	6,1	3,2	1,1	0,0	3,6	5,6
d5	8,2	4,0	2,7	3,6	0,0	8,5
d6	2,1	4,9	6,7	5,6	8,5	0,0

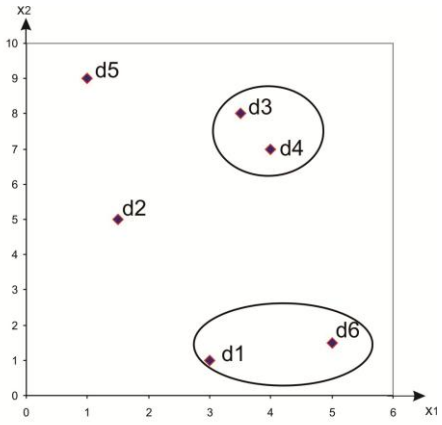
Tabel 3.3 - Matricea de distanţe

Pas 3



Alegem pentru unire doi clusteri care au distanţă minimă. În cazul nostru (d3,d4) şi recalculăm matricea de distanţe utilizând metoda single link:

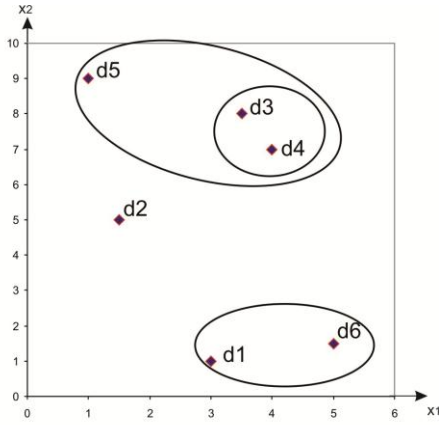
	d1	d2	d3, d4	d5	d6
d1	0,0	4,3	**6,1**	8,2	2,1
d2	4,3	0,0	**3,2**	4,0	4,9
d3, d4	**6,1**	**3,2**	0,0	2,7	5,6
d5	8,2	4,0	2,7	0,0	8,5
d6	2,1	4,9	6,7	8,5	0,0



Alegem distanţa cea mai mică. În cazul nostru, distanţa cea mai mică este între d1 şi d6 şi atunci, aceşti doi clusteri se unesc şi se recalculează matricea de distanţe. Pasul 3 se repetă până când toţi clusterii sunt uniţi într-unul singur.

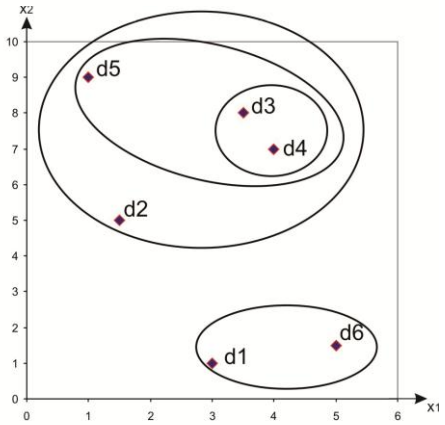
Distanţa	d1,d6	d2	d3, d4	d5
d1, d6	0,0	**4,3**	**6,1**	**8,2**
d2	**4,3**	0,0	**3,2**	4,0
d3, d4	**5,6**	**3,2**	0,0	2,7
d5	**8,2**	4,0	2,7	0,0

Deci vom grupa d5 cu (d3,d4)



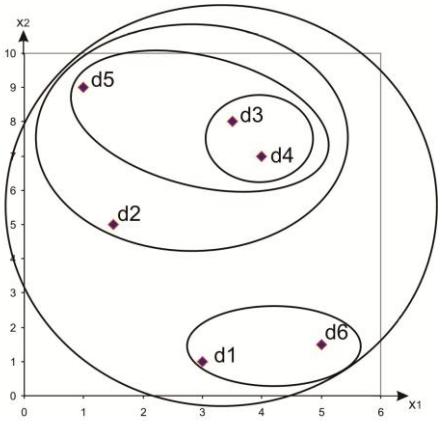
Distanţa	d1,d6	d2	d3, d4, d5
d1, d6	0,0	**4,3**	**5,6**
d2	**4,3**	0,0	**3,2**
d3, d4, d5	**5,6**	**3,2**	0,0

Deci vom grupa d2 cu (d3,d4,d5).



Distanţa	d1,d6	d3, d4, d5, d2
d1, d6	**0,0**	**4,3**
d3, d4, d5, d2	**4,3**	**0,0**

La final obţinem



Rezumând iteraţiile algoritmului clusterii au fost uniţi astfel:

d3 cu d4 la distanţa 1,1
d1 cu d6 la distanţa 2,1
(d3,d4) cu d5 la distanţa 2,7
(d3, d4, d5) cu d2 la distanţa 3,2
(d3, d4, d5, d2) cu (d1,d6) la distanţa 4,3

Fig. 3.10 Rezultatul clusteringului cu algoritmul HAC – implementarea AGNES

În funcţie de distanţa la care au fost uniţi clusterii, putem realiza ierarhizarea clusterilor şi reprezenta dendograma rezultatului clusteringului (rezultatul AGNES este o dendogramă):

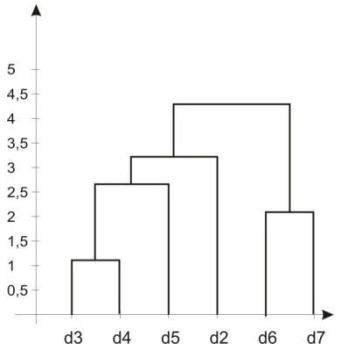


Fig. 3.11 - Dendograma pentru algoritmul HAC – single link

Cu cât urcăm în ierarhie, cu atât clusterii devin mai generali. Principala problemă care se pune este unde ar trebui să oprim algoritmul. O soluţie ar fi să oprim algoritmul când ajunge la un număr *k* de clusteri dat sau când distanţa la care se unesc doi clusteri atinge un anumit prag dat.

3.3 *Algoritmi partiţionali. K-Medoids*

Kaufman şi Rousseeuw prezintă în [Kauf87] algoritmul PAM (Partition Around Medoids) care este o implementare a algoritmului *k*-Medoids. Acest algoritm determină *k* clusteri in mulţimea de *n* obiecte prin găsirea pentru fiecare cluster a unui obiect reprezentativ (a unui medoid). Iniţial, în primul pas, medoizii sunt aleşi arbitrar apoi sunt înlocuiţi câte unul prin alegerea unui alt medoid din mulţimea obiectelor care nu sunt medoizi, atâta timp cât distanţa totală se îmbunătăţeşte (scade).

Exemplu. Am realizat în php o aplicaţie simplă care aplică algoritmul PAM unui set de zece puncte determinate de coordinatele (x,y) din plan astfel:

Obiectul	x	y
x1	2	4.5
x2	1	0
x3	2	2.5
x4	7	3
x5	10	6.5
x6	2.5	3.5
x7	5	10
x8	8	2.5
x9	4	5
x10	0	1.5

Tabel 3.4 Coordonatele a 10 puncte într-un spaţiu bidimensional

Pas1: Stabilim *k=2* şi alegem aleator două punte din mulţimea obiectelor şi le declarăm medoizi. Medoizii se aleg dacă distanţa dintre ei este mai mare decât un prag dat*:* $c_1(1,0)=x2$ *şi* $c_2(8,2.5) = x8$

Pas2: Calculăm distanţele de la fiecare punct la cei doi medoizi aleşi şi atribuim punctele celor doi medoizi alegând distanţele minime (în cazul de faţă s-a utilizat distanţa Manhattan).

Medoidul		x	z	Distanţa
1	0	2	4.5	5.5
		1	0	0
		2	2.5	3.5
		7	3	9
		10	6.5	15.5
		2.5	3.5	5
		5	10	14
		8	2.5	9.5
		4	5	8
		0	1.5	2.5

Tabel 3.5 - Distanţele tuturor punctelor calculate faţă de medoidul $c_1(1, 0)$

Medoidul		x	y	Distanţa
8	2.5	2	4.5	8
		1	0	9.5
		2	2.5	6
		7	3	1.5
		10	6.5	6
		2.5	3.5	6.5
		5	10	10.5
		8	2.5	0
		4	5	6.5
		0	1.5	9

Tabel 3.6 - Distanţele tuturor punctelor calculate faţă de medoidul $c_2(8, 2.5)$

În funcţie de distanţele calculate, punctele se atribuie medoizilor pe baza comparării distanţelor de la fiecare punct la cei doi medoizi alegându-se medoidul cu distanţă minimă. Astfel obţinem doi clusteri. Componenţa clusterilor este prezentată în Fig. 3.12.

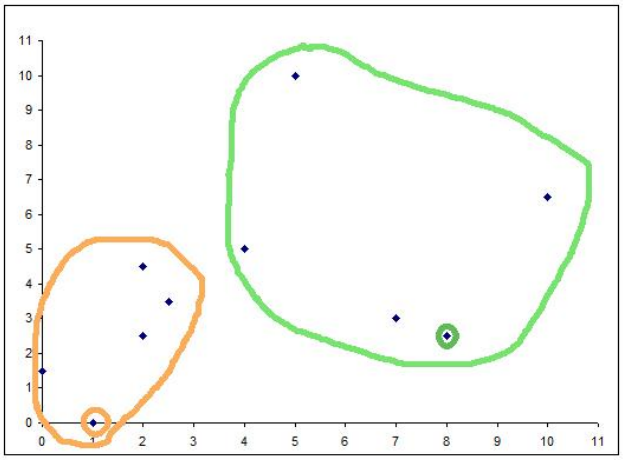


Fig. 3.12 Alegerea medoizilor şi atribuirea punctelor

Calculăm costul total ca şi sumă a distanţelor minime:

CostTotal = d((2, 4),(1, 0)) + d((2, 2.5),(1,0)) + d((2.5, 3.5),(1, 0)) + d((0, 1.5),(1, 0)) +
+ d((7, 3),(8, 2.5)) + d((10, 6.5),(8, 2,5)) + d((5, 10), (8, 2.5)) + d((4, 5), (8, 2.5)) =
=5.5+3.5+5+2.5+1.5+6+10.5+6.5=**41**

Pas3: Înlocuim un medoid cu un alt punct din mulţimea nemedoizilor şi refacem pasul 2. Alegem ca medoidul $c_1(1, 0)$ să fie înlocuit cu medoidul $c'_1(2.5, 3.5)$.

Matricea de distanţe se recalculează astfel:

Medoidul		x	y	Distanţa
2.5	3.5	2	4.5	1.5
		1	0	5
		2	2.5	1.5
		7	3	5
		10	6.5	10.5
		2.5	3.5	0
		5	10	9
		8	2.5	6.5
		4	5	3
		0	1.5	4.5

Medoidul		x	y	Distanţa
8	2.5	2	4.5	8
		1	0	9.5
		2	2.5	6
		7	3	1.5
		10	6.5	6
		2.5	3.5	6.5
		5	10	10.5
		8	2.5	0
		4	5	6.5
		0	1.5	9

Tabel 3.7. Noile valori calculate pentru distanţe de la medoizi la puncte

Pe baza distanţelor calculate se modifică compoziţia celor doi clusteri astfel:

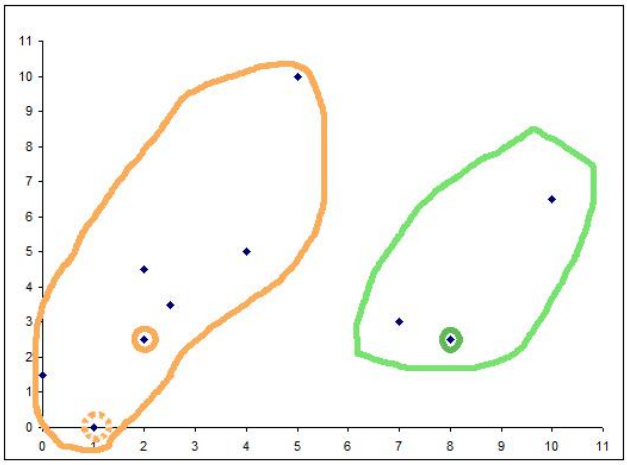


Fig.3.13 Alegerea noului medoid şi atribuirea punctelor

CostTotal=**32**

Comparând cele două costuri observăm că în urma alegerii unui alt medoid s-a dovedit că aceasta a fost inspirată.

Pasul 3 se va repeta până când ***CostTotal*** nu mai scade..

Algoritmul *k*-Medoids este mai puţin influenţat de zgomotul din date decât algoritmul *k*-Means. Deoarece *k*-Medoids lucrează direct cu medoizi nu mai este aşa de tare influenţat de outlineri cum este influenţată calcularea centrului de greutate la algoritmul *k*-Means. Totuşi, din punct de vedere al costului computaţional, algoritmul *k*-Medoids are un cost mai mare deoarece pentru o singură iteraţie, costul computaţional este $O(k(n-k)^2)$. Ca şi în cazul algoritmului *k*-Means, la algoritmul *k*-Medoids trebuie specificat la începutul rulării algoritmului numărul clusterilor *k* care se doresc a fi obţinute.

În [Serr10] este prezentată o altă modalitate de a calcula costul total:

$$Cost = \frac{cIntraCluster}{k} + \frac{cInterCluster}{k} \cdot \frac{UnaryCluster}{k} + \alpha \cdot k \cdot R \cdot \ln R \qquad (3.15)$$

unde:

k: numărul clusterilor [2 ÷ R/8].
cIntraCluster: suma distanţelor intra-cluster.
cInterCluster: suma distanţelor inter-cluster.
UnaryCluster: numărul clusterilor care conţin un singur element (chiar medoidul)
R: dimensiunea setului după eliminarea obiectelor nesemnificative
α: coeficient de dispersie (0.1 - 0.9).

În ecuaţia (3.15) valoarea *UnaryCluster* este interesantă deoarece ea penalizează semnificativ apariţia clusterilor care conţin un singur document.

If the facts don't fit the theory, change the facts.
Albert Einstein

Orice poate fi demonstrat, chiar şi adevărul
Grigore C. Moisil

4 Cercetări privind reprezentarea documentelor în algoritmi de clustering

În acest capitol voi prezenta cercetări privind îmbunătăţirea rezultatelor algoritmilor de clustering prin reprezentarea documentelor utilizând modelul Suffix Tree (STDM – Suffix Tree Document Model) comparativ cu modelul VSM (Vector Space Model). În ceea ce priveşte reprezentarea documentelor, îmi propun o abordare care se distanţează oarecum de metodele pur computaţionale şi care adaugă o cantitate mică de informaţie semantică, astfel încât, s-ar putea considera că se încearcă o apropiere de abordarea semantic-ontologică a clusteringului. Utilizarea modelului STDM este promiţătoarea deoarece această reprezentare ţine cont şi de ordinea cuvintelor din propoziţie, nu doar de frecvenţa acestora. Rezultatele obţinute vor fi comparate cu rezultatele obţinute de algoritmi de clustering utilizând modelul de reprezentare vectorial VSM.

4.1 *Modele de reprezentare utilizate*

4.1.1 Reprezentarea utilizând modelul Vector Space Model - VSM

Cea mai des întâlnită metodă de reprezentare a documentelor text în algoritmii de învăţare este cea de reprezentare vectorială (Vector Space Model - VSM). Această metodă poate fi structurată în două etape: indexarea documentelor şi ponderarea termenilor.

4.1.1.1 Indexarea documentelor

În VSM un document text este reprezentat ca vector de termeni (cuvinte) cu frecvenţe asociate [Salt75]. Definirea noţiunii de termen nu este impusă de model, dar termenii sunt de obicei cuvinte şi expresii. În cazul în care cuvintele sunt alese ca şi termeni, fiecare cuvânt din vocabularul *V* asociat documentelor devine o dimensiune independentă într-un spaţiu vectorial k-dimensional ortogonal. Orice document text poate fi apoi reprezentat de un vector în acest spaţiu dimensional mare.

Fie $d_i = (w_{i1}, w_{i2}, ..., w_{ik})$ reprezentarea unui document, unde w_{ij} reprezintă ponderea termenului *j* (din vocabularul *V)* în documentul d_i iar $k = card(V)$.

4.1.1.2 Tipuri de reprezentare a termenilor

Pentru reprezentarea ponderilor trăsăturilor am folosit în experimente modelele utilizate în [Mora07]. Astfel am utilizat 3 tipuri de reprezentări: reprezentarea binară, reprezentarea

nominală şi cea Cornell-Smart. De asemenea s-au făcut unele experimente utilizând reprezentarea Invers Document Frequency şi cea TF_IDF.

Reprezentarea binară – vectorul va conţine valoare „0" dacă cuvântul respectiv nu apare în document, şi „1" dacă cuvântul respectiv apare în document..

Reprezentare nominală – se normalizează valorile vectorului în intervalul [0,1] conform cu formula (4.1)

$$TF(d,t) = \frac{n(d,t)}{\max_\tau n(d,\tau)} \tag{4.1}$$

n(d,t) este frecvenţa apariţiei al termenului *t* în documentul *d* şi $\max_\tau n(d,\tau)$ valoarea termenului din documentul *d* cu număr maxim de apariţii

Reprezentarea Cornell SMART –valoarea ponderilor se calculează conform formulei:

$$TF(d,t) = \begin{cases} 0 & dacă\ n(d,t) = 0 \\ 1+\log(1+\log(n(d,t))) & altfel \end{cases} \tag{4.2}$$

n(d,t) este frecvenţa apariţiei al termenului *t* în documentul *d*. Domeniul de reprezentare a valorii ponderilor este în intervalul [0,2]. Dacă cuvântul nu apare în document valoarea este 0. Valoarea maximă este 2 pentru un număr de apariţii mai mic decât 10^9, având în vedere faptul că logaritmul este în baza 10.

Reprezentarea IDF (Invers Document Frequency) – în vectorul de intrare sunt ponderate valorile în funcţie de frecvenţa apariţiei termenului în colecţia de documente utilizând formula:

$$IDF(t) = \log\left(\frac{N}{N_t}\right) \tag{4.3}$$

unde *t* este termenul, *N* numărul de documente din colecţie iar N_t numărul de documente care conţin termenul *t*.

Reprezentarea TF_IDF – în vectorul de intrare sunt folosite ambele ponderi descrise în (4.1) şi (4.3) astfel:

$$TF\_IDF = TF \cdot IDF \tag{4.4}$$

Totuşi, această reprezentare vectorială combinată cu operaţiile de eliminare a cuvintelor de legătură (stop-words) şi a extragerii rădăcinilor cuvintelor (stemming) duce la eliminarea oricăror înţelesuri (sensuri) care apar datorită combinaţilor de cuvinte din propoziţie. De exemplu documentele „*Daniel este mai bun ca Radu*" şi „*Radu este mai bun ca Daniel*" sunt identice în reprezentarea VSM dar ca şi semantică sunt complet diferite. Aceste sensuri pot fi utile în algoritmii de învăţare pentru îmbunătăţirea rezultatelor acestora.

4.1.2 Reprezentarea utilizând modelul Suffix Tree Document Model - STDM

Modelul suffix tree (STDM – Suffix Tree Document Model) a apărut în literatura de specialitate legat de algoritmul de clustering Suffix Tree Clustering (STC) [Mann09], [Meye05], [Janr11] şi a fost prezentat în secţiunea 3.1.3. În modelul STDM documentele sunt reprezentate ca şi arbori de sufixe în care nodurile reprezintă cuvintele comune din documente iar frunzele sunt toate cuvintele din documente.

Fie $d=c_1c_2c_3...c_m$ reprezentarea unui document ca o secvenţă de cuvinte c_i, cu $i=1,m$ şi un set *S* de *n* documente. Arborele de sufixe pentru un set *S* care conţine *n* şiruri, fiecare de lungime m_n, este un arbore care conţine un nod rădăcină şi exact $\sum m_n$ frunze.

Reguli de construcţie a arborelui de sufixe:

1. Fiecare nod intern, altul decât rădăcina, trebuie să aibă cel puţin doi copii şi fiecare arc care pleacă dintr-un nod este etichetat cu un subşir nevid al lui *S*.
2. Oricare două arce care pornesc dintr-un nod nu pot începe cu acelaşi cuvânt.

Deoarece fiecare nod conţine cel puţin doi copii, el va conţine partea comună a cel puţin două sufixe. Fiecare frunză din arbore poate reprezenta un document sau mai multe iar toate ramurile dintre nodul rădăcină şi frunze reprezintă câte un document sau un subşir dintr-un document. Cu cât două documente au mai multe noduri comune, cu atât aceste documente tind să fie similare.

4.2 *Metodologia de lucru*

Îmi propun în continuare să analizez dacă utilizarea modelului STDM pentru reprezentarea documentelor în algoritmii de clustering poate duce la îmbunătățirea rezultatelor clustering-ului comparativ cu utilizarea reprezentării vectoriale VSM. Reprezentarea STDM în ceea ce priveşte reprezentarea documentelor de tip text include şi ordinea cuvintelor din propoziţie nu doar cuvintele din document. Astfel, în mod implicit, acest model conţine şi câteva elemente de „semantică" a documentului. Altfel spus, deşi STDM este in fond tot o reprezentare computaţională ea, prin reprezentarea arborescent-ierarhica a documentului, se apropie cumva implicit de o reprezentare (mai) ontologică, ceea ce poate aduce avantaje. Aceasta a fost **ipoteza ştiinţifică** pe baza căreia am demarat această cercetare. În modelul STDM se vor reprezenta toate propoziţiile (frazele) şi toate cuvintele din propoziţie conţinute în documente (fără cuvintele de legătură). Nodurile din arbore vor conţine cuvintele sau frazele comune din documente. Cu cât vor fi mai multe noduri comune în arbore cu atât similaritatea este mai mare între documente. Voi lua în considerare şi numărul de cuvinte din fiecare nod.

De asemenea, voi analiza dacă în etapa de preprocesare, extragerea rădăcinilor cuvintelor din documente (pentru reducerea numărului de cuvinte) influenţează sau nu rezultatul final al algoritmului de clustering.

Cercetările efectuate până în prezent s-au axat în general pe îmbunătăţirea algoritmului STC (Suffix Tree Clustering) propriu-zis. Problema cea mai importantă şi des cercetată o reprezintă problema de alegere a nodurilor din arborele de sufixe care apoi vor deveni clusteri. Algoritmul în sine poate produce foarte multe noduri şi totodată foarte multe suprapuneri între clusterii rezultaţi, astfel că unele documente pot să apară în mai mulţi clusteri diferiţi.

În lucrarea de faţă am decis utilizarea unui algoritm ierarhic (HAC – Hierarchical Agglomerative Clustering) şi a unui algoritm de tip *k*-Medoids (folosind varianta PAM - Partitioning Around Medoids a acestuia). Astfel, în reprezentarea STDM nu voi calcula arborele pentru întreaga colecţie de documente, cum se face în abordările de până acum, ci voi construi câte un arbore de sufixe pentru oricare două documente din setul de date. Avantajul metodei constă în faptul că dimensiunea arborelui rezultat este mult mai mică decât cea a arborelui pentru setul întreg de date. Totuşi, această metodă are dezavantajul că se construiesc $n(n-1)/2$ arbori mai mici, dar timpul de construcţie necesar pentru un astfel de arbore este considerabil redus, deoarece arborii construiţi au puţine ramuri şi atunci căutarea este mult mai rapidă. În plus, la un moment dat avem nevoie doar de un singur arbore deci, şi memoria necesară este mult mai mică. De asemenea, toate abordările care construiesc arborele de sufixe pentru toate documentele (vezi algoritmul STC secţiunea 3.1.3) recurg în final la metode de eliminare a unor noduri astfel încât arborele să poată fi parcurs în timp util. Eliminarea unor noduri poate duce şi la pierderea unor informaţii utile. Deoarece arborii construiţi sunt relativ mici nu voi recurge la nici o metodă de eliminare a nodurilor.

În Fig.4.1 prezint schema aplicației:

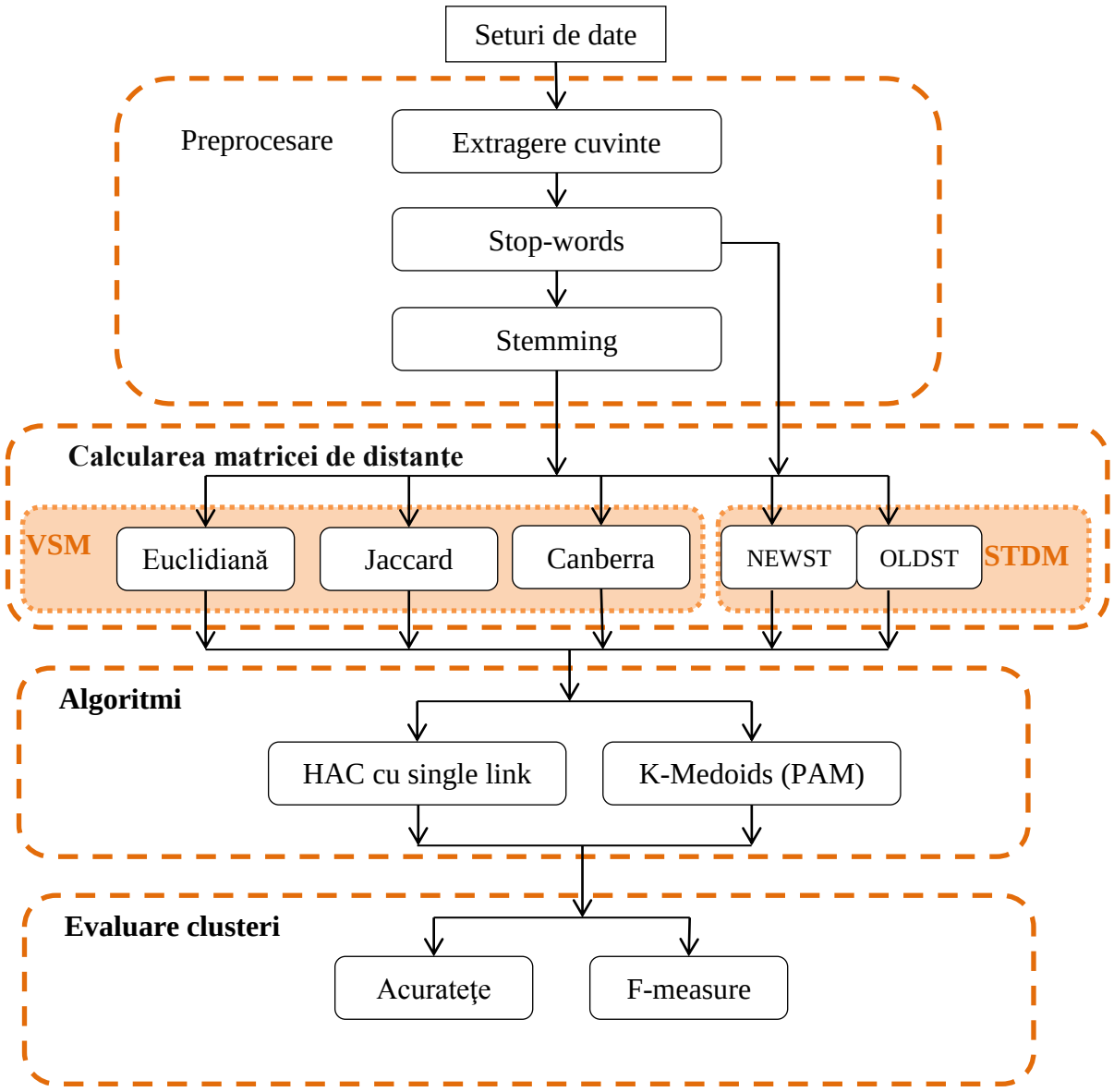


Fig. 4.1 Structura procesului de clustering

4.3 *Metrici pentru calculul matricii de similaritate şi metode de evaluare utilizate. Metrica originală propusă*

Ca şi algoritmi de clustering mi-am propus să utilizez doi algoritmi care au la bază matrice de distanţe care se calculează iniţial, iar pe parcursul rulării algoritmului aceste matrice nu se modifică substanţial. Am ales aceasta deoarece în cazul reprezentării STDM este dificil de reprezentat un document „medoid" pentru a putea reface matricea de distanţe. În alegerea algoritmilor de clustering pe care îi voi utiliza ţin cont de următoarele aspecte: aceştia trebuie să aibă la bază o matrice de distanţe între oricare două documente şi această matrice nu trebuie să se modifice pe tot parcursul rulării algoritmului.

Cei doi algoritmi aleşi sunt:

- Algoritmul ierarhic aglomerativ (HAC) – de tip single link prezentat în secţiunea 3.2.
- Algoritmul partiţional *k-Medoids* (implementarea *PAM - Partitioning Around Medoids*) prezentat în secţiunea 3.3.

Am ales aceşti algoritmi deoarece fac parte din clase de algoritmi de clustering diferite, sunt relevanţi pentru clasele din care provin şi sunt des utilizaţi în diverse aplicaţii de clustering. În [Meye05] au fost comaparaţi algoritmii HAC (average link) cu STC utilizând reprezentarea VSM şi reprezentarea STDM. Totuşi aceste abordări sunt diferite faţă de cercetarea propusă deoarece compară doi algoritmi şi nu două reprezentări de documente. În ambii algoritmi, matricea de distanţe necesită calcularea distanţei dintre oricare două documente. Pentru calculul distanțelor dintre oricare două documente distincte am ales două tipuri de metrici:

A) metrici care descriu similaritatea a două documente:

i.) Metrica notată OLDST – această distanţă a fost preluată din [Meye05] şi modificată (prin simpla adăugare a termenului „1-”) la forma:

$$d_{OLDST}(x_i, x_j) = \left(1 - \frac{\#noduri\_comune}{\#total\_noduri}\right) \tag{4.5}$$

ii.) Metrica notată NEWST – această metrică este dezvoltată de mine plecând de la observaţiile că nu doar numărul nodurilor comune este important pentru a stabili dacă două documente sunt similare ci şi informaţia conţinută în aceste noduri. Astfel, documente conţinute în noduri care au şi propoziţii (şiruri de cuvinte) comune nu numai cuvinte comune vor tinde să fie evaluate ca fiind mai similare. În formulă am introdus termenul „1+” pentru a evita împărţirea la 0.

$$d_{NEWST}(x_i, x_j) = \left(1 - \frac{1+\#noduri\_comune}{1+\#total\_noduri}\right) * \frac{1}{1+\#\text{cuvinte}\_din\_nod\_comune} \tag{4.6}$$

iii.) Metrica Jaccard – este derivată din coeficientul Jaccard utilizat la compararea mulţimilor:

$$d_{JAC}(x_i, x_j) = \left(1 - \frac{\#atribute\_comune}{\#total\_atribute}\right) \tag{4.7}$$

unde

#noduri_comune	reprezintă numărul nodurilor care sunt parcurse de ambele documente;
#total_noduri	reprezintă numărul total de noduri din arborele de sufixe.
#cuvinte_din_nod_comune	reprezintă numărul de cuvinte (comune) – derivate inclusiv din şiruri de cuvinte din diferitele noduri- parcurse pentru a ajunge la acel nod
#atribute_comune	reprezintă numărul de atribute comune celor două documente pe reprezentarea VSM
#total_atribute	reprezintă numărul total de atribute folosit pentru reprezentarea documentelor

Pentru metrica NEWST în momentul în care nu avem nici un nod comun rezultatul tinde la 1, atingând 1 când numărul total de noduri tinde la infinit. Iar rezultatul este 0 în momentul în care toate nodurile din arbore sunt comune ambelor documente.

Pentru metrica OLDST şi distanţa Jaccard codomeniul este [0,1].

Toate metricile prezentate în secţiunea A reprezintă similarităţi şi pentru a le putea folosi în matricea de distanţe am scăzut rezultatul obţinut din valoarea 1.

B) metrici care descriu disimilaritatea a două documente:

i.) Distanţa euclidiană (descrisă şi în ecuaţia 2.11):

$$d_E(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2} \tag{4.8}$$

ii.) Distanţa Canberra (descrisă şi în ecuaţia 2.15):

$$d_{CAN}(x_i, x_j) = \sum_{k=1}^{n} \frac{|x_{ik} - x_{jk}|}{|x_{ik}| + |x_{jk}|} \tag{4.9}$$

unde x_i este un document din setul de date reprezentat prin p atribute (cuvinte).

Cele două distanţe întorc rezultat în intervalul $[0,\infty]$ iar pentru a putea fi comparate cu distanţele din secţiunea A după ce s-a calculat toată matricea de distanţe, toate distanţele au fost aduse în intervalul [0,1] prin împărţirea acestora la valoarea cea mai mare din matrice.

Pentru calcularea distanţelor voi folosi ambele tipuri de reprezentări ale documentelor – atât reprezentarea Suffix Tree (STDM) cât şi cea vectorială (VSM) urmând, ca în funcţie de reprezentare să folosim anumite metode de calcul al distanţei.

În reprezentarea documentelor cu modelul STDM voi construi câte un arbore de sufixe pentru oricare două documente distincte din setul de documente și voi calcula distanța dintre ele utilizând prima dată distanţa NEWST (4.6) și apoi distanţa OLDST (4.5).

Formula de calcul NEWST propusă de mine are următoarele avantaje:

1. Dacă sunt două documente care nu au nici un nod comun, atunci „distanţa” NEWST dintre ele depinde de numărul de cuvinte din cele două documente. Astfel, cu cât documentele sunt mai mari şi nu au nici măcar un nod comun, „distanţa” dintre ele va fi mai aproape de 1 dar totuşi diferită, comparativ cu celelalte metrici care întorc întotdeauna valoarea 1 dacă documentele nu au nici un nod comun. Această mică diferenţiere ne ajută să stabilim ordinea în care vor fi unite documentele pe baza matricei de distanţe. Astfel documentele mari vor fi unite ultimele.
2. În cazul în care documentele au noduri comune am ponderat valoarea întoarsă şi cu numărul de cuvinte (şiruri) din nodul comun. Astfel se fac diferenţe între documentele care au părţi comune mai mari şi cele care au doar cuvinte comune.

În cazul reprezentării VSM, distanța dintre documente se va calcula cu ajutorul distanței euclidiene, a distanței Jaccard sau a distanței Canberra.

Pentru antrenarea şi evaluarea algoritmilor de clustering am folosit seturile de ştiri prezentate în secţiunea 2.7.2. În continuare, fiecare ştire va fi considerată un document. În etapa de preprocesare a seturilor de date prezentate în secțiunea 2.7.2 s-au generat 2 seturi distincte astfel:

a. unul în care s-au eliminat doar cuvintele de legătură;
b. unul în care s-au eliminat cuvintele de legătură și s-a extras rădăcina cuvântului (pentru extragerea rădăcini cuvântului s-a folosit algoritmul Porter implementat în pachetul IR [Rijs80])

Algoritmii de clustering vor primi ca parametri de intrare, pe rând seturile de date precum și numărul de clusteri care trebuiesc obţinuţi (egal cu numărul de categorii dintr-un anumit set).

Pentru evaluarea clusterilor obţinuţi am folosit ca și clase de bază categoria din care provine fiecare document. Astfel, putem utiliza pentru evaluarea algoritmilor de clustering metrici cum ar fi *acuratețea* și scorul *F-measure*. Cele două măsuri sunt utilizate în general în evaluarea algoritmilor de clustering și de clasificare în cazul folosirii unor seturi de date pre-etichetate.

Deoarece în etapa de selecţie a ştirilor am ales doar ştiri pre-etichetate, iar documentele din seturile de date au fost salvate împreună cu denumirea categoriei, avem seturi de date pre-etichetate. Considerăm că etichetarea făcută de agenţii de la site-urile de ştiri este „perfectă” şi o să ne raportăm la ea. În continuare, vom nota „etichetă” ca fiind categoria căreia îi aparţine documentul specificat de agenţia de ştiri, iar „clasă” va fi categoria în care a fost inclus documentul de către algoritmul de clustering.

Numele unui cluster este dat de eticheta care apare cel mai frecvent în clusterul respectiv. În cazul în care există un cluster deja etichetat cu acea valoare se va folosi eticheta imediat următoare (ca şi frecvenţă de apariţie) validă sau se specifică „*No class*” dacă nu mai sunt etichete.

De exemplu, pentru 3 clusteri am obținut clasele:

```
 0    1    2  <-- assigned to cluster
47    0    0 | fotbaleuropean
 0   62    0 | lybia
 0    0   42 | motorsport
```

unde 0, 1, 2 reprezintă clusterii, iar valoarea din stânga reprezintă „clasa” stabilită pe baza celei mai frecvente etichete din acel cluster.

Pentru evaluarea clusterilor am utilizat acurateţea şi scorul F-measure ambele formule fiind prezentate în secţiunile 2.2.1.1 şi 2.6.1. Măsura generală F-measure pentru evaluarea soluţiei unui clustering este ponderată prin dimensiunea relativa a fiecărui cluster.

$$F(S) = \sum_i \frac{n_i}{n} \max(F(C_i, S_j)) \qquad (4.10)$$

unde n_i reprezintă numărul de documente din clusterul *i* şi *n* reprezintă numărul total de documente din set. Valoarea F(S) este în intervalul [0,1]; cu cât valoarea se aproprie de 1 cu atât rezultatul este mai bun.

În timp ce acurateţea pentru categoria curentă ţine cont doar de numărul de documente grupate corect în acea categorie, F-measure ţine cont pe lângă aceasta şi de numărul de documente grupate în altă categorie decât categoria curentă şi care într-adevăr nu trebuiau grupate în categoria curentă. Astfel F-measure este o măsură mai fină a calităţii grupării datelor.

4.4 *Rezultate obţinute pe seturile RSS*

Toate testele ale căror rezultate sunt prezentate mai jos au fost rulate pe un sistem Intel(R) Core2 Duo cu procesoare T6600 la 2,20GHz, 4GB DRAM şi sistem de operare Windows7 Home Premium licenţiat.

Algoritmii de clustering au fost implementați în limbajul Java folosind platforma Eclipse. Pentru faza de preprocesare a documentelor s-au utilizat şi anumite clase din pachetul WEKA 3.7 [WEKA]. Documentele procesate au fost iniţial transferate într-un format utilizat şi de alte aplicaţii consacrate pentru algoritmi de învăţare cum ar fi formatul „arff”.

În continuare, voi prezenta detaliat rezultatele obţinute pe seturile de date RSS de către cei doi algoritmi de clustering implementaţi HAC (Hierarchical Agglomerative Clustering) şi *k*-Medoids (cu implementarea PAM – Partitioning Around Medoids). Rezultatele for fi prezentate separat pe algoritmi cât şi pe modul de reprezentare al datelor VSM (Vector Space Model) şi STDM (Suffix Tree Document Model). În final, prezint şi câteva rezultate comparative între cele două reprezentări şi cei doi algoritmi de clustering utilizaţi. De asemenea, pentru fiecare model de reprezentare şi pentru fiecare algoritm de clustering voi prezenta rezultatele obţinute pe datele

pe care nu s-a aplicat algoritmul de stemming dar şi pe datele pe care s-a aplicat algoritmul de stemming. Pentru evaluarea rezultatelor voi folosi acurateţea şi scorul F-measure, metode de evaluare externă prezentate în secţiunea 2.6.1. Utilizez ambele măsuri pentru a mă asigura de validitatea rezultatelor. Acurateţea şi scorul F-measure trebuie să reflecte în principiu aceleaşi tendinţe. Rezultatele pentru acurateţe vor fi prezentate sub formă de procente iar rezultatele pentru scorul F-measure vor fi numere în intervalul [0, 1] valoarea 1 însemnând valoarea cea mai bună.

4.4.1 Rezultatele obţinute de algoritmul HAC cu reprezentare VSM

În tabelul 4.1 şi în Figura 4.3 prezint, pentru modelul VSM şi pentru fiecare din cele trei metode de calcul a distanţei prezentate în secţiunea 4.3, rezultatele pentru seturile de date S01-S07 la care am eliminat cuvintele de legătură dar nu am extras și rădăcinile cuvintelor.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
Jaccard	**71.52%**	**53.80%**	**66.67%**	**79.10%**	31.09%	**65.52%**	**79.33%**	**63.86%**
Euclidian	41.06%	35.09%	39.74%	34.46%	**32.64%**	36.21%	34.64%	36.26%
Canberra	42.38%	34.50%	39.74%	33.33%	32.12%	25.86%	32.96%	34.42%

Tabel 4.1 Rezultatele acurateţei obţinute de algoritmul HAC – fără stemming cu reprezentarea VSM

În tabel am scos în evidenţă cele mai bune rezultate obţinute pe fiecare set în parte în funcţie de metrica utilizată.

Se observă faptul că algoritmul HAC care utilizează distanţa Jaccard obţine cea mai bună acurateţe pe 6 seturi de date. Rezultatul mai puţin bun pe setul S05 s-a obţinut din cauza faptului că există clusteri slab reprezentaţi comparativi cu alţii care sunt mai bine reprezentaţi. Valoarea medie a acurateţei obţinută de algoritm cu distanţa Jaccard este de 63,86% iar valoarea maximă atinsă este de 79,33%.

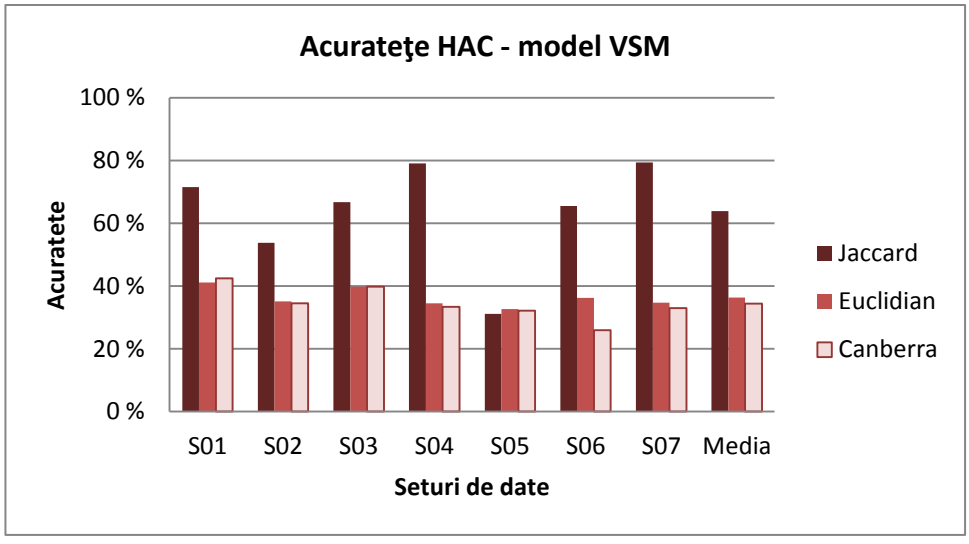


Fig. 4.3 HAC - Acurateţea clustering-ului pentru toate cele 7 seturi fără stemming

În tabelul 4.2 şi în Figura 4.4 prezint scorul *F-measure* pentru seturile de date S01-S07 şi modelul VSM. Se observă comparativ cu graficul prezentat în Fig. 4.3 că în cazul setului de date S06 acurateţea obţinută de măsura Jaccard este semnificativ mai mare decât scorul F-measure. Scorul F-measure fiind un indicator care ţine cont şi de documentele care într-adevăr nu aparţin unei categorii şi nu au fost atribuite acelei categorii a penalizat rezultatul măsurii Jaccard care a ales o altă etichetă decât cea reală pentru un cluster mai mare.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
Jaccard	**0.8078**	**0.5645**	**0.7731**	**0.7600**	0.3739	**0.3744**	**0.7627**	**0.6309**
Euclidian	0.4738	0.3393	0.4598	0.3528	0.3812	0.2714	0.3513	0.3757
Canberra	0.4728	0.3491	0.5463	0.3427	**0.3816**	0.2669	0.3301	0.3842

Tabel 4.2 Rezultatele F-measure obţinute de algoritmul HAC – fără stemming cu reprezentarea VSM

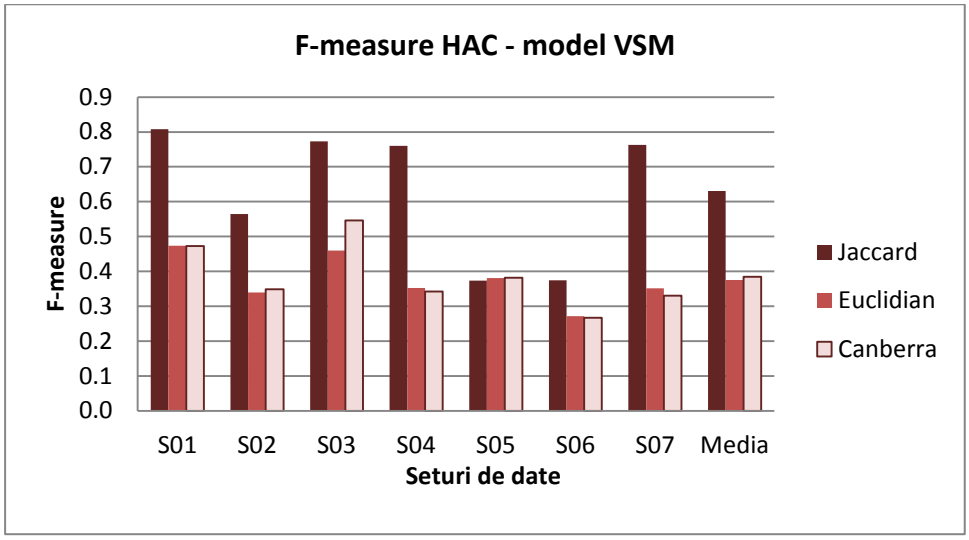


Fig. 4.4 HAC - F-measure pentru toate cele 7 seturi de date fără stemming

Măsura *F-measure* fiind un scor pentru media ponderată dintre precizie şi recall îl voi prezenta peste tot ca valoare obţinută (nu procentual).

În Tabelul 4.3 şi în Figura 4.5 prezint rezultatele pentru algoritmul HAC pe seturile de date S01-S07 la care am eliminat cuvintele de legătură și am extras și rădăcinile cuvintelor prin aplicare algoritmului de stemming Porter.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
Jaccard	41.72%	33.92%	39.10%	**34.46%**	**33.68%**	**68.97%**	**34.64%**	**40.93%**
Euclidian	**43.05%**	**35.09%**	**39.74%**	**34.46%**	32.64%	39.66%	**34.64%**	37.04%
Canberra	42.38%	33.92%	39.10%	33.33%	32.12%	25.86%	32.96%	34.24%

Tabel 4.3 Rezultatele acurateţei obţinute de algoritmul HAC – cu stemming cu reprezentarea VSM

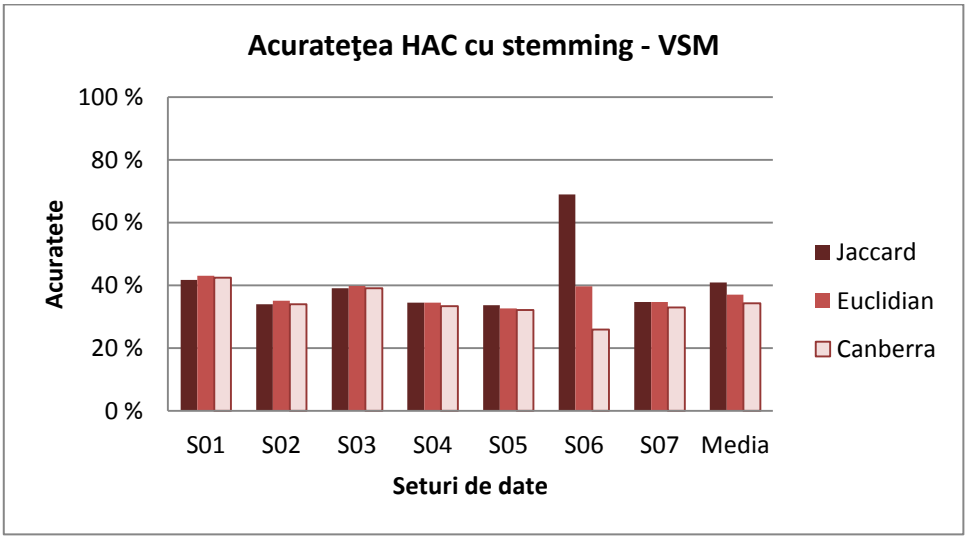


Fig. 4.5 HAC - Acuratețea clustering-ului pentru toate cele 7 seturi cu stemming

Rezultatele scorului F-measure prezentate în Tabelul 4.4 şi în Figura 4.6 respectă tendinţele acurateţei obţinute de algoritmul HAC în reprezentarea VSM cu aplicarea algoritmului Porter de extragere a rădăcinilor cuvintelor.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
Jaccard	**0.4735**	0.3532	0.4515	0.3416	0.3808	**0.6827**	0.3379	**0.4316**
Euclidian	0.4720	0.3622	0.4598	**0.3528**	0.3812	0.2058	**0.3513**	0.3693
Canberra	0.4728	**0.3623**	**0.4605**	0.3427	**0.3816**	0.2669	0.3301	0.3739

Tabel 4.4 Rezultatele F-measure obţinute de algoritmul HAC – fără stemming cu reprezentarea VSM

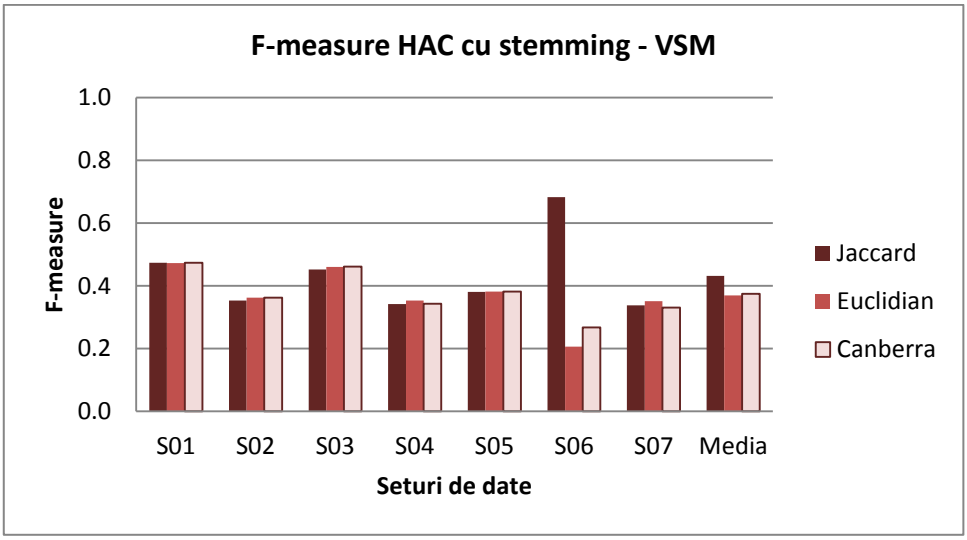


Fig. 4.6 HAC - F-measure pentru toate cele 7 seturi de date cu stemming

Analizând rezultatele obţinute de algoritmul HAC utilizând modelul VSM de reprezentare se observă din valorile medii obţinute de cele trei metrici utilizate că aplicarea algoritmului de stemming a influenţat negativ metrica Jaccard, scăzând acurateţea cu 22.93%; în schimb, pentru metrica euclidiană acurateţea s-a îmbunătăţit cu 0.78%. Scăderea acurateţei la metrica Jaccard se poate explica prin faptul că, fiind o metrică de similaritate care se bazează pe numărul cuvintelor comune distincte, în momentul în care se aplică algoritmul de stemming se reduce semnificativ acest număr iar diferenţierea între documente devine dificilă. Rezultatul fracţiei devine identic pentru mai multe perechi de documente care poate nu sunt cu adevărat similare. Pentru distanţa euclidiană care se bazează pe calculul vectorial, reducerea numărului de atribute este benefică pentru că reduce calculul.

4.4.2 Rezultatele obţinute de algoritmul HAC cu reprezentare STDM

În Tabelul 4.5 şi în Figura 4.7 prezint pentru modelul STDM, pentru fiecare din cele două metode de calcul a distanţei prezentate în par. 4.3, rezultatele pentru seturile de date S01-S07 la care am eliminat cuvintele de legătură dar nu am extras și rădăcinile cuvintelor.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
NEWST	**100.00%**	99.42%	**95.51%**	77.40%	**76.17%**	**84.48%**	77.65%	**87.23%**
OLDST	**100.00%**	**100.00%**	65.38%	**96.05%**	72.02%	70.69%	**96.09%**	85.75%

Tabel 4.5 Rezultatele pentru acurateţe obţinute de algoritmul HAC – fără stemming cu reprezentarea STDM

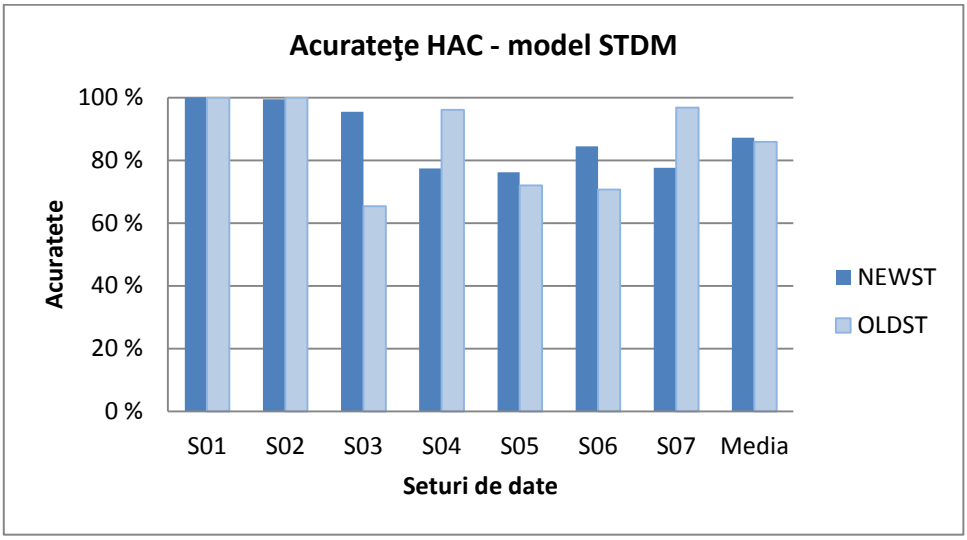


Fig. 4.7 HAC - Acurateţea clustering-ului pentru toate cele 7 seturi de date la care s-au eliminat doar cuvintele de legătură

Dacă analizăm rezultatele prezentate în Tabelul 4.5 (rezultate obţinute de modelul STDM fără stemming) şi le comparăm cu rezultatele prezentate în Tabelul 4.1 (rezultate obţinute de modelul VSM fără stemming) se observă că rezultatul clusteringului s-a îmbunătăţit, fapt încurajator pentru metrica NEWST propusă de mine. Precizăm faptul că iniţial am rafinat procesarea algoritmului HAC pe anumite seturi de date favorabile. Procesul de rafinare a fost şi reciproc. Astfel se explică faptul că rezultatele obţinute pe aceste seturi de date, oarecum artificioase (S01, S02), sunt hiperbolizate. Scopul acestui demers a fost legat de necesitatea garanţiei corectitudinii implementării mele. Mai precis, procentajele de 100% au fost posibile pe setul S01 deoarece categoriile alese sunt suficient de diferite iar documentele conţinute au fost selectate astfel încât să conţină cuvinte identice şi astfel implicit distanţa la care algoritmul le uneşte este mică făcând posibilă o diferenţire clară între cele trei categorii. Pentru a demonstra utilitatea metricii NEWST voi prezenta în Fig. 4.8 rezultatele comparative privind acurateţea obţinută de metrica NEWST propusă de mine şi măsura OLDST pentru algoritmul HAC fără stemming. În grafic am reprezentat diferenţa calculată dintre valoarea pentru acurateţe obţinută de metrica NEWST şi valoarea pentru acurateţe obţinută de metrica OLDST. Pe unele seturi, metrica NEWST a funcţionat mai bine decât OLDST şi atunci diferenţa dintre valorile obţinute este pozitivă. În cazul în care metrica OLDST a obţinut rezultate mai bune faţă de metrica NEWST, diferenţa este negativă.

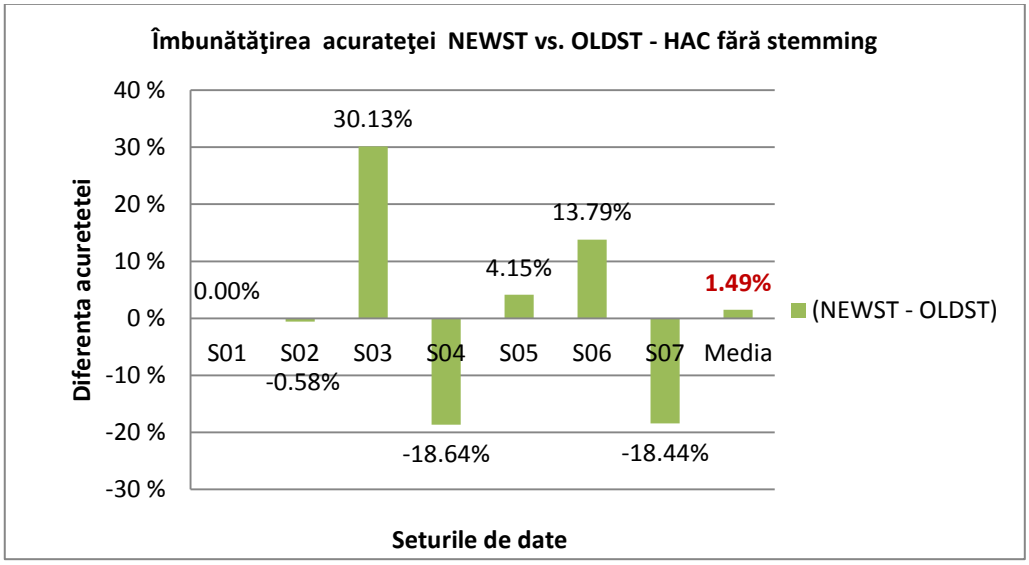


Fig. 4.8 HAC – Îmbunătăţirea acurateţei obţinută de NEWST versus OLDST în reprezentarea seturile de date fără stemming

Se observă din Fig. 4.8 că metrica NEWST propusă a îmbunătăţit în medie cu 1.49% acurateţea algoritmului de clustering aplicat în cazul utilizării modelului de reprezentare a datelor STDM pe seturile de date, fără aplicarea algoritmului de extragere a rădăcinilor cuvintelor.

În Tabelul 4.6 şi Figura 4.9 prezint scorul *F-measure* pentru seturile de date S01-S07 şi modelul STDM fără aplicarea algoritmului de extragere a rădăcinilor cuvintelor.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
NEWST	**1.0000**	0.9942	**0.9718**	0.8357	0.5222	**0.8576**	**0.8375**	**0.8599**
OLDST	**1.0000**	**1.0000**	0.6756	**0.9738**	**0.5461**	0.6927	0.8129	0.8144

Tabel 4.6 Rezultatele pentru F-measure obţinute de algoritmul HAC – fără stemming cu reprezentarea STDM

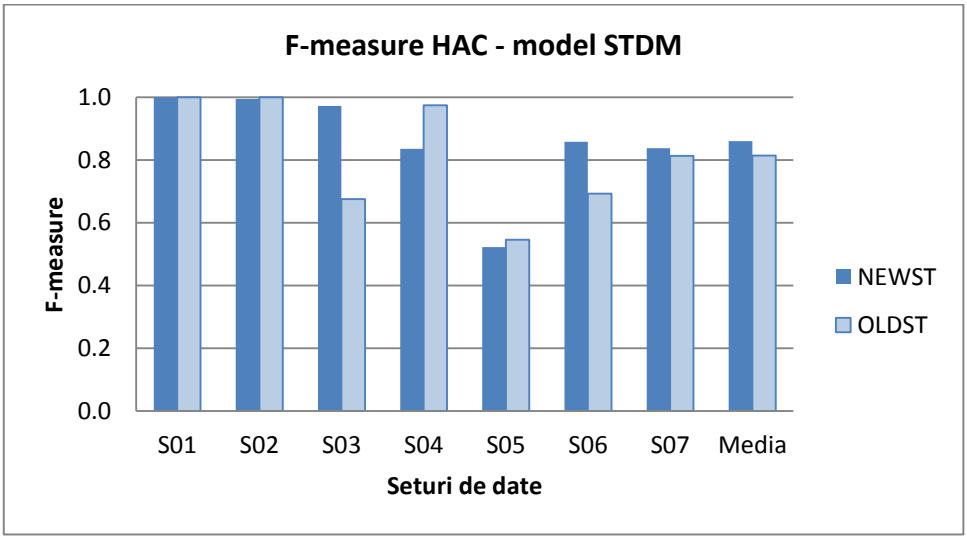


Fig. 4.9 HAC - F-measure pentru toate cele 7 seturi de date fără stemming

Valorile obţinute de scorul F-measure respectă tendinţele valorilor obţinute de acurateţe în cazul utilizării reprezentării STDM. În Fig. 4.10 prezint îmbunătăţirea adusă de metrica NEWST comparativ cu metrica OLDST în cazul scorului F-measure. Am reprezentat diferenţele între valoarea obţinută de metrica NEWST şi metrica OLDST. Diferenţa este pozitivă dacă metrica NEWST a obţinut rezultatul mai bun şi este negativă în cazul în care metrica OLDST a obţinut rezultatul mai bun.

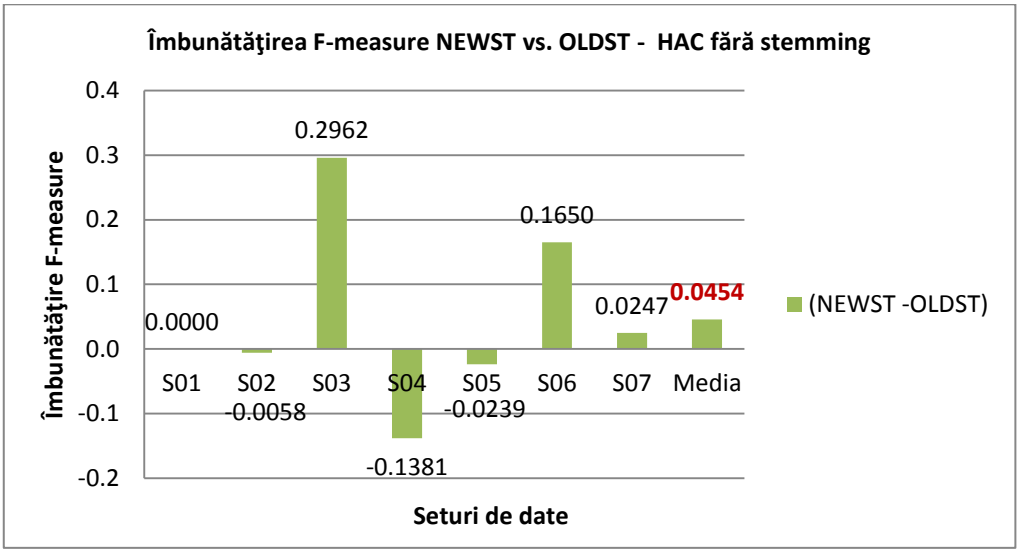


Fig. 4.10 HAC – F-measure NEWST versus OLDST în reprezentarea seturile de date fără stemming

În tabelul 4.7 şi figura 4.11 voi prezenta rezultatele obţinute de algoritmul HAC utilizând reprezentarea STDM, dar aplicând pe seturile de date S1-S7 algoritmul de stemming al lui Porter.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
NEWST	**100.00%**	99.42%	**95.51%**	77.40%	**76.17%**	**84.48%**	77.65%	**87.23%**
OLDST	**100.00%**	**100.00%**	65.38%	**96.05%**	72.02%	70.69%	**96.09%**	85.75%

Tabel 4.7 Rezultatele pentru acurateţe obţinute de algoritmul HAC – cu stemming cu reprezentarea STDM

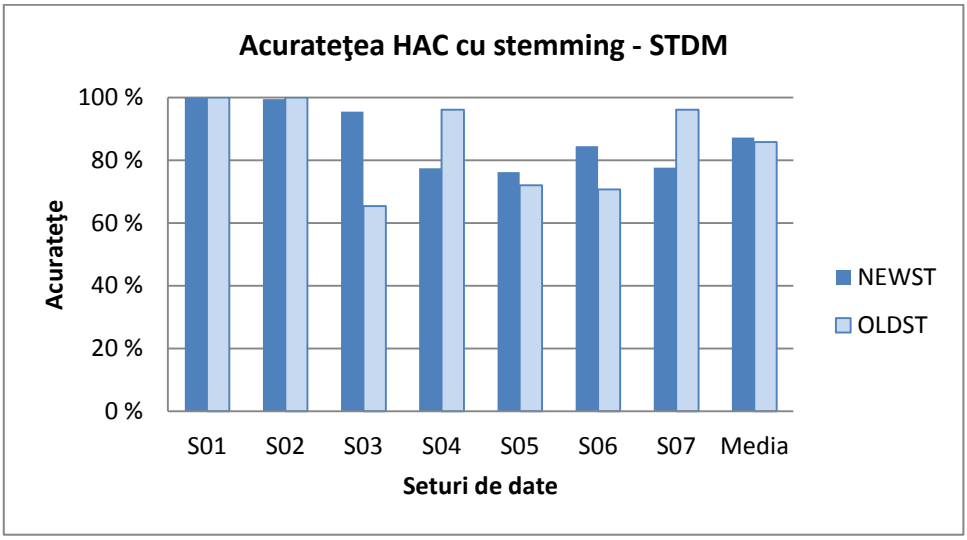


Fig. 4.11 HAC - Acurateţea clustering-ului pentru seturile de date în reprezentarea STDM cu stemming

Pentru a compara din nou cele două metrici în cazul utilizării algoritmului de stemming pe seturile de date, am calculat diferenţele dintre valorile obţinute pentru acurateţe de cele două măsuri NEWST şi OLDST. Din nou, o valoare pozitivă reprezintă diferenţa în favoarea metricii NEWST iar o valoare negativă reprezintă diferenţa în favoarea metricii OLDST. Diferenţele dintre valorile obţinute pentru acurateţe le prezint în Fig. 4.12.

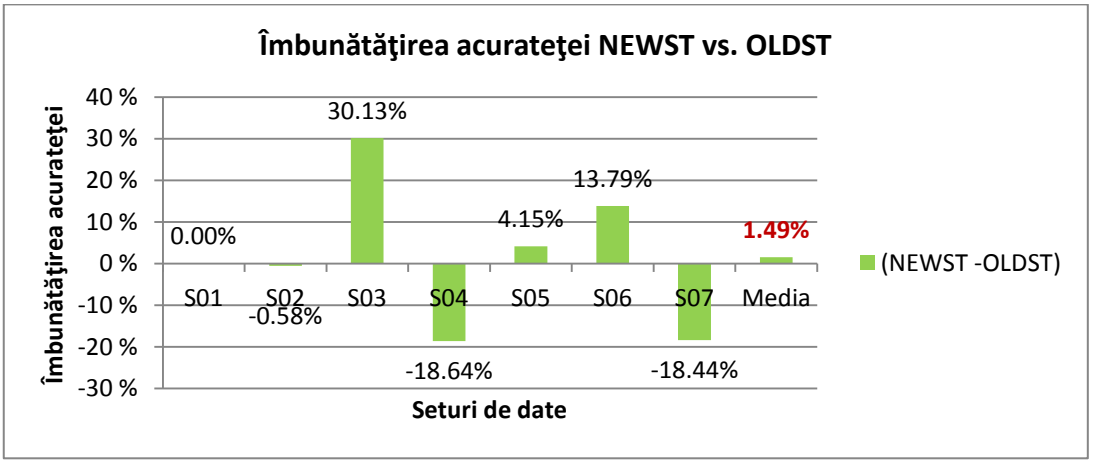


Fig. 4.12 HAC - Acurateţea NEWST vs. OLDST în reprezentarea STDM pe seturile de date cu stemming

Şi în cazul utilizării algoritmului de stemming, metrica NEWST a obţinut rezultate în medie cu 1.49% mai bune ca metrica OLDST. Această valoare medie este identică cu valoarea pentru îmbunătăţirea medie obţinută de acelaşi algoritm cu reprezentarea STDM dar fără extragerea rădăcinilor din seturile de date. Analizând mediile de îmbunătăţire între metricile NEWST şi OLDST prezentate în Fig.4.8 şi Fig. 4.12 putem afirma că aplicarea algoritmului de stemming nu a adus îmbunătăţiri în cazul reprezentării STDM.

Valorile obţinute de scorul F-measure pentru seturile de date S1-S07 pentru algoritmul HAC utilizând reprezentarea STDM cu stemming le prezint în tabelul 4.8 şi figura 4.12.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
NEWST	**1.0000**	0.8100	**0.8084**	0.8357	**0.8407**	**0.8576**	**0.8375**	**0.8557**
OLDST	**1.0000**	**1.0000**	0.6756	**0.8581**	0.7730	0.6927	0.8129	0.8303

Tabel 4.8 Rezultatele pentru F-measure obţinute de algoritmul HAC – cu stemming cu reprezentarea STDM

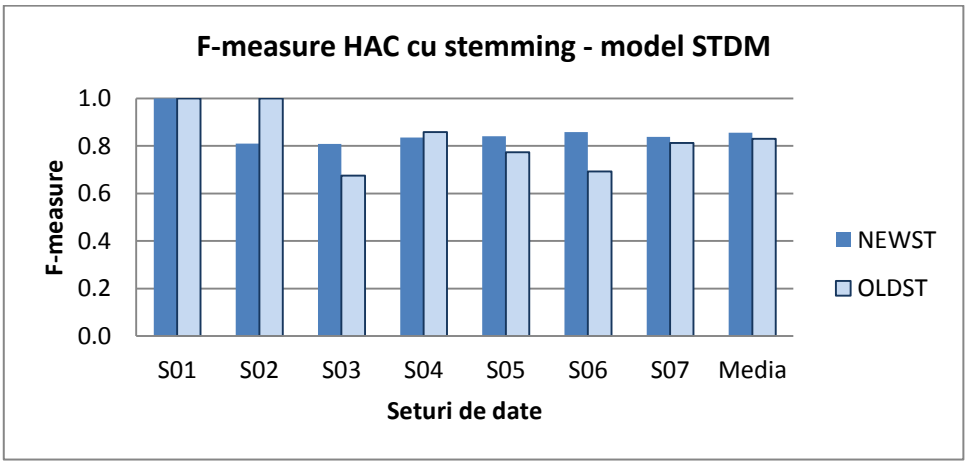


Fig. 4.12 HAC – F-measure clustering-ului pentru seturile de date în reprezentarea STDM cu stemming

Din Fig. 4.12 se poate observa faptul că pe setul S02 măsura NEWST care a obţinut o acurateţe de 99,42% totuşi obţine un scor F-measure mai mic faţă de OLDST, aceasta explicându-se prin faptul că probabil a inversat două clase una având mai multe documente decât cealaltă şi i s-a atribuit numele unei clase clusterului mai mic iar clusterului mai mare a trebuit să i se atribuie alt nume.

În Fig. 4.13 voi prezenta îmbunătăţirea adusă scorului F-measure de metrica NEWST faţă de OLDST în cazul algoritmului HAC pentru reprezentarea STDM aplicând algoritmul de stemming.

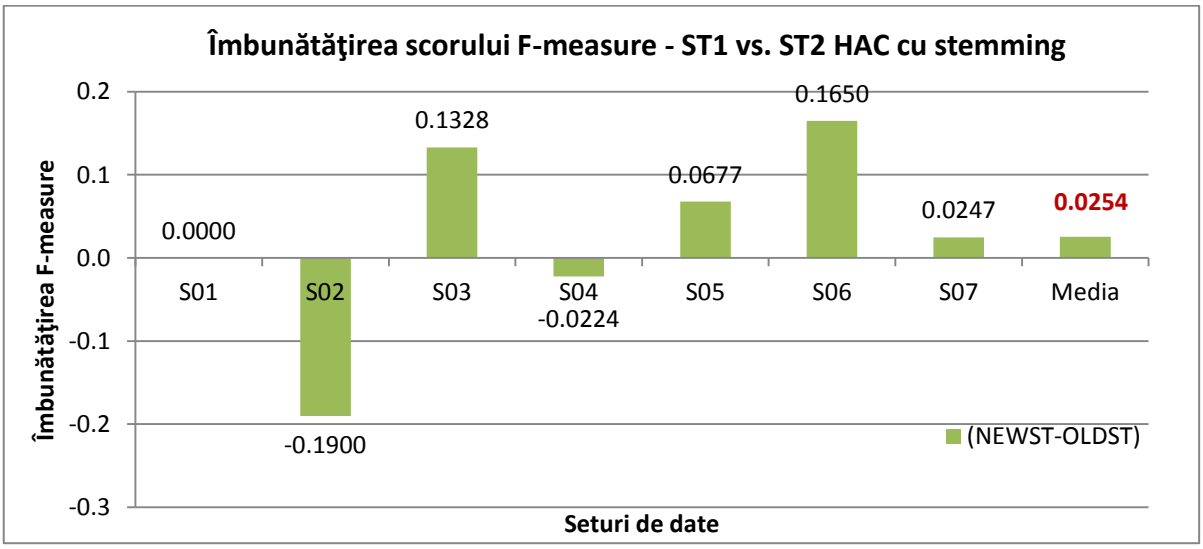


Fig. 4.13 HAC – F-measure NEWST vs. OLDST în reprezentarea STDM pe seturile de date cu stemming

Îmbunătăţirea scorului F-measure este mai mică cu 0.02 faţă de cazul când nu am extras rădăcinile cuvintelor. Reducerea atributelor prin extragerea rădăcinii cuvintelor a dus la confundarea a doi clusteri pe setul S02 şi implicit la scăderea scorului F-measure. Oricum,

măsura NEWST propusă a demonstrat că aduce îmbunătăţiri. Ipoteza mea este că pe documente „mai semantice” (de exemplu documente care folosesc aceleaşi grupuri de cuvinte în aceeaşi ordine) utilizarea reprezentării STDM împreună cu măsura NEWST este utilă.

4.4.3 Rezultatele obţinute de algoritmul *k*-Medoids cu reprezentare VSM

Pentru a valida rezultatele prezentate în secţiunile 4.4.1 şi 4.4.2 am repetat experimentele utilizând algoritmul *k*-Medoids. Experimentele au fost efectuate pe seturile de date RSS fără şi apoi cu aplicarea algoritmului de extragere a rădăcinilor documentelor.

În continuare, voi prezenta în tabelul 4.9 şi figura 4.14 rezultatele comparative obţinute în condiţiile utilizării algoritmului *k*-Medoids pentru seturile de date S01-S07. S-a utilizat modelul de reprezentare VSM împreună cu distanţa euclidiană, Canberra şi Jaccard.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
Jaccard	**89.40%**	**85.96%**	**71.79%**	**73.45%**	**74.61%**	**63.79%**	**62.01%**	**74.43%**
Euclidiană	56.29%	34.50%	56.41%	22.03%	42.49%	53.45%	22.91%	41.15%
Canberra	35.76%	32.75%	31.41%	31.07%	33.16%	41.38%	27.37%	33.27%

Tabel 4.9 Rezultatele acurateţei obţinute de algoritmul *k*-Medoids – fără stemming cu reprezentarea VSM

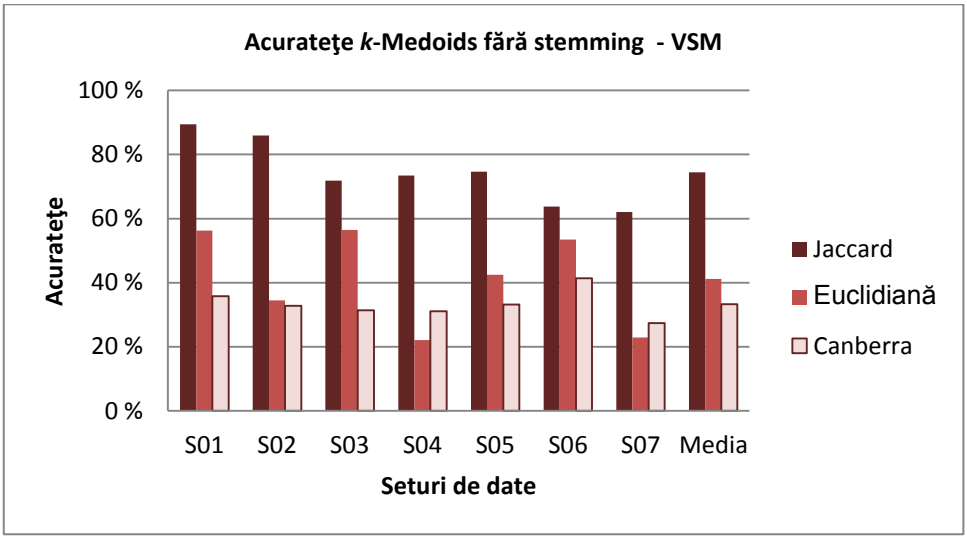


Fig 4.14 *k*-Medoids - Acurateţea calculată pentru seturile S01-S07 fără stemming

Se observă că în cazul utilizării algoritmului *k*-Medoids metrica de similaritate Jaccard obţine în mod clar rezultatele cele mai bune deoarece în cazul seturilor de date utilizate numărul de cuvinte per document este mic iar metricile care folosesc distanţe nu pot diferenţia puternic documentele. Metrica Jaccard fiind o metrică bazată pe numărul cuvintelor este mai sensibilă şi poate diferenţia documentele.

În cazul scorului F-measure, rezultatele obţinute prezentate în tabelul 4.10 şi figura 4.15 arată că măsura Jaccard obţine rezultatele cele mai bune, excepţie făcând doar rezultatul obţinut pe setul S03 unde distanţa euclidiană obţine un scor cu 0.002 mai bun decât distanţa Jaccard. Analizând rezultatul obţinut pentru acurateţe pe setul S03 se observă o diferenţă clară în favoarea distanţei Jaccard. Acest lucru se poate explica din nou prin inversarea claselor a doi clusteri de dimensiune diferită.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
Jaccard	**0.9070**	**0.8746**	0.6374	**0.6438**	**0.6437**	**0.5140**	**0.5055**	**0.6751**
Euclidiană	0.6218	0.3758	**0.6401**	0.2391	0.4816	0.4849	0.2282	0.4388
Canberra	0.3702	0.3551	0.2802	0.2796	0.3254	0.4010	0.2131	0.3178

Tabel 4.10 Rezultatele F-measure obţinute de algoritmul *k*-Medoids – fără stemming cu reprezentarea VSM

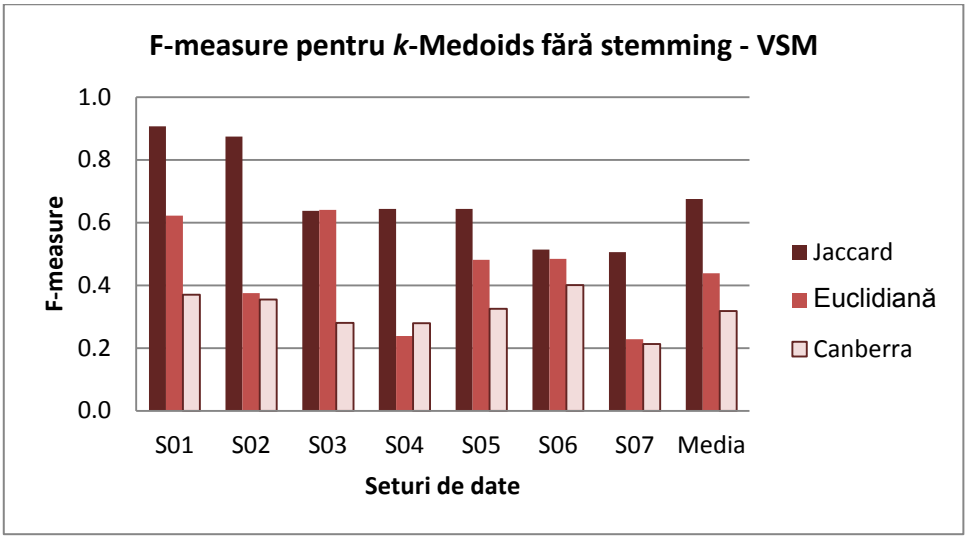


Fig 4.15 *k*-Medoids – F-measure calculat pentru seturile S01-S07 fără stemming

În continuare, voi prezenta rezultatele experimentelor care au fost obţinute pe seturile S01-S07 la care în prealabil am aplicat extragerea rădăcinilor cuvintelor. Rezultatele pentru acurateţe şi F-measure sunt prezentate în figura 4.16 şi respectiv figura 4.17 iar valorile efective obţinute sunt prezentate în tabelele 4.11 respectiv 4.12.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
Jaccard	**92.05%**	**87.72%**	**91.03%**	**76.84%**	**77.72%**	**65.52%**	**59.78%**	**78.66%**
Euclidiană	61.59%	45.03%	50.00%	51.41%	39.90%	51.72%	36.31%	47.99%
Canberra	39.07%	32.75%	40.38%	31.07%	39.90%	41.38%	29.05%	36.23%

Tabel 4.11 Rezultatele *k*-Medoids pentru acurateţe – cu stemming cu reprezentarea VSM

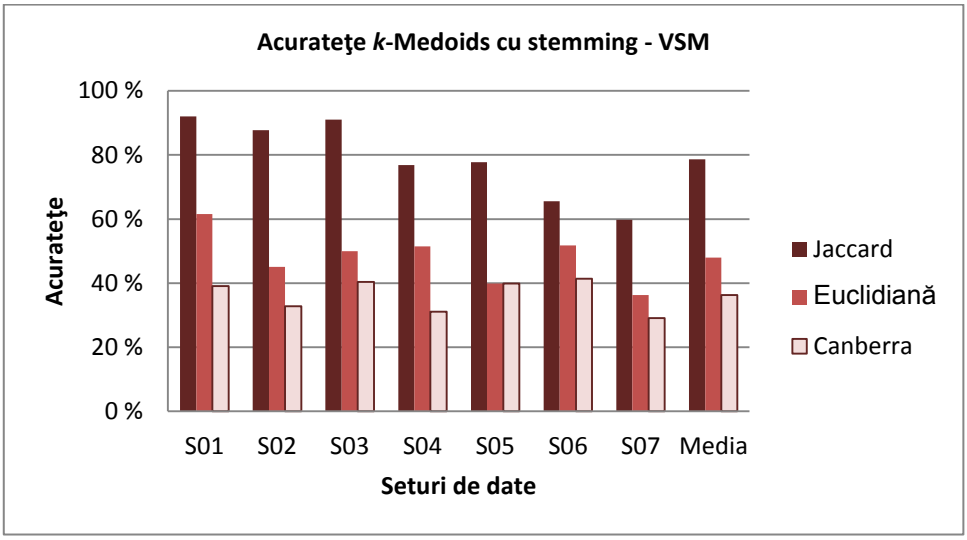


Fig. 4.16 Algoritmul de clustering *k*-Medoids. Acurateţea pe seturi cu stemming – model VSM

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
Jaccard	**0.9169**	**0.8826**	**0.8792**	**0.6619**	**0.6611**	0.5153	**0.5041**	**0.7173**
Euclidiană	0.6687	0.4471	0.4814	0.4926	0.3751	**0.5185**	0.3405	0.4748
Canberra	0.3975	0.3547	0.3753	0.2793	0.3511	0.4012	0.2237	0.3404

Tabel 4.12 Rezultatele *k*-Medoids pentru F-measure – cu stemming cu reprezentarea VSM

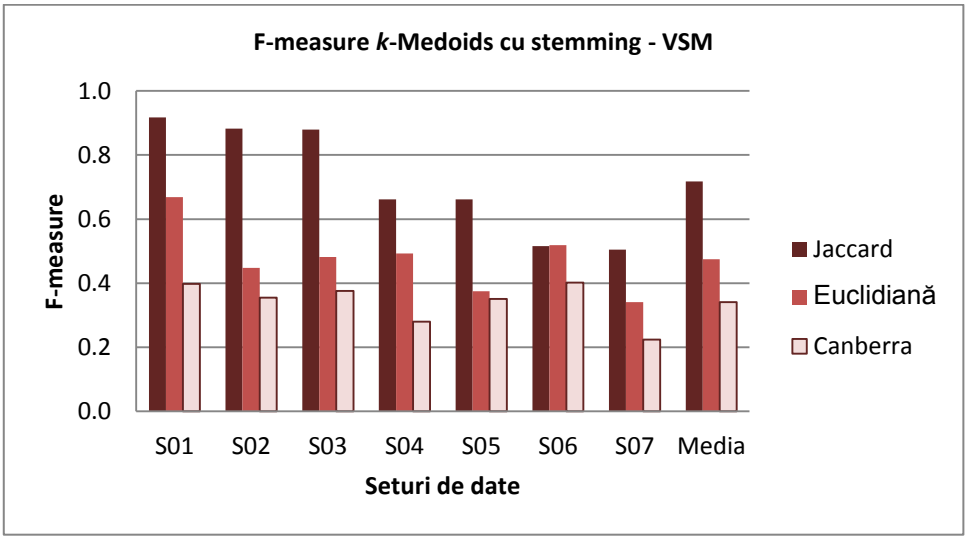


Fig. 4.17 Algoritmul de clustering *k*-Medoids. F-measure pe seturi cu stemming – model VSM

În concluzie, putem afirma că pentru reprezentarea VSM metrica Jaccard a obţinut rezultate mai bune utilizând algoritmul *k*-Medoids decât în cazul utilizării algoritmului HAC. Dacă pentru algoritmul HAC, acurateţea pe seturile RSS fără utilizarea algoritmului de stemming în cazul metricii Jaccard a fost în medie de 63.86%, în cazul algoritmului *k*-Medoids aceasta a ajuns la 74.43%. În cazul aplicării algoritmului de extragere a rădăcinilor cuvintelor acurateţea în cazul utilizării algoritmului HAC cu metrica Jaccard a fost de 40.93% şi a ajuns pentru algoritmul *k*-Medoids, utilizând aceeași măsură Jaccard, la 78.66%. Este evident faptul că, pe seturi de documente relativ mici reprezentate cu ajutorul modelului VSM, algoritmul *k*-Medoids cu distanţa Jaccard şi utilizarea algoritmului de extragere a rădăcinii cuvintelor obţine rezultatele cele mai bune.

4.4.4 Rezultatele obţinute de algoritmul *k*-Medoids cu reprezentare STDM

Pentru a valida utilitatea modelului de reprezentare STDM l-am aplicat cu aceleaşi formule de calcul a similarităţii (formulele NEWST şi OLDST) şi la algoritmul de clustering *k*-Medoids (varianta PAM), care foloseşte la fel ca şi algoritmul HAC matricea de distanţe dintre documente în procesul de creare de clusterilor.

Acurateţea obţinută de algoritmul *k*-Medoids pe seturi la care nu am aplicat algoritmul de stemming sunt prezentate în tabelul 4.13 şi figura 4.18.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
NEWST	89.40%	**88.89%**	**86.54%**	**77.97%**	75.13%	**87.93%**	**64.25%**	**81.44%**
OLDST	**92.72%**	87.13%	81.41%	46.89%	**78.76%**	72.41%	63.13%	74.64%

Tabel 4.13 Rezultatele *k*-Medoids pentru acurateţe– fără stemming cu reprezentarea STDM

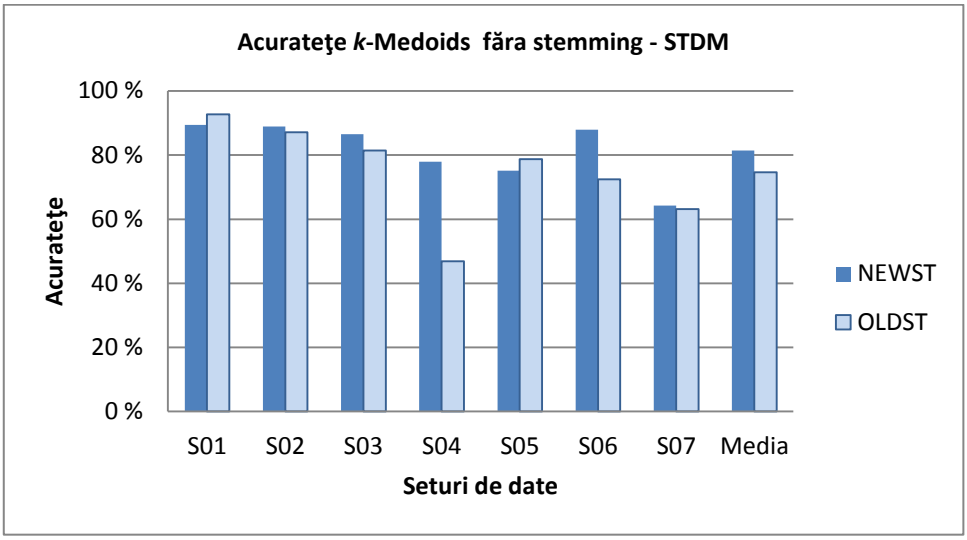


Fig. 4.18 Algoritmul de clustering *k*-Medoids Acurateţea pe seturi fără stemming – model STDM

Ca şi în cazul algoritmului HAC şi la algoritmul *k*-Medoids se observă că metrica NEWST, propusă de mine, obţine rezultate mai bune decât metrica OLDST. Astfel, pentru setul de date pe care nu am aplicat algoritmul de stemming, rezultatele privind îmbunătăţirea acurateţei adusă de metrica NEWST faţă de metrica OLDST sunt prezentate în Fig. 4.19.

Îmbunătăţirea acurateţei pentru rezultatul clusteringului am calculat-o ca diferenţă dintre acurateţea obţinută de metrica NEWST şi metrica OLDST. În cazul când metrica NEWST a obţinut un rezultat mai bun decât metrica OLDST, diferenţa este pozitivă, altfel ea devine negativă. Se observă o îmbunătăţire a acurateţei în medie de 6.81% adusă de utilizarea metricii NEWST. Reamintim faptul că, metrica NEWST a obţinut în cazul algoritmului HAC în aceleaşi condiţii de reprezentare a documentelor, o îmbunătăţire a acurateţei cu 1.49% faţă de metrica OLDST.

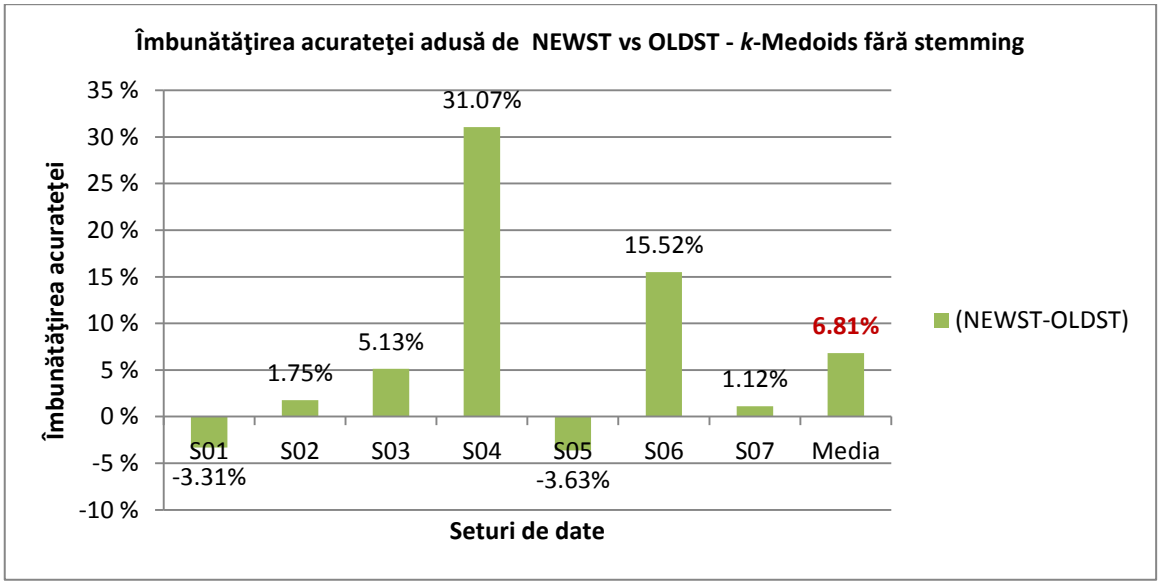


Fig. 4.19 *k*-Medoids – Îmbunătăţirea acurateţei adusă de NEWST vs. OLDST în utilizând reprezentarea STDM pe seturile de date fără stemming

Rezultatele obţinute, utilizând F-measure ca metrică de evaluare, pentru *k*-Medoids cu reprezentarea STDM fără stemming sunt prezentate în tabelul 4.14 şi figura 4.20.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
NEWST	0.8958	**0.8903**	**0.8390**	**0.6619**	0.6455	**0.8883**	**0.5167**	**0.7625**
OLDST	**0.9288**	0.8708	0.6783	0.4111	**0.6926**	0.7311	0.5105	0.6890

Tabel 4.14 Rezultatele *k*-Medoids pentru F-measure– fără stemming cu reprezentarea STDM

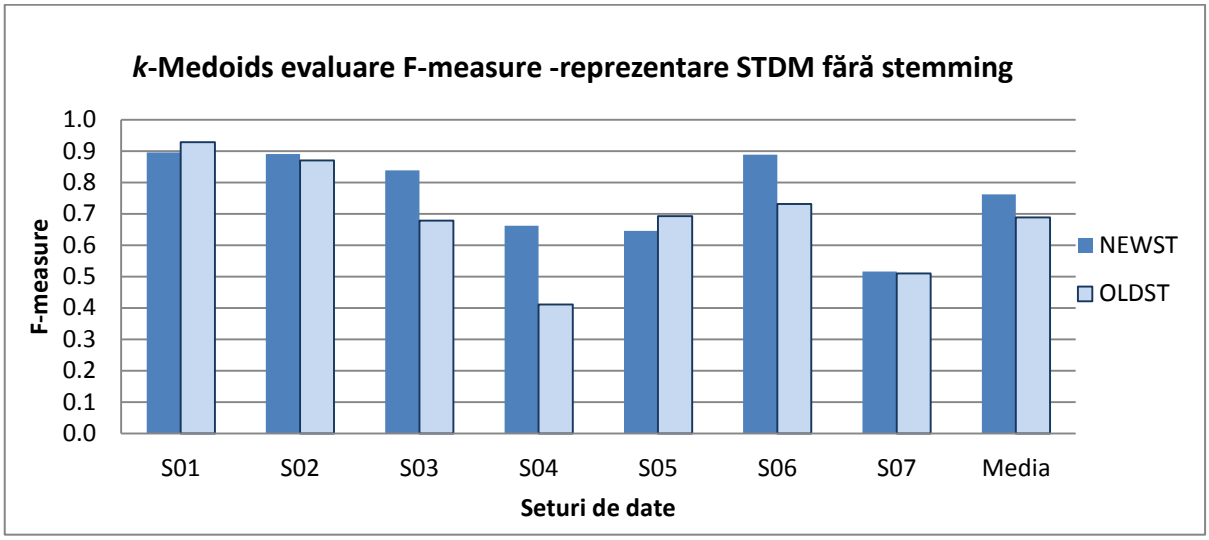


Fig. 4.18 Algoritmul de clustering *k*-Medoids. F-measure pe seturi fără stemming – model STDM

Pentru a valida îmbunătăţirea adusă de metrica NEWST faţă de metrica OLDST am calculat şi în cazul scorului F-measure diferenţa dintre metrica NEWST şi metrica OLDST. Şi în acest caz, se obţine o diferenţă pozitivă în cazul în care metrica NEWST obţine un rezultat mai bun decât metrica OLDST, diferenţa fiind negativă în caz contrar. În medie, rezultate în cazul F-measure pentru metrica NEWST faţă de metrica OLDST sunt cu 0.735 mai bune ca metrica OLDST. Valorile pentru îmbunătăţirea scorului F-measure adusă de metrica NEWST faţă de metrica OLDST sunt prezentate în Fig. 4.21.

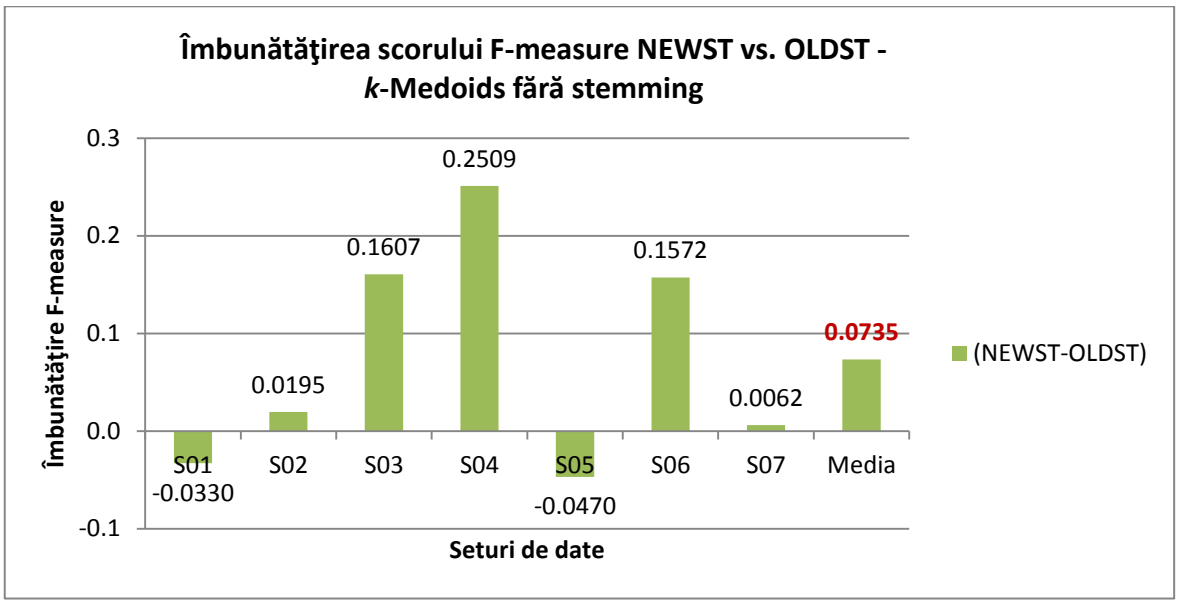


Fig. 4.21 *k*-Medoids –Îmbunătăţirea scorului F-measure NEWST vs. OLDST în reprezentarea seturile de date fără stemming

În continuare, voi prezenta în tabelul 4.15 şi figura 4.22 rezultatele obţinute de algoritmul *k*-Medoids pe seturile de date reprezentate utilizând modelul STDM şi aplicarea algoritmului de stemming.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
NEWST	88.08%	**88.89%**	**86.54%**	**77.97%**	75.13%	**87.93%**	**73.74%**	**82.61%**
OLDST	**91.39%**	70.18%	83.33%	46.89%	**77.72%**	72.41%	63.13%	72.15%

Tabel 4.15 Rezultatele *k*-Medoids pentru acurateţe– cu stemming cu reprezentarea STDM

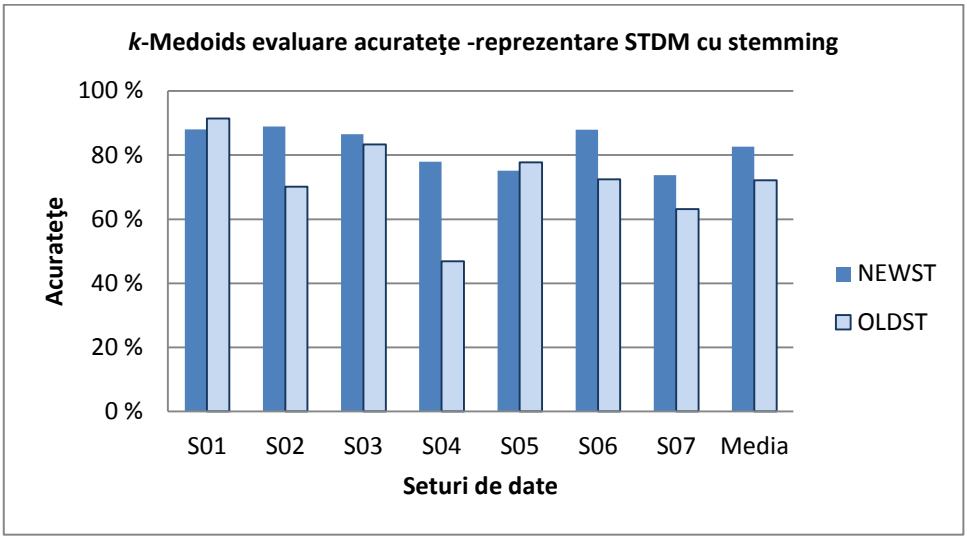


Fig. 4.22 Algoritmul de clustering *k*-Medoids. Acurateţea pe seturi cu stemming – model STDM

Analizând rezultatele obţinute de metrica NEWST se observă că, din nou, obţine rezultate superioare faţă de metrica OLDST.

În continuare, calculez îmbunătăţirea adusă de metrica NEWST ca diferenţă între rezultatele obţinute de cele două metrici în cazul utilizării algoritmului de stemming iar rezultatele sunt prezentate în Fig. 4.23.

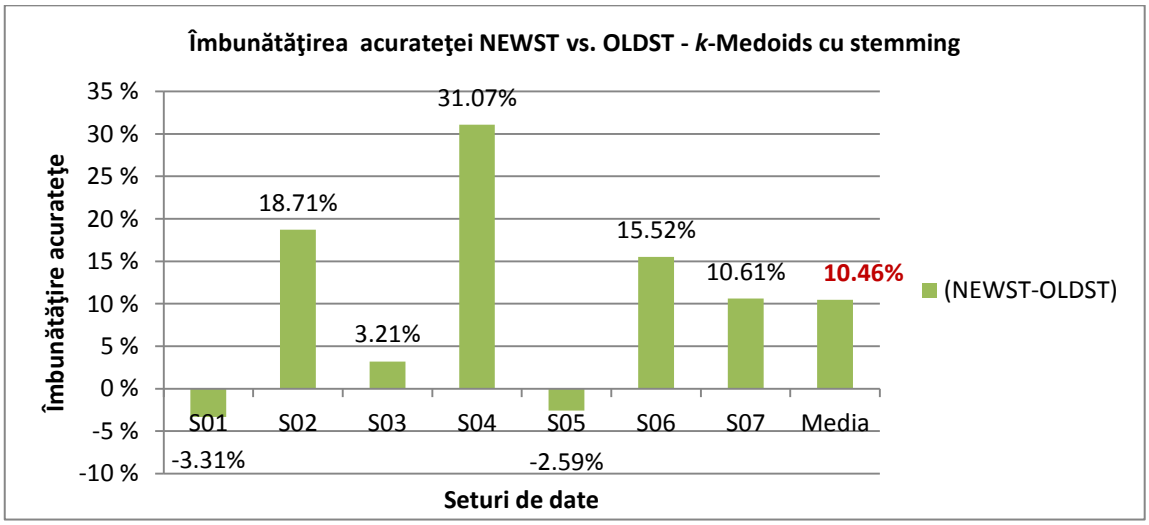


Fig. 4.23 *k*-Medoids – Îmbunătăţirea acurateţei NEWST vs. OLDST în reprezentarea seturile de date cu stemming

Dacă comparăm îmbunătăţirea adusă de măsura NEWST faţă de măsura OLDST în cazul neutilizării algoritmului de stemming, superioară cu 6.81%, faţă de îmbunătăţirea de 10.46% adusă de măsura NEWST utilizând algoritmul de stemming, putem afirma că aplicarea algoritmului de stemming pe seturile de date a fost în cazul algoritmului *k*-Medoids benefică deoarece algoritmul *k*-Medoids fiind un algoritm partiţional nu acceptă clusteri suprapuşi cum este cazul algoritmului HAC iar utilizarea a mai puţine atribute permite o delimitare mai uşoară a clusterilor.

În tabelul 4.16 şi figura 4.24 prezint scorul F-measure obţinut de algoritmul *k*-Medoids pe seturile de date la care am aplicat algoritmul de stemming.

Setul / Metrica	S01	S02	S03	S04	S05	S06	S07	Media
NEWST	0.8882	**0.8903**	**0.8390**	**0.6619**	0.6555	**0.8883**	**0.5513**	**0.7678**
OLDST	**0.9183**	0.7101	0.6981	0.4111	**0.6632**	0.5466	0.5105	0.6368

Tabel 4.16 Rezultatele *k*-Medoids pentru F-measure– cu stemming cu reprezentarea STDM

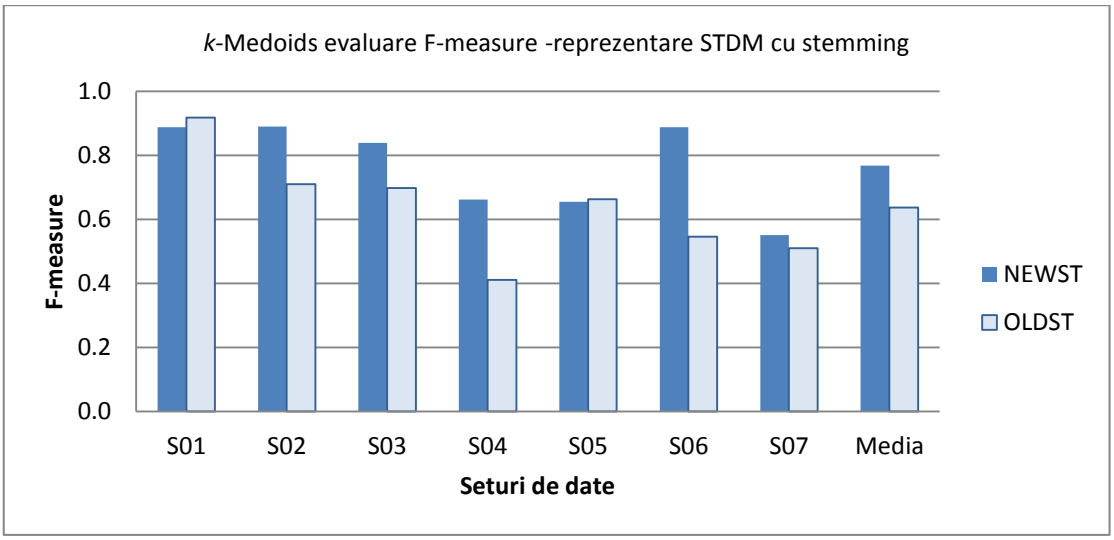


Fig. 4.24 Algoritmul de clustering *k*-Medoids. F-measure pe seturi fără stemming – model STDM

Calculând diferenţele dintre rezultatele obţinute de metrica NEWST faţă de metrica OLDST pe baza F-measure în cazul algoritmului *k*-Medoids am obţinut rezultatele prezentate în Fig. 4.25.

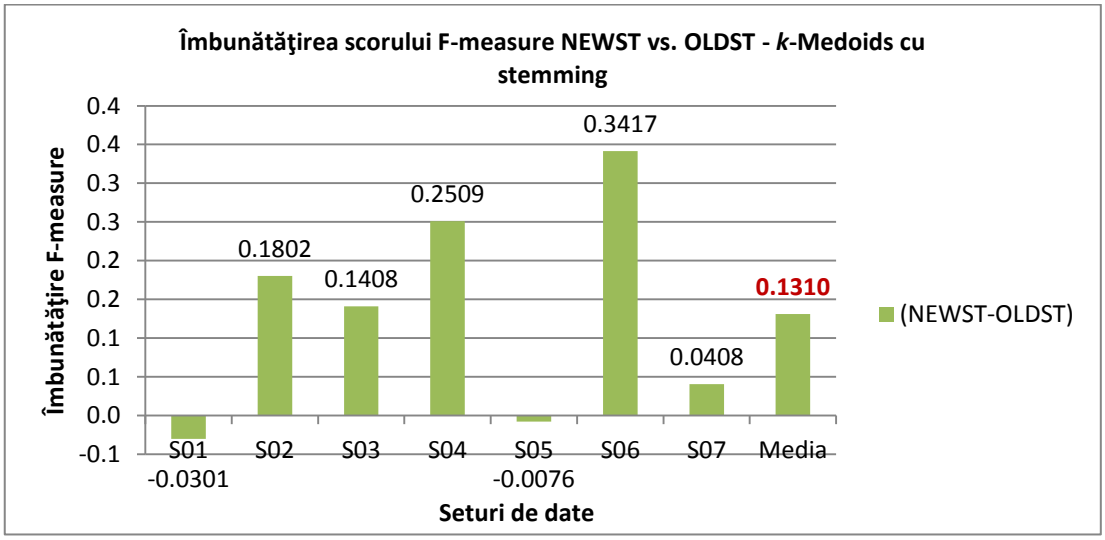


Fig. 4.25 K-Medoids – Îmbunătăţirea F-measure NEWST vs. OLDST în reprezentarea seturile de date cu stemming

4.4.5 Comparaţii între algoritmii de clustering şi între modurile de reprezentare. Superioritatea metricii propuse

Rezultatele prezentate mai sus demonstrează faptul că utilizarea modelului STDM pentru reprezentarea documentelor text este aplicabilă și la alți algoritmi de clustering, alții decât STC dar care folosesc în procesul de antrenare matrice de distanţe. Modelul STDM devine mai greu de utilizat în cazul algoritmilor de clustering care utilizează centroizi cum ar fi algoritmul k-Means. În implementarea algoritmului *k*-Medoids am folosit implementarea PAM a acestuia care nu necesită simpla calculare a unui centru de greutate şi alegerea documentului care este cel mai apropiat faţă de acesta, ci căutarea exhaustivă a medoizilor, fapt ce a dus implicit la creșterea timpului de execuţie. Introducerea implicită a unor elemente indirecte de „semantică” a documentelor – în cazul nostru ordinea cuvintelor – a dus la o îmbunătățire semnificativă a rezultatelor clusteringului. Prezentăm în tabelele 4.17 şi 4.18 rezultatele experimentelor pentru valorile medii, obținute de algoritmul HAC cu reprezentările STDM şi VSM pe toate cele 7

seturi de date, atât din punct de vedere al acurateţei cât şi din punct de vedere al măsurii F-measure. De asemenea, valorile sunt prezentate separat pentru date pe care s-a făcut stemming şi pe date pe care nu s-a făcut stemming.

Modelul	Metrica	Fără stemming	Cu stemming
VSM	Jaccard	63.86%	40.93%
	Euclidian	36.26%	37.04%
	Canberra	34.42%	34.24%
STDM	NEWST	**87.23%**	**87.23%**
	OLDST	85.75%	85.75%

Tabelul 4.17 Rezultate ale acurateţei pe valori medii pentru algoritmul HAC

Modelul	Metrica	Fără stemming	Cu stemming
VSM	Jaccard	0.6309	0.4316
	Euclidian	0.3757	0.3693
	Canberra	0.3842	0.3739
STDM	NEWST	**0.8599**	**0.8557**
	OLDST	0.8144	0.8303

Tabelul 4.18 Rezultate ale F-measure pe valori medii pentru algoritmul HAC

În Fig.4.26 şi Fig. 4.27 prezint grafic rezultatele obţinute de algoritmul HAC pentru acurateţe respectiv F-measure.

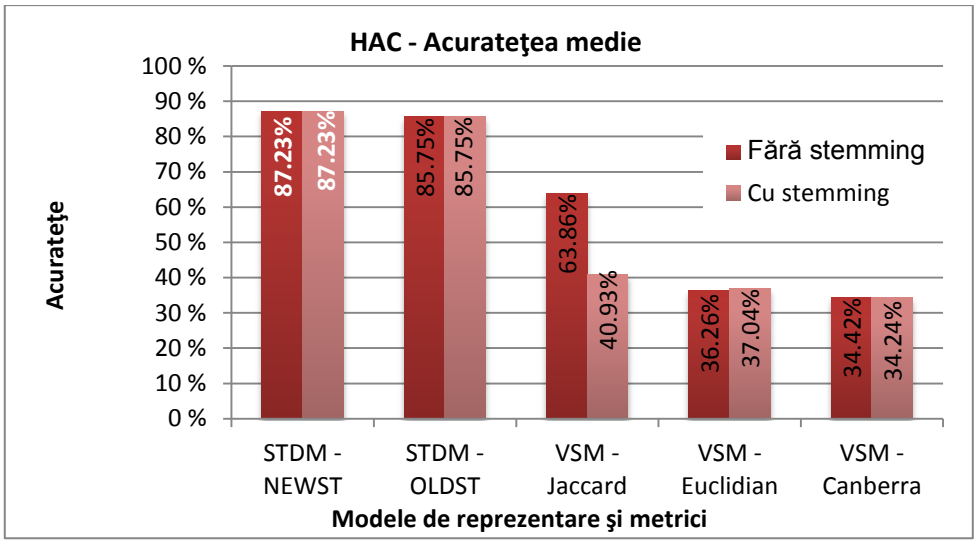


Fig. 4.26 HAC - Acurateţea medie pentru toate cele 7 seturi de date

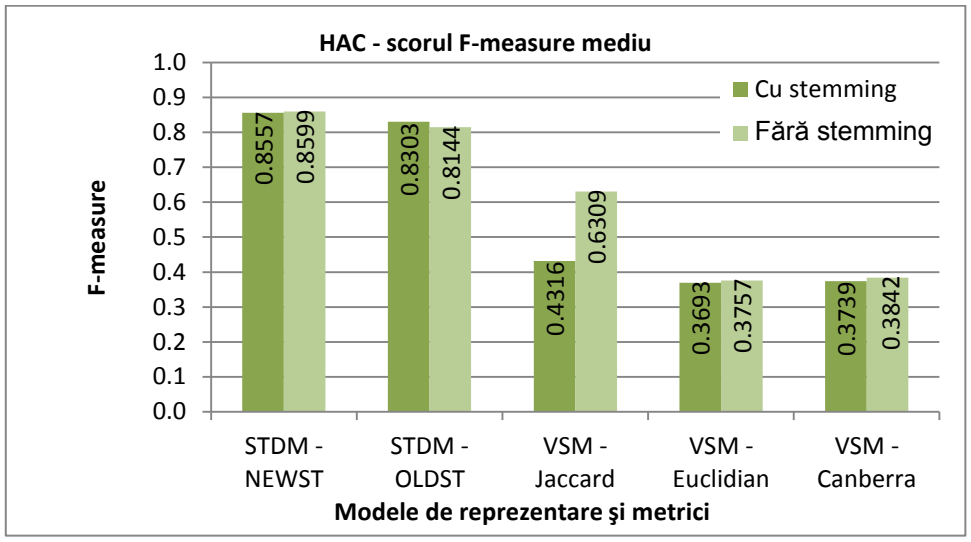


Fig. 4.27 HAC – F-measure mediu pentru toate cele 7 seturi de date

Din rezultatele prezentate în Fig.4.26 se observă că, în cazul utilizării modelului STDM acurateţea algoritmului de clustering nu este afectată de preprocesarea datelor, respectiv extragerea sau nu a rădăcinilor cuvintelor. Acest fapt este explicabil deoarece prin extragerea rădăcinilor nu se pierde ordinea cuvintelor deci, se păstrează oarecum „semantica" documentului. Pentru modelul VSM extragerea rădăcinilor cuvintelor este benefică doar pentru distanţa euclidiană şi se constată o îmbunătăţire nesemnificativa, cu doar 0,78% a acurateţei. În cazul celorlalte distanţe utilizate în modelul vectorial, se constată o înrăutăţire: în cazul distanţei Canberra de 0,18% iar în cazul distanţei Jaccard înrăutăţirea este de 22,93%. Acest fapt este explicabil deoarece distanţa Jaccard se bazează pe numărul de elemente comune între două mulţimi iar extrăgând rădăcinile s-au obţinut mai multe cuvinte identice pierzându-se astfel diferenţa între documente iar acurateţea a avut de suferit.

În tabelele 4.19 şi 4.20 voi prezenta rezultatele obţinute pentru valorile medii ale acurateţei şi ale scorului F-measure obţinute de algoritmul *k*-Medoids, calculate similar ca în cazul algoritmului HAC, utilizând modelele de reprezentare STDM şi VSM.

Model	Metrica	Fără stemming	Cu stemming
VSM	Jaccard	74.43%	78.66%
	Euclidiana	41.15%	47.99%
	Canberra	33.27%	36.23%
STDM	NEWST	**81.44%**	**82.61%**
	OLDST	74.64%	72.15%

Tabelul 4.19 Rezultate ale acurateţei pe valori medii pentru algoritmul *k*-Medoids

Model	Metrica	Fara stemming	Cu stemming
VSM	Jaccard	0.7034	0.7173
	Euclidiană	0.4739	0.4748
	Canberra	0.3352	0.3404
STDM	NEWST	**0.8035**	**0.7678**
	OLDST	0.7188	0.6368

Tabelul 4.20 Rezultate ale F-measure pe valori medii pentru algoritmul *k*-Medoids

În Fig.4.28 şi Fig. 4.29 prezint grafic rezultatele obţinute de algoritmul *k*-Medoids pentru acurateţe respectiv F-measure.

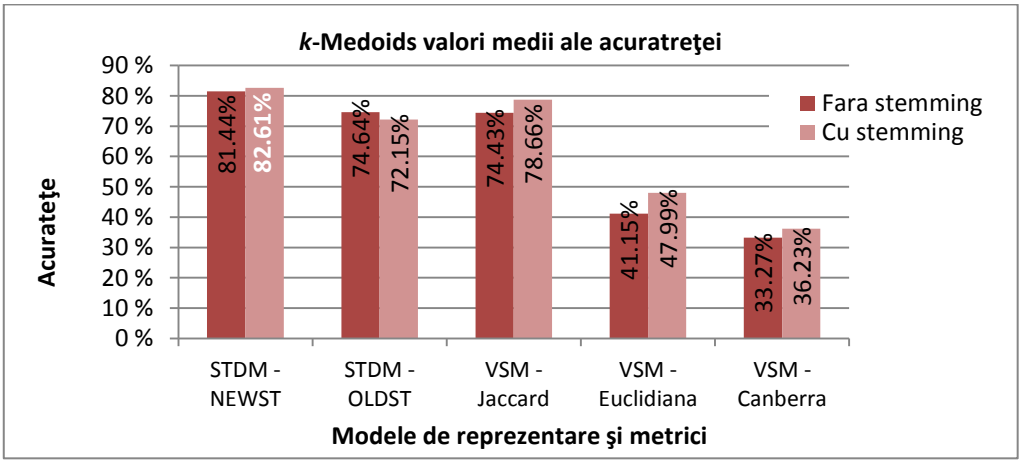


Fig. 4.28 *k*-Medoids - Acurateţea medie pentru toate cele 7 seturi de date

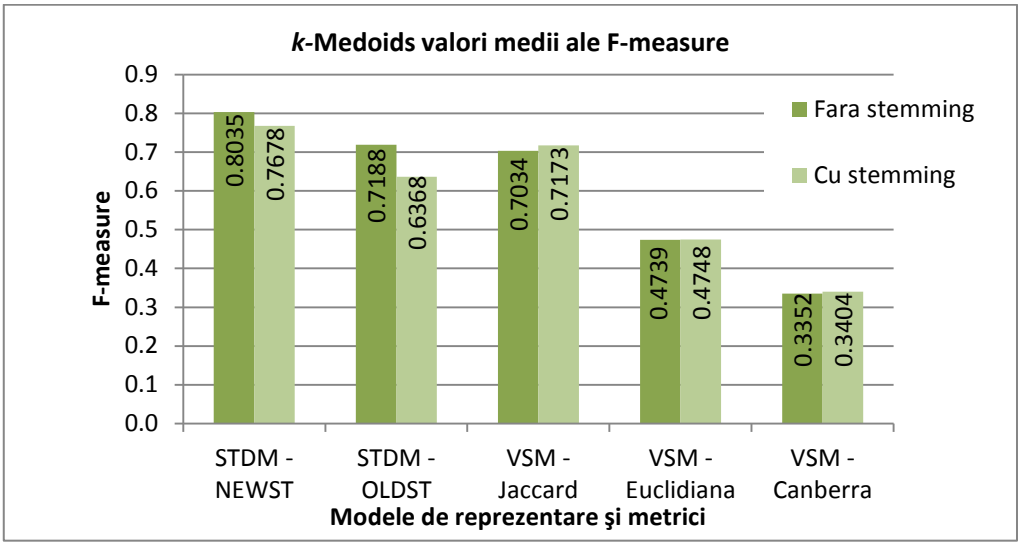


Fig. 4.29 *k*-Medoids – F-measure mediu pentru toate cele 7 seturi de date

Se observă, la fel ca şi în cazul algoritmului HAC, că extragerea rădăcinilor cuvintelor nu influenţează semnificativ acurateţea în modelul STDM. Se observă chiar o uşoară scădere a acurateţei pentru seturile de date în care a fost aplicat algoritmul de stemming (0,22% pentru NEWST şi 2,90% pentru OLDST). Situaţia este complet schimbată în cazul modelului VSM de reprezentare. În cazul algoritmului *k*-Medoids - care este un algoritm partiţional şi nu acceptă suprapuneri în ceea ce priveşte clusterii rezultaţi - s-au obţinut rezultate mai bune în cazul utilizării modulului de extragere a rădăcinilor cuvintelor.

Multe studii care au utilizat reprezentarea VSM au arătat că utilizarea unui număr mai mic de cuvinte prin extragerea rădăcinii acestora duce la îmbunătăţiri ale calităţii grupării datelor [Mora06_c]. O altă problemă care apare este calitatea procesului de stemming. Pentru extragerea rădăcinii cuvintelor am folosit implementarea Porter din pachetul Information Retrieval de la Universitatea din Texas [IR] preluat în anul 2007. Chiar daca s-a observat de către comunitatea ştiinţifică faptul că această implementare nu extrage întotdeauna corect rădăcina cuvintelor ea este referinţa în domeniu şi am folosit-o ca atare.

O altă explicaţie a faptului că acurateţea medie scade pe seturile fără stemming la algoritmul *k*-Medoids comparativ cu algoritmul HAC este aceea că algoritmul *k*-Medoids nu acceptă suprapuneri pentru grupele create iar utilizarea unui număr mai mic de cuvinte ajută algoritmul să găsească mai uşor liniile de separare între grupele create.

Astfel, se obţine o îmbunătăţire a acurateţei cu 5,75% în cazul distanţei euclidiene, cu 5,31% în cazul distanţei Jaccard şi cu 3,17% în cazul distanţei Canberra.

Din rezultatele prezentate se observă că utilizarea măsurilor de similaritate (categoria A) obţine rezultate mai bune decât cele de disimilaritate (categoria B), aceasta explicându-se prin faptul că distanţele din categoria A calculate în matricea de distanţe sunt uniform distribuite în domeniul [0, 1]. Metricile de distanţă din categoria B pot returna valori foarte apropiate de 0 pentru majoritatea cazurilor în momentul în care există în set două documente pentru care distanţa între ele este foarte mare comparativ cu celelalte distanţe şi astfel se poate intra în problema care apare în momentul reprezentării numerelor foarte mici. O altă diferenţă între metricile NEWST, OLDST şi celelalte metrici folosite este aceea că NEWST şi OLDST folosesc reprezentarea STDM a documentelor având şi informaţii de „semantică” (doar ordinea cuvintelor din document) comparativ cu celelalte metrici care folosesc reprezentarea VSM. Distanţa Jaccard obţine rezultate mai bune decât celelalte metrici care utilizează modelul VSM de reprezentare dar obţine rezultate mai slabe decât metricile care utilizează şi reprezentarea STDM a documentelor.

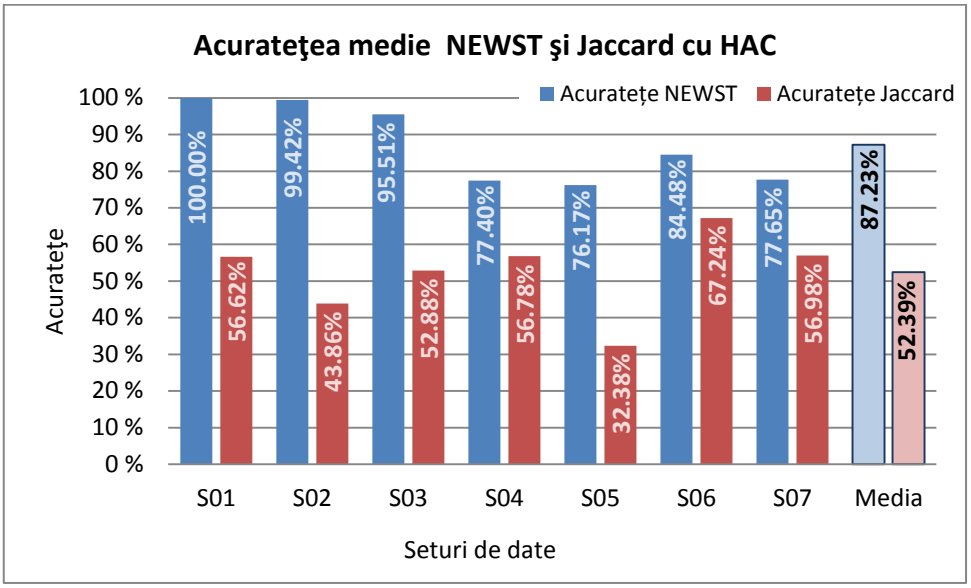


Fig. 4.30 HAC - acurateţei pentru NEWST utilizând modelul STDM şi Jaccard utilizând modelul VSM

În Fig. 4.30 prezint comparativ acurateţea obţinută de „cele mai bune" metrici pentru modelul STDM respectiv, pentru modelul VSM. Pentru modelul STDM s-a ales metrica care a obţinut cele mai bune rezultate şi aceasta este metrica introdusă de noi NEWST iar pentru reprezentarea VSM am ales metrica Jaccard. Valorile prezentate reprezintă media obţinută de cele două metrici NEWST şi Jaccard pe seturile S01-S07 cu stemming şi fără stemming.

În Fig. 4.31 prezint comparativ rezultatele scorului F-measure obţinute de cele două metrici comparate în Fig. 4.30 pentru modelele de reprezentare STDM respectiv VSM. Valorile reprezentate sunt calculate ca medii ale valorilor obţinute pentru scorul F-measure pe seturile de date cu stemming şi fără stemming de către metrica NEWST şi Jaccard.

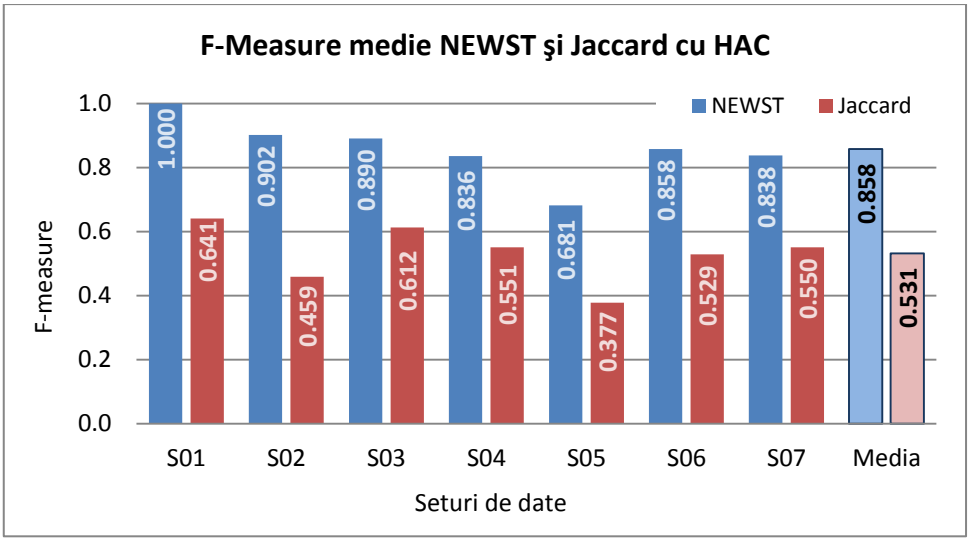


Fig. 4.31 HAC - Media scorului F-measure pentru NEWST utilizând modelul STDM şi Jaccard utilizând modelul VSM

Se constată o îmbunătăţire importantă a rezultatului clusteringului folosind modelul STDM. În figura 4.32 prezint diferenţa dintre rezultatele obţinute utilizând modelul STDM şi modelul VSM.

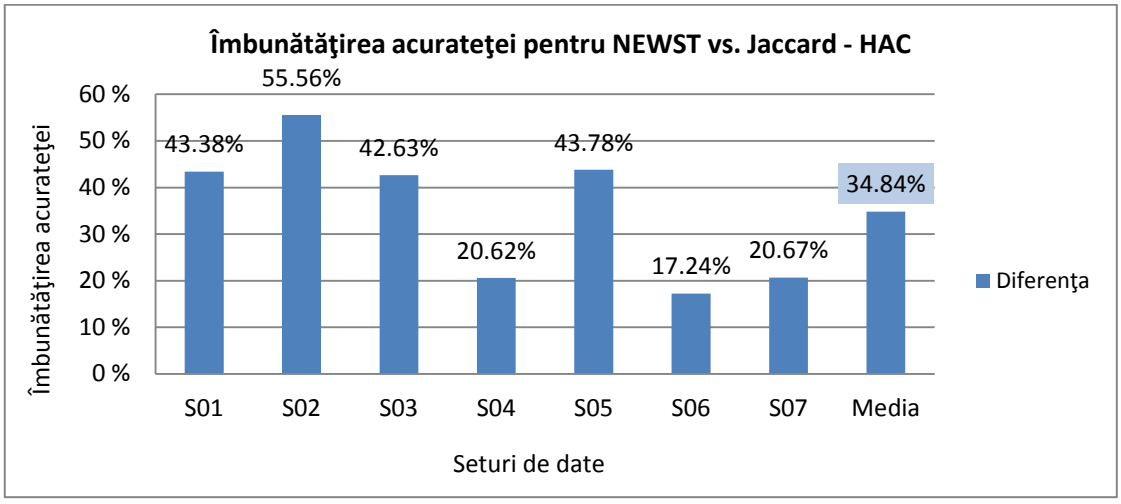


Fig. 4.32 HAC - Îmbunătăţirea acurateţei medie pentru NEWST vs. Jaccard

Ca şi în cazul utilizării algoritmului HAC împreună cu modelul STDM şi în cazul algoritmului *k*-Medoids, utilizarea modelului STDM a îmbunătăţit rezultatele clusteringului.

În Fig.4.33 respectiv Fig. 4.34 prezentăm valorile obţinute pentru acurateţe respectiv scorul F-measure pentru seturile de date S01-S07 obţinute de metrica NEWST şi modelul STDM şi metrica Jaccard şi modelul VSM.

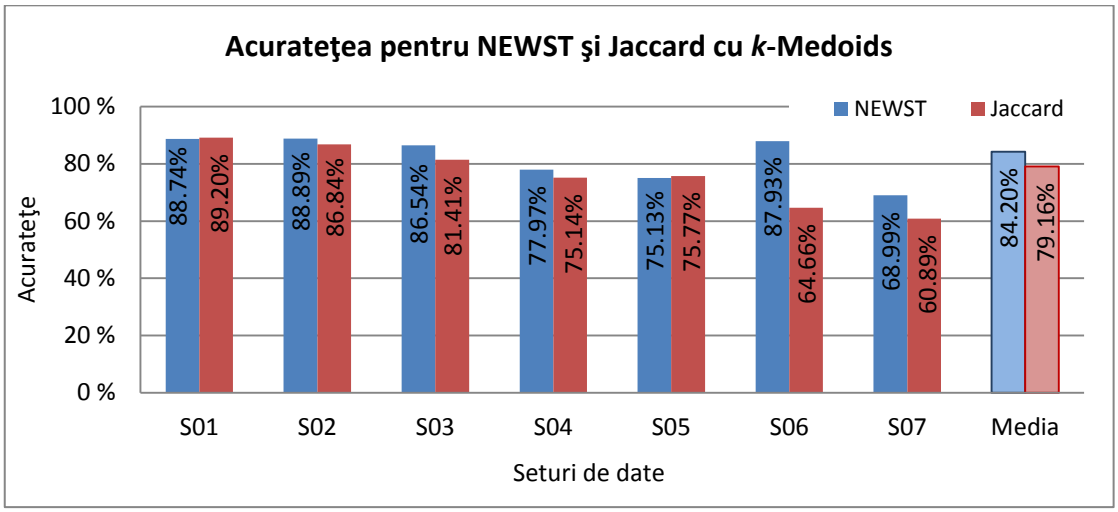


Fig. 4.33 *k*-Medoids – Acurateţea obţinută pentru cele două tipuri de reprezentări STDM (cu NEWST) şi VSM (cu Jaccard)

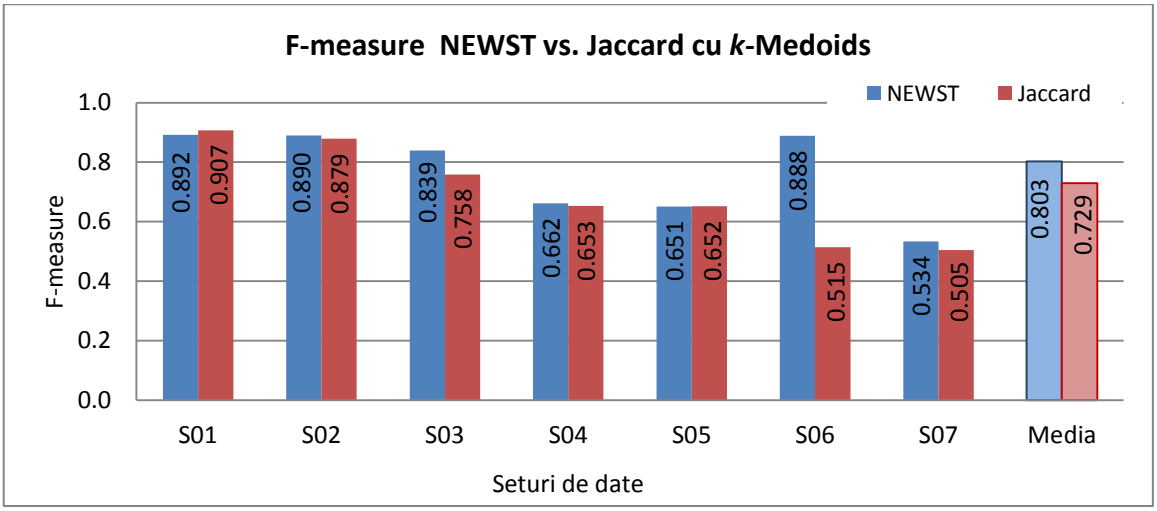


Fig. 4.34 K-Medoids –F-measure obţinut pentru cele două tipuri de reprezentări STDM (cu NEWST) şi VSM (cu Jaccard)

În Fig.4.35 se prezintă diferenţa acurateţei dintre modelul STDM şi VSM precum şi îmbunătăţirea medie pe toate seturile adusă de utilizarea modelului STDM pentru algoritmul *k*-Medoids

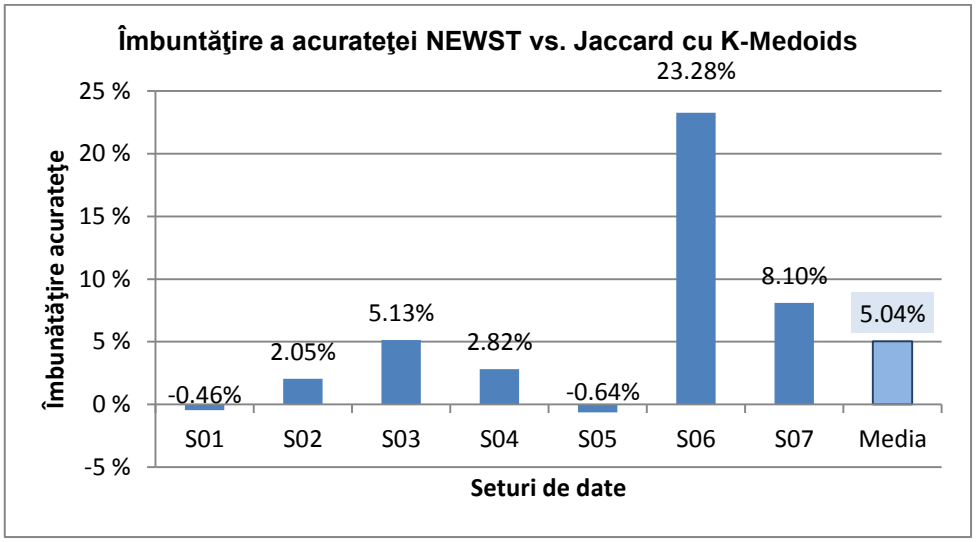


Fig. 4.35. Îmbunătăţirea acurateţei prin utilizarea modelului STDM cu *k*-Medoids.

În cazul utilizării algoritmului *k*-Medoids, modelul de reprezentare STDM împreună cu metrica NEWST a obţinut rezultate superioare modelului VSM cu metrica Jaccard. Îmbunătăţirea de 5.04% (Fig. 4.35) adusă de metrica NEWST pentru algoritmul k-Medoids este mai modestă decât îmbunătăţirea de 34.84% (Fig. 4.32) adusă de metrica NEWST pentru algoritmul HAC.

În continuare prezint comparativ rezultatele obţinute de metrica NEWST şi OLDST pentru algoritmul HAC (NEWST având cele mai bune rezultate pe acest algoritm) şi pentru algoritmul *k*-Medoids (NEWST având şi în acest caz cele mai bune rezultate pe acest algoritm).

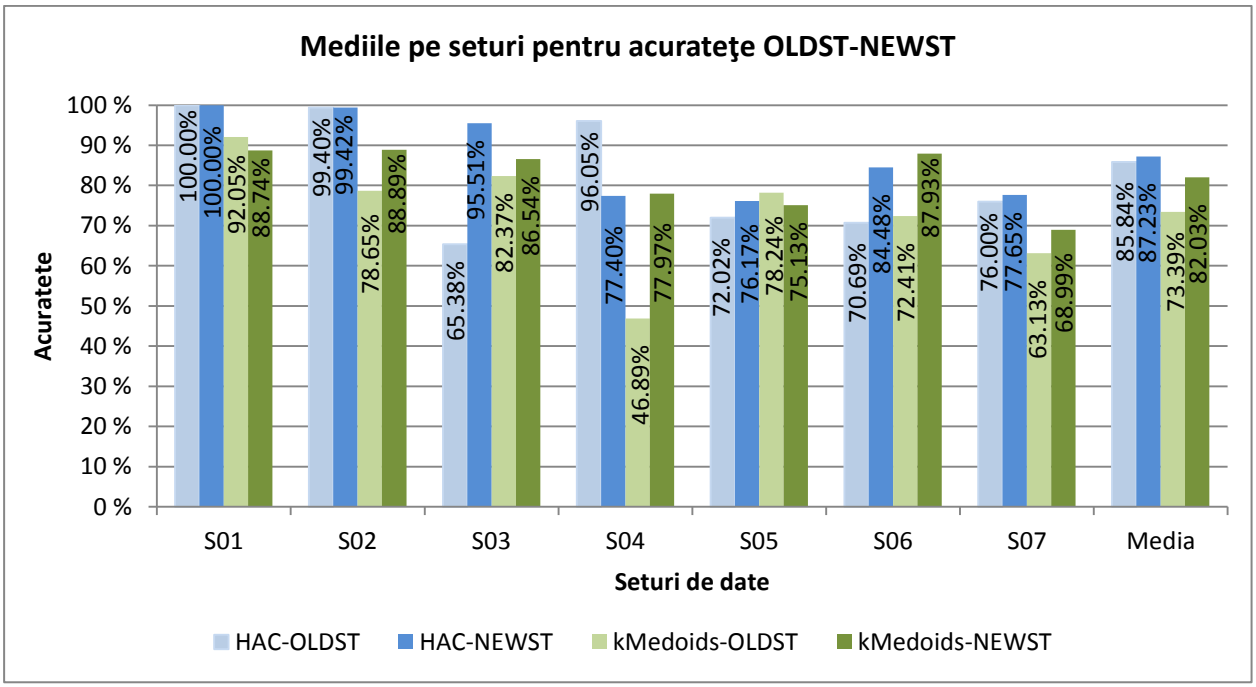


Fig. 4.36 Compararea acurateţei obţinute de măsurile NEWST şi OLDST pentru algoritmul HAC şi *k*-Medoids.

Din figura 4.36 se poate observa că algoritmul HAC împreună cu metrica NEWST şi reprezentarea STDM obţine cea mai bună acurateţe a clasificării. Doar în cazul setului S06 care este un set care conţine şase clase dar cu puţine documentele algoritmul k-Medoids obţine

rezultate mai bune deoarece documentele nu sunt „semantic” suprapuse. În figura 4.37 prezint scorul F-measure obţinut de NEWST în cazul algoritmilor HAC şi *k*-Medoids.

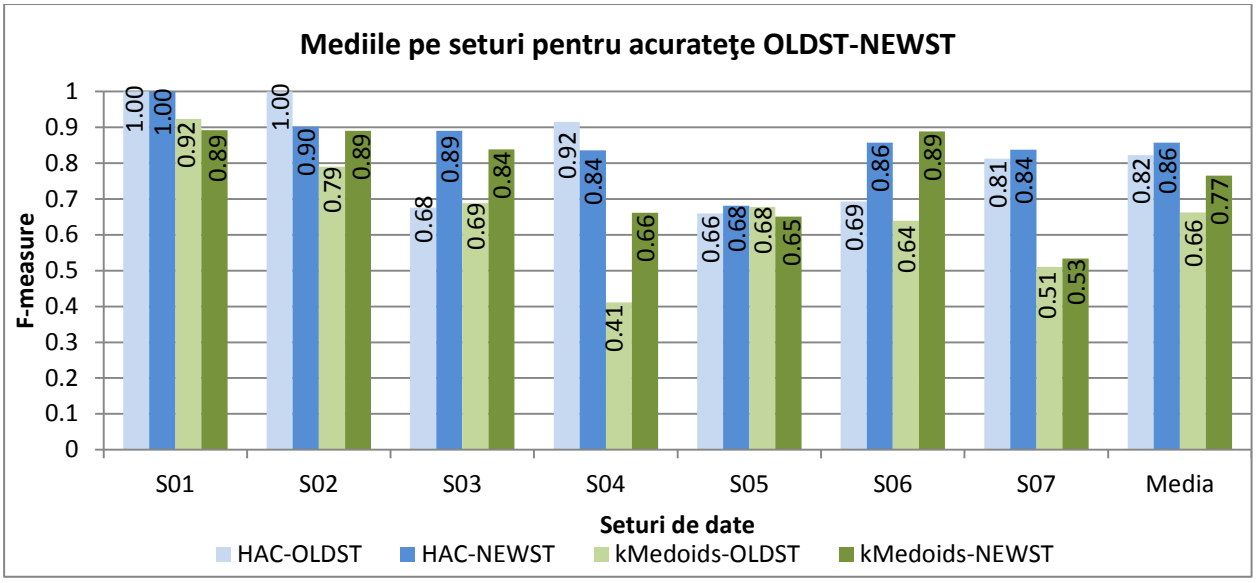


Fig. 4.37 Compararea scorului F-measure obţinut de măsurile NEWST şi OLDST pentru algoritmul HAC şi *k*-Medoids.

Şi în cazul scorului F-measure putem concluziona că metrica NEWST în reprezentarea STDM cu algoritmul HAC pentru documente care sun semantic „mai asemănătoare” obţine rezultate bune.

În tabelele 4.21 şi 4.22 voi prezenta timpii de execuţie necesari pentru algoritmul HAC – single link pentru seturile de date S01-S07 pe un sistem Intel(R) Core2 Duo cu procesoare T6600 la 2,20GHz, 4GB DRAM şi sistem de operare Windows7 Home Premium licenţiat. Timpii din tabel sunt măsuraţi în secunde.

Model	Metrica	S01	S02	S03	S04	S05	S06	S07
STDM	NEWST	66.7	92.6	68.6	97.5	116.9	6.5	146.3
	OLDST	66.7	92.1	68.7	95.8	116.5	6.5	122.3
VSM	Jaccard	**1.0**	**1.0**	**0.9**	**1.1**	**1.2**	**0.3**	**1.0**
	Euclidian	4.3	5.7	4.4	5.8	7.2	0.5	6.1
	Canberra	**1.0**	1.2	1.0	1.3	1.5	**0.3**	1.3

Tabelul 4.21 Rezultate obţinute pentru timpii de execuţie pe algoritmul HAC pe seturi cu stemming

Model	Metrica	S01	S02	S03	S04	S05	S06	S07
STDM	NEWST	67.9	85.6	67.3	93.5	117.0	6.5	96.1
	OLDST	66.3	87.7	68.3	93.8	116.0	6.6	96.2
VSM	Jaccard	**0.9**	**1.2**	**0.9**	**1.2**	**1.3**	0.5	**1.3**
	Euclidian	4.6	6.2	4.8	6.8	8.5	0.5	6.9
	Canberra	1.1	1.4	1.1	1.4	1.8	**0.3**	1.4

Tabelul 4.22 Rezultate obţinute pentru timpii de execuţie pe algoritmul HAC pe seturi fără stemming

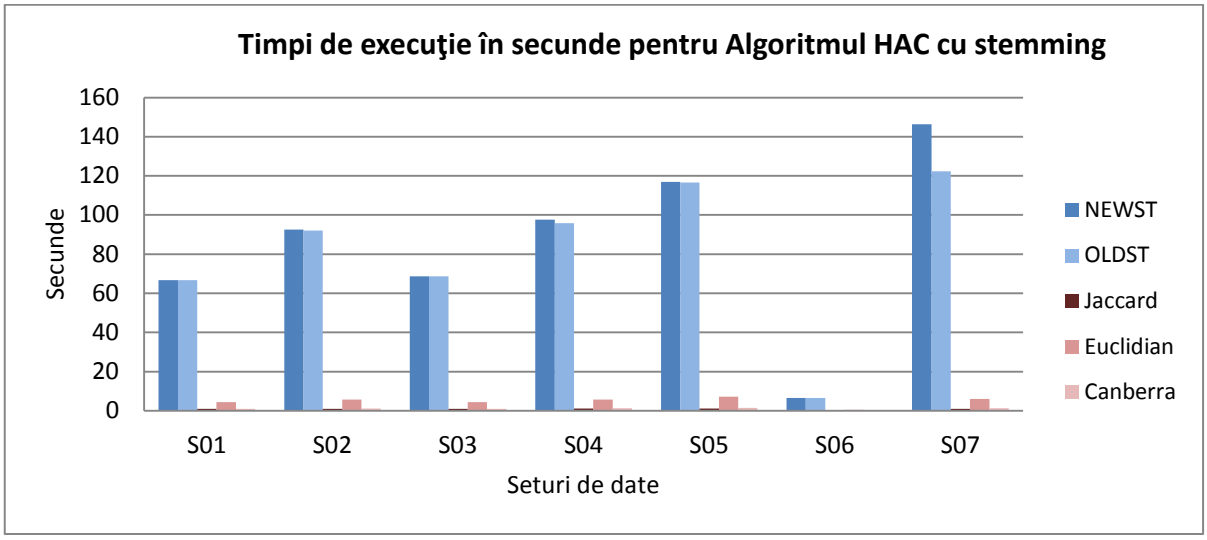


Fig.4.38. Timpii de execuţie pentru algoritmul HAC pe seturile de date S01-S07 cu stemming.

În figura 4.38 sunt prezentaţi timpii de execuţie ai algoritmului HAC pentru seturile de date S01-S07 la care am aplicat stemming. Se poate observa că timpii obţinuţi de algoritm utilizând măsurile Jaccard, Canberra şi euclidiană cu reprezentarea VSM se încadrează în intervalul 0.3 secunde până la 7.2 secunde. În cazul utilizării distanţei euclidiene, algoritmul este puţin mai lent.

În cazul utilizării distanţelor NEWST şi OLDST împreună cu reprezentarea STDM, timpii cresc semnificativ ajungând să depăşească două minute. Acest lucru era de aşteptat având în vedere că se construiesc *n(n-1)/2* arbori. Se poate observa că timpul de execuţie pentru aceste două distanţe este influenţat de numărul de documente şi mai puţin de numărul de clusteri rezultaţi. Se poate observa că în cazul setului S06, care este un set mic, algoritmul obţine timpi foarte buni chiar dacă sunt şase clusteri de calculat. În momentul de faţă, chiar dacă algoritmul HAC cu reprezentarea STDM şi distanţa NEWST obţine rezultatele cele mai bune din punct de vedere al acurateţei, din păcate necesită un timp de execuţie mai mare. O implementare optimizată ar reduce substanţial timpii de calcul şi l-ar face utilizabil şi în clusteringul online. Dezvoltări ulterioare ar trebui să acorde atenţie şi acestui aspect.

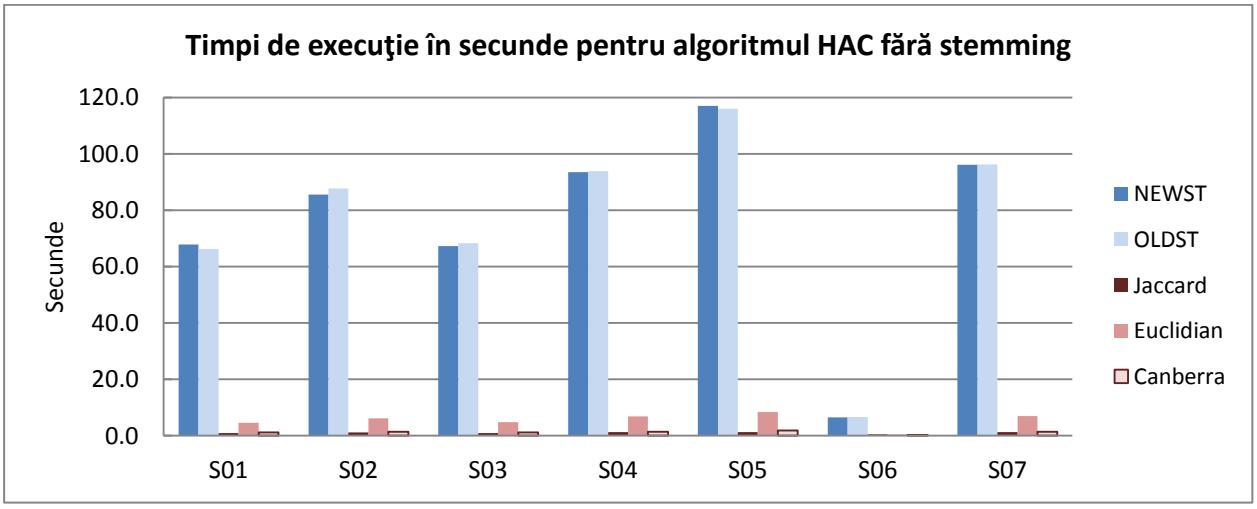


Fig. 4.39 - Timpii de execuţie pentru algoritmul HAC pe seturile de date S01-S07 fără stemming.

În Fig. 4.39 sunt prezentaţi timpii de execuţie pentru algoritmul HAC pe seturile de date S01-S07 la care nu s-a mai aplicat algoritmul de stemming. Se observă că timpii de execuţie cresc, dar nu semnificativ. Celelalte tendinţe, observate pe seturile de date unde am aplicat

algoritmul de stemming, se pot observa şi în cazul de faţă. Algoritmul cu reprezentarea STDM este şi în acest caz mai lent.

Pentru algoritmul *k*-Medoids, timpii de execuţie cresc semnificativ, ajungând în cazul setului de date S07 (atât cu stemming cât şi fără stemming) pentru reprezentarea STDM şi metricile NEWST şi OLDST la două ore. Pentru reprezentarea VSM am obţinut timpi mai mici dar şi aceştia se situează în jurul valorii de o oră. Acest fapt se poate explica prin implementarea PAM utilizată pentru algoritmul *k*-Medoids. Reamintim faptul că, în implementarea PAM se calculează toate distanţele faţă de fiecare medoid în parte. În implementarea *k*-Medoid clasică se calculează în primul pas apartenenţa documentelor la medoizii aleşi iar în al doilea pas alegerea medoizilor se face alegând cel mai apropiat document de centrul de greutate al clusterului format anterior, astfel ajungându-se mult mai repede la situaţia când medoizii nu se mai modifică. În implementarea PAM se ia iterativ fiecare submulţime de documente care poate forma un set complet de medoizi şi se evaluează pentru a vedea dacă dă rezultatele cele mai bune. La sfârşit se păstrează setul de documente care oferă cea mai bună evaluare.

Unele dintre rezultatele obţinute au fost prezentate şi în [Mora11].

Partea a III-a. Clasificare

Prediction is very difficult, especially if it's about the future.
Niels Bohr

5 Algoritmi de clasificare. Paradigma actuală

5.1 *Generalităţi*

Clasificarea datelor reprezintă procesul prin care unui tuplu i se atribuie una sau mai multe etichete dintr-o mulţime de etichete dată. Clasificarea este un proces în două etape. În prima etapă se va construi un clasificator care va descrie un set predefinit de date sau concepte. Această etapă se numeşte şi etapă de învăţare sau etapă de antrenare, în care un algoritm de clasificare construieşte clasificatorul învăţând pe baza unor tupluri de date (set de antrenament) şi etichetele asociate claselor acestora. Un tuplu X este reprezentat de un vector de atribute n-dimensional $X = (x_1, x_2, ..., x_n)$ care reprezintă măsurători efectuate în cadrul tuplului asupra a n atribute A_1, A_2, ..., A_n. Se presupune că fiecare tuplu aparţine unei clase, care la rândul ei, are o etichetă de clasă. Eticheta clasei este o valoare discretă şi nu este ordonată. În contextul clasificării, tuplurile se numesc exemple, instanţe sau obiecte. Deoarece eticheta clasei este dată pentru fiecare tuplu în etapa de antrenare, acest pas este un caz de învăţarea supervizată.

Astfel, prima etapă a procesului de clasificare poate fi văzută ca şi învăţarea unei funcţii de mapare $y=f(X)$ care poate prezice pentru un tuplu X dat eticheta y a unei clase. Această mapare este reprezentată sub forma unor reguli de clasificare, arbori de decizie sau formule matematice.

A doua etapă în procesul de clasificare o reprezintă utilizarea modelului elaborat în prima etapă, în clasificarea unui set de test. Pentru a măsura eficienţa clasificatorului trebuie efectuată o aproximare a acurateţei de clasificare a acestuia. Dacă am măsura acurateţea de clasificare pe baza setului de antrenament valorile obţinute ar fi foarte optimiste iar efectul de „over-fit” poate apărea (supraînvăţare – algoritmul învaţă unele anomalii existente în setul de antrenament care nu pot fi aplicate cazului general). Din acest motiv se creează un set de test care constă din tupluri şi clasele asociate acestora, altele decât cele prezente în setul de antrenament. Astfel, acurateţea clasificatorului este calculată ca procentaj al numărului de tupluri corect clasificate de către clasificator. Metrici pentru evaluarea acurateţei clasificatorilor au fost prezentate în secţiunea 2.6.

În secţiunile următoare voi prezenta o taxonomie posibilă pentru algoritmii de clasificare, prezentând pentru fiecare categorie clasificatorii reprezentativi pentru acea categorie.

5.2 *Algoritmi stohastici*

Cel mai reprezentativ clasificator stohastic este clasificatorul bayesian. El prezice cu o anumită probabilitate apartenenţa la o clasă dată, cum ar fi de exemplu, probabilitatea ca un document să facă parte dintr-o anumită clasă dată. Clasificarea bayesiană se bazează pe teorema

lui Bayes, şi este descrisă în secţiunea 5.2.1. Studiile care au comparat algoritmi de clasificare au identificat clasificatorul bayesian simplu cunoscut sub numele de clasificator bayesian naiv (Naïve Bayes Classifier) ca având, în practică, performanţe bune atunci când este aplicat la seturi de date mari, cum este cazul de faţă, unde există multe documente, fiecare reprezentat prin multe atribute.

Clasificatorul bayesian simplu pleacă de la premisa „naivă" că atributele pentru o anumită clasă sunt independente unele de altele. Această prezumţie se numeşte independenţă condiţională de clasă. În cadrul acestui clasificator se porneşte de la această prezumţie, pentru a simplifica calculele.

5.2.1 Clasificarea bayesiană

Fie *Y* o variabilă pentru o clasă (categorie) care poate lua valorile *{y1, y2, ..., y_m}*.

Fie *X* o instanţă a unui vector cu *n* atribute <x_1, x_2, ..., x_n> şi x_k o valoare posibilă pentru *X* şi x_{ij} o valoare posibilă pentru x_i. Pentru clasificarea de tip Bayes calculăm probabilităţile $P(Y = y_i \mid X = x_k)$ pentru $i = \overline{1,m}$. Asta ar însemna calcularea tuturor probabilităţilor pentru fiecare categorie, pentru fiecare instanţă posibilă din spaţiul de instanţe – ceea ce este foarte greu de calculat pentru un set rezonabil de date.

Practic, pentru a determina categoria lui x_k, trebuie să determinăm pentru fiecare y_i probabilitatea:

$$P(Y = y_i \mid X = x_k) = \frac{P(Y = y_i)P(X = x_k \mid Y = y_i)}{P(X = x_k)} \tag{5.1}$$

Probabilitatea $P(X = x_k)$ poate fi determinată, deoarece mulţimea categoriilor este completă şi disjunctă. Rezultă imediat relaţia de echilibru de mai jos:

$$\sum_{i=1}^{m} P(Y = y_i \mid X = x_k) = \sum_{i=1}^{m} \frac{P(Y = y_i)P(X = x_k \mid Y = y_i)}{P(X = x_k)} = 1 \tag{5.2}$$

aşadar:

$$P(X = x_k) = \sum_{i=1}^{m} P(Y = y_i)P(X = x_k \mid Y = y_i) \tag{5.3}$$

Probabilitatea $P(Y = y_i)$ poate fi uşor aproximată având în vedere faptul că dacă n_i exemple din *D* se regăsesc în y_i atunci $P(Y = y_i) = \frac{n_i}{|D|}$, unde D reprezintă mulţimea documentelor din setul de antrenament.

Probabilitatea $P(X = x_k \mid Y = y_i)$ trebuie estimată (deoarece există 2^n posibile instanţe pentru a calcula probabilitatea). De aceea, dacă presupunem că atributele unei instanţe sunt independente (condiţional independente), atunci:

$$P(X \mid Y) = P(X_1, X_2, \cdots X_n \mid Y) = \prod_{i=1}^{n} P(X_i \mid Y) \tag{5.4}$$

Astfel, trebuie să calculăm doar $P(X_i \mid Y)$ pentru fiecare pereche posibilă "valoare atribut"-"categorie".

Dacă *Y* şi toate X_i sunt binare, atunci trebuie să calculăm doar *2n* valori:

$P(X_i = true \mid Y = true)$ şi $P(X_i = true \mid Y = false)$ pentru fiecare X_i

$P(X_i = false) \mid Y) = 1 - P(X_i = true \mid Y)$

faţă de 2^n valori, dacă nu am presupune independenţa atributelor.

Practic, dacă setul de date D conţine n_k exemple din categoria y_k şi n_{ij_k} din aceste n_k exemple au a *j*-a valoare pentru atributul X_i pe x_{ij} atunci estimăm că:

$$P(X_i = x_{ij} \mid Y = y_k) = \frac{n_{ij_k}}{n_k} \tag{5.5}$$

Această estimare poate genera erori la seturi foarte mici de date, deoarece un atribut rar într-un set de antrenament face ca X_i să fie fals în setul de antrenament $\forall y_k \; P(X_i = true \mid Y = y_k) = 0$.

Dacă X_i=*true* într-un exemplu de test, atunci $\forall y_k \; P(X \mid Y = y_k) = 0$ şi $\forall y_k \; P(Y = y_k \mid X) = 0$

Pentru a evita acest lucru, se utilizează uniformizarea (normalizarea) lui Laplace. Această normalizare pleacă de la premisa că fiecare atribut are o probabilitate *p* observată într-un exemplu virtual de dimensiune *m*.

Astfel,

$$P(X_i = x_{ij} \mid Y = y_k) = \frac{n_{ijk} + mp}{n_k + m} \tag{5.6}$$

unde *p* este o constantă, de exemplu, pentru atribute binare p=0,5

Pentru clasificarea de tip text, clasificatorul Bayes generează pentru un document dintr-o anumită categorie c_i, reprezentat printr-un "bagaj de cuvinte", dintr-un vocabular V = {w_1, w_2,…w_m}, probabilitatea $P(w_j|c_i)$. Pentru normalizarea Laplace, se presupune existenţa unei distribuţii uniforme a tuturor cuvintelor (adică, ar fi echivalentul unui exemplu virtual în care fiecare cuvânt apare doar o singură dată). Aşadar se poate scrie:

$p = \frac{1}{|V|}$ şi $m = |V|$

5.2.2 Antrenarea clasificatorului Bayes

Probabilitatea ca un document *Y* să aparţină clasei X_i se calculează după cum urmează:

$$P(X_i \mid Y) \simeq P(X_i) \prod_{j=1}^{n} P(y_j \mid X_i) \tag{5.7}$$

unde $P(y_j \mid X_i)$ este probabilitatea condiţională ca termenul y_j să apară într-un document al clasei X_i. Interpretăm $P(y_j \mid X_i)$ ca fiind o măsură a contribuţiei lui y_j, în stabilirea faptului că X_i este clasa corectă.

$P(X_i)$ este probabilitatea apariţiei unui document în clasa X_i.

$\langle y_1, y_2, ..., y_j \rangle$ sunt termeni din documentul *Y* şi reprezintă o submulţime a vocabularului utilizat pentru clasificare, iar *n* reprezintă numărul termenilor.

În clasificarea documentelor text, scopul nostru este de a găsi cea mai bună clasă pentru respectivul document. În clasificarea Naive Bayes, cea mai bună clasă se stabileşte după metoda maximului aposteriori (MAP) şi o notăm cu c_{map}:

$$c_{map} = \arg\max_{1<i<m} \overline{P}(X_i \mid Y) = \arg\max_{1<i<m} \overline{P}(X_i) \prod_{j=1}^{n} \overline{P}(y_j \mid X_i) \tag{5.8}$$

Am utilizat notaţia $\overline{P}$ pentru *P*, deoarece nu cunoaştem exact valorile parametrilor $P(X_i)$ şi $P(y_j \mid X_i)$; aceştia pot fi însă estimaţi pe baza setului de antrenament.

În cazul de faţă, estimarea parametrilor $\overline{P}(X_i)$ şi $\overline{P}(y_j \mid X_i)$ se face după cum este descris în continuare. Pentru antrenarea clasificatorului, fie *V* vocabularul de cuvinte al documentelor conţinute în *D* şi pentru orice categorie $X_i \in X$ fie D_i un subset de documente din *D* din categoria X_i, atunci:

$$\overline{P}(X_i) = \frac{|D_i|}{|D|} \tag{5.9}$$

Fie Y_i concatenarea tuturor documentelor din D_i şi n_i numărul apariţiilor tuturor cuvintelor din Y_i. Pentru fiecare cuvânt $y_j \in V$ fie n_{ij} numărul apariţiilor cuvântului y_j în Y_i. Se poate scrie:

$$\overline{P}(y_j | X_i) = \frac{(n_{ij} + 1)}{(n_i + |V|)} \tag{5.10}$$

Parametrii luaţii în considerare pentru cazul nostru sunt:

Numărul total de atribute *n = 1309*, numărul total de documente din setul de antrenare *D= 4702*, numărul total de clase *m = 16* şi de asemenea, numărul total de documente existente în fiecare clasă D_1, ... D_{16}.

De exemplu, din setul de date de antrenament luăm clasa C18 (X_1=C18) care conţine 328 de documente. Algoritmul utilizează uniformizarea Laplace şi probabilitatea clasei C18 se calculează astfel:

$$\overline{P}(X_1) = \frac{Nr.\ documente + 1}{Nr.\ totaldocumente + Nr.\ clase} = \frac{328 + 1}{4702 + 16} = 0,069732938 \tag{5.11}$$

Pentru fiecare din cele 1309 atribute calculăm probabilităţile în raport cu fiecare clasă în parte.

Atribute	X_1 (C18)		X_2 -(C15)		...		X_{16} -(m11)	
	Nr. apariţii	$\overline{P}(y_j \mid X_1)$	Nr. apariţii	$\overline{P}(y_j \mid X_2)$			Nr. apariţii	$\overline{P}(y_j \mid X_{16})$
y_1	50		12				1	
y_2	12							
...								
y_{1309}	0							

Tabel 5.1 Calcularea probabilităţilor pentru fiecare trăsătură (1309)

Probabilitatea condiţională ca un atribut să fie într-o anumită clasă se calculează astfel:

$$\overline{P}(y_1 | X_1) = \frac{Nr.apariţii\ y_1 + 1}{Nr.total\ \ cuv.din X_1 + Nr.cuv.din Y} = \frac{50 + 1}{2570 + 1309} = 0,013147718$$

ş.a.m.d.

5.2.3 Testarea clasificatorului Bayes

Fie D_1 un document de test care conţine n_{D_1} termeni. În cazul nostru, pentru un document care trebuie clasificat, extragem toate atributele şi extragem din tabelul prezentat mai sus probabilităţile condiţionale.

În ecuaţia (5.8) din 5.2.2 trebuie calculate multe probabilităţi condiţionale şi datorită faptului că unele pot fi foarte mici, prin înmulţire se poate ajunge la *floating point underflow*. Pentru a evita acest fapt, ne folosim de binecunoscuta proprietate a logaritmului conform căreia:

$$\log(xy) = \log x + \log y \tag{5.12}$$

Deoarece funcţia logaritmică este monotonă, logaritmarea ecuaţiei (5.8) nu va modifica rezultatul alegerii clasei.

Astfel,

$$c_{map} = \arg\max_{1<i<m} \left[\log \overline{P}(X_i) + \sum_{j=1}^{n} \log \overline{P}(y_j | X_i) \right] \tag{5.13}$$

Ecuaţia (5.13) are o interpretare simplă: fiecare parametru condiţional $\log \overline{P}(y_j \mid X_i)$ este o pondere care arată cât de bun este un "indicator" y_j pentru X_i. Similar probabilitatea $\log \overline{P}(X_i)$ este o pondere care indică frecvenţa relativă a clasei c. Suma acestora este o măsură a evidenţei ca un document să aparţină unei clase. Ecuaţia (5.13) alege cea mai semnificativă clasă.

Astfel, vom obţine pentru fiecare clasă o valoare (care va fi negativă, deoarece logaritmăm o valoare subunitară) şi alegem clasa cu valoarea cea mai mare, adică cea mai aproape de valoarea 0. De exemplu, pentru fişierul nostru de test obţinem următoarele valori pentru fiecare clasă în parte:

```
Document: TestFile1578
Results:      c18(-366.59583445019484)
              c15(-277.3757195772894)
              c11(-393.7555314343488)
              c14(-376.69712554934097)
              c22(-405.2708070760941)
              gcat(-390.2472558128614)
              c33(-393.06805295995537)
              c31(-379.5501501924242)
              c13(-397.21992866728175)
              c17(-371.92813590774523)
              c12(-398.6571293708641)
              c21(-387.3768662210122)
              c23(-403.69793168505237)
              c41(-409.92000232701463)
              ecat(-390.18163176978925)
              m11(-395.70584000569795)
Correct class: 1 (c15), Predicted class: 1 (c15)
```

5.2.4 **Rezultate obţinute cu clasificatorului Bayes**

Am testat clasificatorul naiv Bayes (NB) pe un sistem cu procesor AMD X2 Turion la 1,7GHz şi 3 GB DRAM. Pentru validarea datelor de test, am utilizat metoda "*n-Fold Crossvalidation*". Am ales *n=10*, ceea ce înseamnă că setul de date existent se va împărţi în 10 submulţimi disjuncte, fiind utilizate 9 submulţimi pentru antrenament şi a 10-a submulţime neutilizată la antrenare să fie folosită la testare (evaluare). Această operaţie se execută de 10 ori, astfel încât toate submulţimile alese vor fi utilizate o singură dată în testarea clasificării. De asemenea, pentru a urmări şi acurateţea învăţării, am ales procente diferite din datele de intrare care vor fi folosite pentru antrenarea clasificatorului astfel: 0%, 1%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% şi 100%. În figura Fig.5.1 prezint rezultatele obţinute.

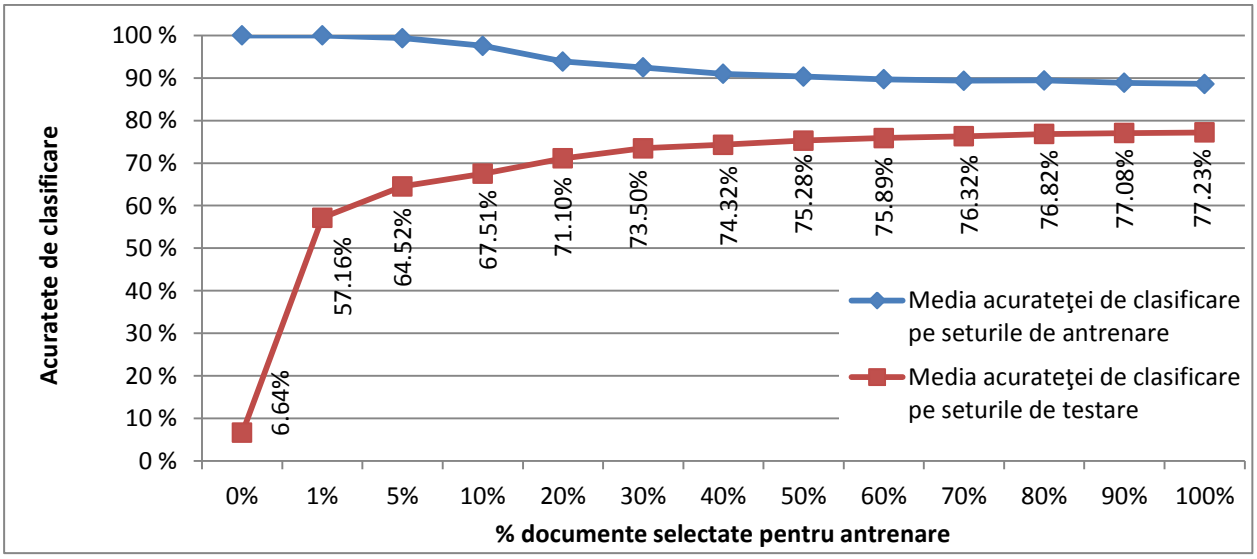


Fig. 5.1 Acurateţea clasificării şi curba de învăţare a clasificatorului Bayes

În graficul prezentat în Fig. 5.1 pe axa x sunt reprezentate numărul de documente utilizate succesiv pentru antrenare. Valorile corespund procentajelor: 0%, 1%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% şi 100%. Pe axa y sunt reprezentate valorile acurateţei la testare şi respectiv la antrenare.

În acest experiment s-au utilizat setul de antrenament A1 şi setul de testare T1 (7053 de documente din baza de date Reuters) împreună, urmând ca pentru împărţire în date de antrenare şi testare să se utilizeze metoda „*n-Fold Crossvalidation*". Ne propunem să utilizăm acest clasificator în clasificarea documentelor din setul de testare T2, adică testăm dacă poate să clasifice corect documente care nu au putut fi clasificate corect de clasificatorii de tip SVM (Support Vector Machine) într-o teză de doctorat anterioară [Mor08]. Seturile A1, T1, T2 au fost prezentate în secţiunea 2.7.1.

Astfel, se va utiliza setul de antrenare A1 şi pentru testare se va utiliza setul T2 care este format doar din 136 documente pe care clasificatorul Bayes din [IR] le împarte în 10 subseturi. În medie, rezultatele clasificării sunt mai bune decât rezultatele obţinute cu SVM (care a obţinut maxim 18% pe când Bayes a obţinut un maxim de 33.56%). Deoarece am folosit implementarea propusă în pachetul [IR] pentru algoritmul Bayes, (acesta îl voi numi în continuare BNN – Bayes Naive Nemodificat) fără a face alte modificări, şi deoarece acesta alege aleator subseturile pentru antrenare şi testare, nu putem specifica în cazul clasificatorului Bayes numărul de atribute alese din cele 1306 prezentate la intrare, şi nici metoda de selecţie folosită.

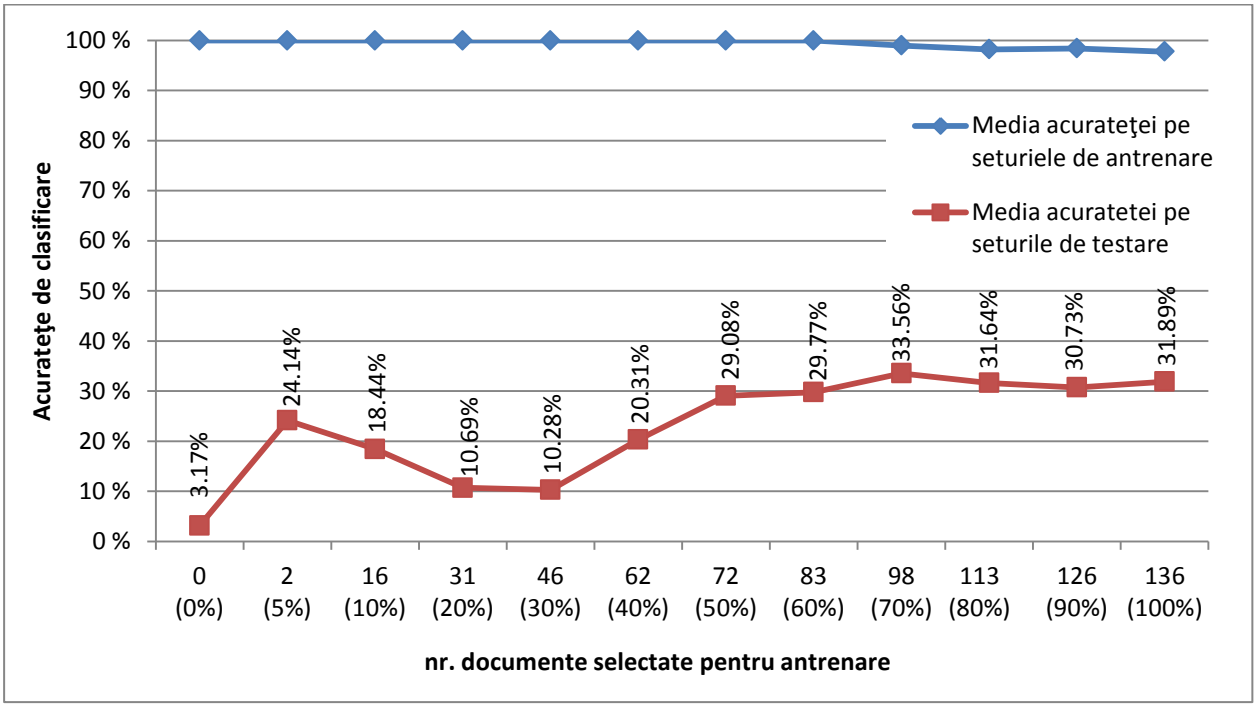


Fig. 5.2 Utilizarea clasificatorului Bayes în clasificarea unor documente care nu au fost clasificate corect de clasificatorul SVM (T2)

În Fig. 5.2 sunt prezentate rezultatele de clasificare ca şi medie pentru cele zece situaţii descrise. Testarea acestui clasificator s-a făcut pentru a vedea dacă poate furniza rezultate bune pe seturile de date prezentate A1 şi T1 respectiv A2 şi T2 şi pentru a fi ulterior inclus într-un metaclasificator care este prezentat în capitolul 6. Analizând rezultatele, am considerat că acest clasificator poate fi inclus în cercetările ulterioare, mai ales că, dacă îl comparăm cu rezultatele obţinute de clasificatorul SVM care sunt prezentate în secţiunea 6.1; acest clasificator obţine rezultate mai bune decât clasificatorul SVM pe setul T2.

5.3 *Algoritmi de învăţare bazaţi pe Backpropagation*

Cercetătorii din multe domenii ştiinţifice proiectează reţele neuronale artificiale pentru a rezolva o varietate de probleme cum ar fi: recunoaşterea de pattern-uri, predicţie, optimizare şi control etc. Abordări convenţionale au fost propuse pentru rezolvarea acestor probleme. De asemenea, pot fi aplicate cu succes în foarte multe domenii care nu sunt suficient de flexibile pentru utilizarea altor metode. În acestea, reţelele artificiale neuronale furnizează o alternativă viabilă [Hayk94].

În lungul curs al evoluţiei, creierul uman a dobândit multe trăsături care nu se regăsesc în modelul von Neumann sau se regăsesc mult prea timid în calculatoarele paralele moderne [Vint00]. Unele dintre aceste trăsături ar fi (conform [Jain96]).

- paralelism masiv
- reprezentare şi procesare distribuită
- abilităţi de învăţare
- abilităţi de generalizare
- adaptabilitate
- procesarea informaţiei pe bază de context
- toleranţă la erori
- consum redus de energie

Calculatoarele numerice actuale domină net omul în ceea ce priveşte prelucrările numerice. Totuşi, omul poate fără efort să rezolve unele probleme complexe de percepţie şi recunoaştere a formelor cu o viteză incomparabil superioară celor mai performante calculatoare. Aceasta diferenţă provine din arhitectura complet diferită faţă de cea a maşinii von Neuman.

Inspirate din reţelele neuronale biologice, **R**eţelele **N**euronale **A**rtificiale (**RNA**) sunt sisteme de calcul cu paralelism masiv constituite dintr-un număr mare de elemente de procesare simple - numite **neuroni** - cu multe interconexiuni între ele. Modelele propuse pentru RNA respectă anumite principii de organizare presupuse ca fiind folosite în creierul uman.

În evoluţia RNA există trei perioade distincte. Prima are loc în anii '40 prin munca de pionierat a lui McCulloch şi Pitts. A doua perioadă în anii '60 are la bază teorema lui Rosenblatt de convergenţă a perceptronului şi demonstrarea de către Minsk şi Papert a limitărilor perceptronului simplu (ex. imposibilitatea învăţării unor seturi de date neseparabile liniar). Începând doar cu anii '80, domeniul RNA şi-a redobândit interesul. Acesta are la bază introducerea noţiunii de energie în reţeaua Hopfield în 1982 şi, main ales, găsirea algoritmului de învăţare cu retropropagarea erorii ("Backpropagation") pentru reţele cu propagare înainte ("feedforward") multistrat, propus iniţial de Paul Werbos în 1974 (U. Berkeley) şi redescoperit şi popularizat de Rumelhart et al în 1986 [Maca03].

5.3.1 Modelul neuronului artificial

Modelul neuronului artificial [Wass89], [Kung93] are la bază modelul propus de McCulloch şi Pitts şi generalizat apoi în multe feluri. Prezentăm în continuare cea mai întâlnită variantă.

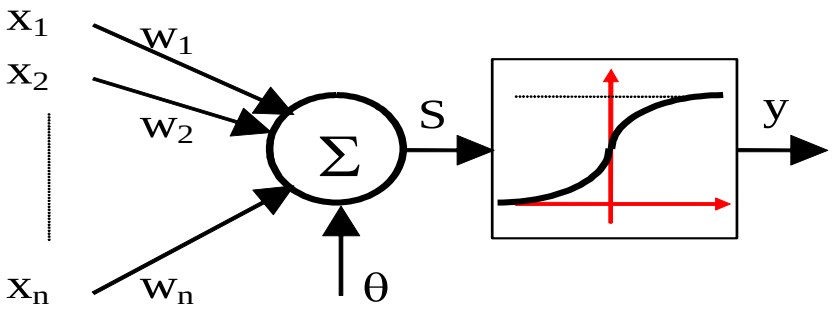


Fig. 5.3 Modelul neuronului artificial

Acest neuron artificial calculează suma ponderată a *n* semnale de intrare, adaugă o valoare numită prag şi apoi, aplică acestei valori o funcţie de activare generând ca ieşire o valoare cuprinsă în intervalul (0,1)

$$S = \sum_{i=1}^{n} x_i w_i + \theta \qquad y = f(S) \tag{5.14}$$

În aceste relaţii, x_i reprezintă semnalul intrării i şi w_i sinapsa (ponderea, tăria sinaptică) asociată acestei intrări. Termenul θ reprezintă o valoare de prag (de offset, bias), care deplasează (transpune) ieşirea S a neuronului. Ieşirii S i se aplică o funcţie de activare f, care va transpune (normaliza) ieşirea neuronului în domeniul de valori dorit.

Pentru funcţia de activare cele mai întâlnite funcţii sunt cele prezentate în Fig. 5.4:

a - funcţia de activare treaptă, $step(x) = \begin{cases} 1, if\ x \geq 0 \\ 0, if\ x < 0 \end{cases}$

b - funcţia de activare semn, $sign(x) = \begin{cases} +1, if\ x \geq 0 \\ -1, if\ x < 0 \end{cases}$

c - funcţia de activare sigmoidală, $sigmoid(x) = \frac{1}{1+e^{-x}}$

d - funcţia de activare gaussiană, $Gauss(x) = e^{-\frac{(m-x)^2}{2\sigma^2}}$

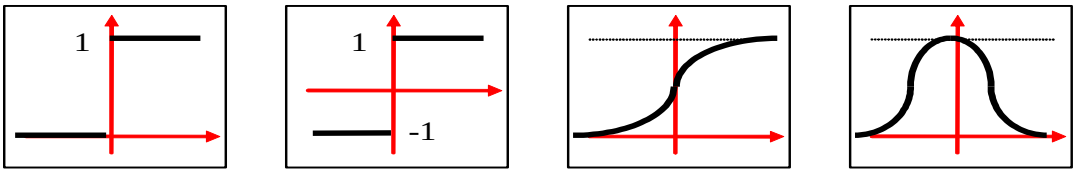


Fig. 5.4 Funcţiile de activare cele mai frecvent întâlnite

Modelul neuronului prezentat anterior, având funcţia de activare treaptă, este modelul iniţial propus de McCulloch şi Pitts în 1943. Cel mai popular model al neuronului a devenit însă cel cu funcţia de activare sigmoidală care este strict monoton crescătoare, mărginită şi derivabilă:

$$f(x) = \frac{1}{1+e^{-\alpha x}} \tag{5.15}$$

unde α este un factor de scară luând valori strict pozitive. Pentru α tinzând la infinit funcţia sigmoidă devine funcţia treaptă.

5.3.2 Arhitectura reţelelor neuronale

Există o varietate de tipuri de structuri de reţele, fiecare dintre acestea rezultând din diferite posibilităţi de calcul şi în funcţie de problema care trebuie rezolvată. O reţea neuronală poate fi privită ca un graf orientat ponderat, în care neuronii sunt nodurile iar arcele orientate (cu ponderile asociate) sunt legăturile între neuroni Din punct de vedere al construcţiei, reţelele neuronale se împart în două categorii principale: reţelele ***feed-forward*** şi reţelele ***recurente*****.**

- În reţelele **feed-forward** (cu propagare înainte), legăturile dintre neuroni sunt unidirecţionale şi nu există bucle de reacţie (legături de la un neuron de pe un strat superior la un neuron de pe un strat inferior). În reţelele feed-forward, legăturile pot pleca de la topologii arbitrare, neexistând nici o legătură spre stratul anterior sau legături care sar peste un anumit strat. În aceste tipuri de reţele, toţi neuronii de pe un anumit strat sunt actualizaţi sincron, la o anumită perioadă de timp. Aceste reţele sunt reţele statice, ele neavând un comportament dinamic propriu-zis, ieşirea reţelei depinzând doar de valoarea curentă a intrării nu şi de valorile anterioare ale intrării.
- Reţelele **recurente** sunt reţele în care pot exista legături înapoi, un neuron de pe un strat superior având legături cu neuroni de pe nivelurile inferioare. De asemenea, în reţelele recurente pot exista legături care sar peste anumite straturi. În acest caz, reţeaua reprezintă un graf orientat complet şi are un comportament dinamic propriu-zis. În această categorie se încadrează reţelele competitive, hărţile topografice ale lui Kohonen, reţeaua Hopfield, reţeaua recurentă Elman, maşina Boltzmann şi modelele ART.

5.3.3 Învăţarea reţelelor neuronale

Capacitatea de învăţare este o trăsătură fundamentală a inteligenţei. O definiţie a învăţării este dificil de dat; dar vom afirma că, în contextul RNA, învăţarea constă în modificarea reţelei pentru a-şi adapta comportamentul la necesităţile rezolvării unei probleme. În general, sunt modificaţi coeficienţii de conectivitate sinaptică, uneori modificându-se şi numărul de unităţi şi / sau configuraţia reţelei. Învăţarea este importantă prin posibilitatea de adaptare la un mediu în schimbare (sistemele de IA clasice sunt rigide) fiind utilă şi în cazul în care mediul nu evoluează dar pentru care nu dispunem de un model.

Principalul avantaj al RNA în raport cu sistemele expert clasice este acela că, în loc de a folosi un **set de reguli** date de un expert uman, are loc o **învăţare prin exemple.**

Din punct de vedere al organizării datelor de intrare, există două categorii de învăţare [Bish95]:

- învăţarea ***nesupervizată*** în care se prezintă reţelei doar datele de intrare fără a se specifica şi ieşirea dorită pentru acestea, astfel că reţeaua nu are nici o informaţie despre prezenţa sau valoarea erorii. În acest caz, reţeaua este lăsată să evolueze liber, urmând ca la sfârşit să constatăm rezultatul învăţării. Reţeaua analizează corelaţiile între datele de intrare şi organizează datele în categorii pe baza acestor corelaţii.
- învăţarea ***supervizată*** în care mulţimea de exemple de antrenament este organizată sub forma de perechi intrare-ieşire, specificând reţelei la fiecare pas, care trebuie să fie ieşirea corectă, urmând ca reţeaua să generalizeze datele de intrare. Ponderile sunt modificate astfel încât reţeaua să producă ieşiri cât mai apropiate de răspunsul corect. Învăţarea prin întărire ("reinforcement learning") este o variantă a învăţării supervizate în care se furnizează reţelei doar o informaţie despre prezenţa erorii nu şi a valorii propriu zise a ei.

Fiecare tip de reţea îşi modifică ponderile în funcţie de anumite reguli de învăţare, care depind atât de tipul datelor de intrare cât şi de modul de construcţie al reţelei. Din punctul acesta de vedere există patru tipuri consacrate de reguli de învăţare:

- învăţare prin corecţia erorii;
- regula lui Boltzmann;
- regula lui Hebb;
- învăţarea competitivă.

5.3.4 Reguli de învăţare prin corecţie a erorii ("error-correction rules")

În învăţarea supervizată, reţeaua dispune de ieşirea dorită pentru fiecare vector de intrare. În timpul învăţării, ieşirea reţelei nu este de obicei egală cu această valoare dorită. Principiul corecţiei erorii foloseşte semnalul de eroare pentru modificarea ponderilor în scopul minimizării erorii. Relaţia cea mai generală de modificare a unui coeficient sinaptic *w* conform acestei reguli este (evoluţie în sensul invers gradientului erorii; conduce deci spre descreşterea erorii):

$$\Delta w = -\eta \frac{\partial E}{\partial w} \tag{5.16}$$

unde *E* este eroarea globală (dependentă de *w*) şi η este viteza de învăţare. Această relaţie stă la baza învăţării în reţelele feed-forward multistrat.

Ideea de bază este de a utiliza *panta gradientului* pentru a căuta în spaţiul ipotezelor de posibili vectori de ponderi pentru a găsi acele ponderi care aproximează cel mai bine exemplele de antrenament. Această regulă este importantă, deoarece furnizează bazele algoritmului Backpropagation, care este utilizat în cazul reţelelor cu mai multe unităţi interconectate.

Panta gradientului caută să determine vectorul pondere care minimizează eroarea pornind de la un vector pondere iniţial arbitrar, care este apoi modificat repetat în paşi mici. La fiecare pas, vectorul pondere este modificat în direcţia în care produce o pantă descendentă de-a lungul suprafeţei erorii. Acest proces continuă până când eroarea minimă globală este atinsă.

Regula de învăţare a perceptronului simplu propusă de Rosenblatt în 1962 foloseşte o variantă simplificată a regulii de minimizare a erorii. În acest caz avem:

$$\Delta w = \eta(d - y)x \tag{5.17}$$

în care *w* şi *x* sunt vectorul ponderilor şi vectorul de intrare, d este ieşirea dorită şi y ieşirea reală.

Regula de învăţare pentru reţeaua Adaline (strat de perceptroni cu ieşirea liniară) cunoscută şi ca Regula Widrow-Hoff are şi ea la bază minimizarea erorii

$$\Delta w_{ij} = \eta\, x_j (T_i - y_i) \tag{5.18}$$

unde w_{ij} este ponderea legăturii ieşirii *i* cu intrarea *j*, *x* vectorul de intrare, *T* vectorul dorit la ieşire şi *y* vectorul ieşirii reale. Se poate demonstra, că regula anterioară este o particularizare a regulii gradientului în cazul definirii erorii conform

$$E = \frac{1}{2}\sum_{i=1}^{N}(T_i - y_i)^2 \tag{5.19}$$

5.3.4.1 Perceptronul [Vint07]

Una din cele mai simple reţele neurale este perceptronul (o singură celulă). este prezentată în Fig. 5.5.

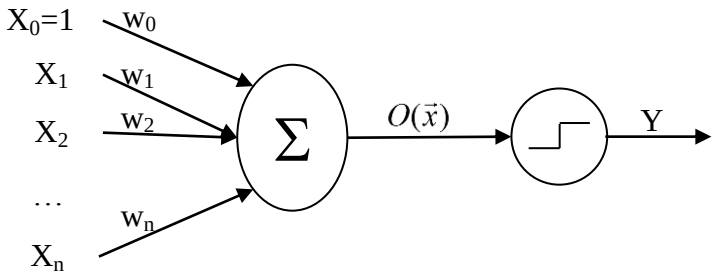


Fig. 5.5 Calcularea ieşirii perceptronului

$\overline{X} = \{x_0, x_1, ..., x_n\}$, reprezintă vectorul cu valorile de intrare,

$\overline{W} = \{w_0, w_1, ..., w_n\}$, reprezintă vectorul cu valorile ponderilor

$O(\overline{X}) = \overline{W} \cdot \overline{X} = w_0 + \sum_{k=1}^{n} w_k \cdot x_k$, reprezintă ieşirea

$Y = \begin{cases} +1 & dacă\ O(\overline{X}) > 0 \\ -1 & dacă\ O(\overline{X}) \le 0 \end{cases}$, reprezintă semnul la ieşire.

Perceptronul poate fi considerat a fi reprezentarea unei suprafeţe de decizie într-un hiperplan în spaţiul *n*-dimensional al intrărilor. Ecuaţia acestui hiperplan de decizie este

$\overline{W} \cdot \overline{X} = \mathbf{0}$

Astfel, perceptronul poate fi utilizat ca fi un clasificator binar sau un predictor (Taken = +1 sau Not_Taken = -1). Bineînţeles, acest perceptron poate clasifica corect doar un set de exemple ($\overline{X}$) care sunt linear separabile. De exemplu, funcţia logică XOR nu poate fi reprezentată de un singur perceptron.

Problema principală este cum să formulăm o regulă de învăţare pentru un perceptron simplu, pentru a învăţa corect un set de vectori de antrenament pe care îl vom nota cu *D*. Dacă considerăm pentru fiecare exemplu (vector de antrenament) o regulă de învăţare supervizată $d \in D$ este necesar să cunoaştem ieşirea corespunzătoare denumită t_d.

Dacă $O_d = w_0 + \sum_{k=1}^{n} w_k x_{dk}$ este ieşirea reală o măsură comună a erorii E este:

$$E(\overline{w}) = \frac{1}{2} \sum_{d \in D} (t_d - O_d)^2 \qquad (5.20)$$

Dată fiind formula pentru $E(\overline{w})$ suprafaţa trebuie să fie întotdeauna un paraboloid cu un singur minim global. Bineînţeles, în particular $\overline{w}$ care dă minimul clasifică în cea mai bună măsură exemplul X_{dk}, *k=0,1,..,n*. Gradientul $E(\overline{w})$ se notează

$$\nabla E(\overline{w}) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, ..., \frac{\partial E}{\partial w_n} \right] = \sum_{k=0}^{n} \frac{\partial E}{\partial w_k} \overline{i}_k \qquad (5.21)$$

unde $\overline{i}_k$ sunt vectorii unitate ortogonali in spaţiul n dimensional. Se ştie că gradientul specifică direcţia în care se produce cea mai rapidă micşorare a lui E. În acest caz, regula de învăţare ar fi:

$\overline{W} \leftarrow \overline{W} + \Delta\overline{W}$, unde $\Delta\overline{W} = -\alpha \nabla E(\overline{W})$, α = rata de învăţare (a un număr real mic pozitiv). Aceasta este echivalent cu :

$$w_k \leftarrow w_k - \alpha \frac{\partial E}{\partial w_k}, (\forall) k = 0,1,..., n$$

Dar: $\frac{\partial E}{\partial w_k} = \frac{\partial}{\partial w_k}\left(\frac{1}{2}\sum_{d\in D}(t_d - O_d)^2\right) = \sum_{d\in D}(t_d - O_d)\frac{\partial(t_d - \overline{W}\cdot\overline{X})}{\partial w_k} = -\sum_{d\in D} x_{dk}\cdot(t_d - O_d)$

În final, regula de învăţare supervizată este:

$$w_k \leftarrow w_k + \alpha\sum_{d\in D}(t_d - O_d)x_{dk}, (\forall)k = 0,1,...,n \tag{5.22}$$

Această regulă se numeşte regula de gradient descendent sau regula delta. Implementarea algoritmului este descrisă mai jos.

Initialize each W_k to random values $\in\left[-\frac{2}{n},\frac{2}{n}\right]$

Until $E(\vec{w}) < T(threshold)$, DO:

Initialize each $\Delta W_k = 0$

For each pair (x_d, t_d), from training examples, DO:

Compute O_d

For each W_k, DO:

$\Delta w_k \leftarrow \Delta w_k + \alpha(t_d - O_d)x_{dk}$

For each w_k, DO:

$w_k = w_k + \Delta w_k$

O idee alternativă este găsirea aproximării gradientului descendent prin actualizarea ponderilor incremental, urmat de calcularea erorii pentru fiecare exemplu de antrenament. O modalitate de a implementa stohastic acest gradient descendent este să considerăm eroarea distinctă:

$$E_d(\overline{w}) = \frac{1}{2}\left(t_d - O_d\right)^2 \tag{5.23}$$

Utilizând aleator exemplele X_d obţinem o aproximare rezonabilă a micşorării gradientului în comparaţie cu eroarea globală $E(\overline{w})$

Regula stohastică pentru gradientul descendent este:

Initialize each w_k randomly to $\left[-\frac{2}{n},+\frac{2}{n}\right]$

Until the termination condition is met $(E_d(\overline{w}) < T \text{ or } |O_d| > T, etc.)$, DO:

For each (x_d, t_d), DO:

Compute O_d

For each w_k, do:

$w_k \leftarrow w_k + \alpha(t_d - O_d)x_{dk}$

Regula standard a gradientului descendent este consumatoare de timp datorită însumării multiplelor exemple, dar se utilizează adesea cu o rată de învăţare per exemplu mai mare decât rata de învăţare per exemplu de la regula stohastică cu gradientul incremental descendent. Dacă $\mathbf{E}(\overline{W})$ are multiple minime locale, gradientul stohastic poate evita în unele cazuri oprirea în aceste minime locale deoarece utilizează diverse $\nabla\mathbf{E}_d(\overline{W})$ în găsirea minimului.

Dacă considerăm ieşirea perceptronului $O(\overline{X}) = \text{sgn}(\overline{W}\cdot\overline{X})$ în locul $O(\overline{X}) = \overline{W}\cdot\overline{X}$, atunci această regulă se denumeşte regula de antrenare a perceptronului:

$$w_{\mathbf{k}} \leftarrow w_{\mathbf{k}} + \alpha(t-o)x_{\mathbf{k}}, \quad (\forall)k = 0,1,...,n \tag{5.24}$$

Dacă exemplul de antrenament este corect clasificat, (t-o=0), nu se actualizează nici o pondere. Presupunem acum o=-1 şi t = +1. În acest caz, toate ponderile w_k cu valorile pozitive x_k vor fi incrementate, iar celelalte ponderi w_k vor fi decrementate. Similar pentru o = +1 and t = -1

toate ponderile w_k cu valori x_k negative vor fi incrementate iar restul ponderilor w_k vor fi decrementate. Ca şi o regulă intuitivă, dacă $\operatorname{sgn} t = \operatorname{sgn} x_k$, atunci w_k va fi incrementat iar altfel w_k va fi decrementat.

5.3.5 **Regula de învăţare Boltzmann**

Maşinile Boltzmann sunt reţele recurente simetrice ($w_{ij} = w_{ji}$), constând din unităţi binare. Un subset al neuronilor reţelei sunt **vizibili** şi interacţionează cu mediul (cei de intrare şi de la ieşire) iar ceilalţi sunt **ascunşi**. Starea unei ieşiri este 0 sau 1 cu o anumită probabilitate:

$$p_i = \frac{1}{1+e^{-\frac{\tilde{x}_i}{T}}} \quad \text{unde} \quad \tilde{x}_i = \sum_{j \neq i} w_{ij} x_j - \theta_i \tag{5.25}$$

x fiind stările celorlalte unităţi, w_{ij} coeficienţii sinaptici, θ valori de prag iar T "temperatura". Alegerea noii stări se face în concordanţă cu probabilitatea p_i.

Ponderile coeficienţilor sinaptici se modifică apoi conform regulii de învăţare Boltzmann

$$\Delta w_{ij} = \eta \left(< ij >^{+} - < ij >^{-} \right) \tag{5.26}$$

unde η este rata de învăţare, estimându-se: $<ij>^{+}$ probabilitatea ca unităţile i şi j să fie active simultan când unităţile vizibile sunt forţate la valorile dorite şi $<ij>^{-}$ probabilitatea ca unităţile i şi j să fie active simultan când unităţile de ieşire sunt libere.

Regula de învăţare Boltzmann poate fi privită ca un caz special de învăţare prin reducerea erorii, în care eroarea nu este măsurată direct, ci ca şi diferenţă a corelaţiei între ieşiri în cele două moduri. Se încearcă astfel, ca reţeaua să evolueze la fel atât în mod forţat cât şi liber.

5.3.6 **Regula de învăţare Hebb**

Una dintre cele mai vechi reguli de învăţare este postulatul lui Hebb apărut în 1949 în „Organization of behavior” [Hebb49]. Aceasta are la baza observaţia neurobiologică:

> *Dacă ambii neuroni legaţi printr-o sinapsă sunt activi simultan, coeficientul sinaptic al acestei legături creşte.*

Matematic, regula lui Hebb poate fi descrisă astfel:

$$\Delta w_{ij} = \eta y_i x_i \tag{5.27}$$

unde x_i şi y_j sunt activităţile celor doi neuroni i şi j conectaţi prin sinapsa w_{ij} şi η este rata de învăţare.

Regula lui Hebb este plauzibilă biologic şi prezintă avantajul că învăţarea se face în mod local, modificarea ponderii unei sinapse depinzând doar de neuronii alăturaţi, ceea ce facilitează implementarea în circuite VLSI, atât sub formă digitală cât şi analogică.

5.3.7 **Regula de învăţare competitivă**

Spre deosebire de învăţarea bazată pe regula lui Hebb (în care mai mulţi neuroni pot fi activi simultan), în cazul învăţării competitive, între unităţile de ieşire are loc o competiţie pentru activare. În final, o singură unitate va fi activă la un moment dat. Acest fenomen este cunoscut ca „învingătorul ia totul” („winner takes all”).

Cea mai simplă reţea competitiva constă dintr-un singur strat de neuroni, fiecare conectat la vectorul de intrare. Fiecare neuron îşi stabileşte activarea, după care, în urma unui proces de competiţie se determină un singur neuron i^* câştigător.

Regula de modificare a ponderilor sinaptice este:

$$\Delta w_{ij} = \begin{cases} \eta(x_j - w_{i^*j}) & i = i^* \\ 0 & i \neq i^* \end{cases} \quad (5.28)$$

Se observă că se modifică numai vectorul ponderilor legăturilor sinaptice al neuronului câştigător. Efectul aplicării acestei reguli de învăţare este acela că vectorul *w* (memorat) se apropie de vectorul de intrare. Conform regulii de învăţare competitive, reţeaua va termina învăţarea (actualizarea ponderilor) doar în momentul în care rata de învăţare η este 0. Un pattern de intrare particular poate activa diferite unităţi de ieşire la iteraţii diferite pe durata învăţării. Aceasta duce la un comportament stabil al sistemului de învăţare. Un sistem este stabil dacă nici un pattern din datele de antrenament nu-şi schimbă categoria după un număr finit de iteraţii de învăţare. O metodă de a obţine un sistem stabil este de a forţa rata de învăţare să descrească gradual pe parcursul procesului de învăţare până când devine 0. Această îngheţare artificială a învăţării cauzează pierderea caracteristicii numită *adaptabilitate*, care reprezintă abilitatea unei reţele de a se adapta la noi date. Aceasta este cunoscută ca dilema „stabilitate-adaptabilitate" a lui Grossberg.

5.3.7.1 Metoda Backpropagation

5.3.7.2 Perceptroni multistrat cu funcţie de activare neliniară

Perceptronii cu un singur strat de parametri modificabili nu pot clasifica decât mulţimi liniar separabile de vectori de intrare. Acest lucru a fost demonstrat încă din 1969 de către Minsky şi Papert şi a îndepărtat interesul cercetătorilor de reţelele neuronale. Se ştia de atunci că pentru perceptronii multistrat aceste probleme nu apar dar nu era clar cum să se modifice ponderile straturilor ascunse. Problema contribuţiei unităţilor interne a fost rezolvată şi diseminată pe scară largă abia în 1986 ducând la renaşterea interesului pentru reţelele neuronale.

5.3.7.3 Perceptronul multistrat

Cea mai populară categorie de reţele feed-forward multistrat este perceptronul multistrat în care fiecare unitate de calcul utilizează funcţia de prag sau funcţia sigmoidă.

Fie

$$x_i(k+1) = f\left(\sum_{j=1}^{N_k} w_{ij}(k) \cdot x_j(k)\right) \quad (5.29)$$

activarea unităţii *i* din stratul *k+1*, N_k numărul de unităţi din stratul *k* şi *f* este funcţia de activare.

Notăm $u_i(k+1)$ argumentul funcţiei *f* deci:

$$u_i(k+1) = \sum_{j=1}^{N_k} w_{ij}(k) \cdot x_j(k) \quad (5.30)$$

Pentru fiecare vector de ieşire eroarea globală este dată de relaţia:

$$E = \frac{1}{2}\sum_{i=1}^{N}\left(T_i - x_i\right)^2 \tag{5.31}$$

x_i fiind activităţile stratului de ieşire şi T_i valorile dorite la ieşire.
Numim eroare a unei unităţi:

- pentru ultimul strat

$$err_i = -(T_i - x_i)\cdot f'(u_i) \tag{5.32}$$

- pentru un strat ascuns k

$$err_i(k) = f'\left[u_i(k)\right]\cdot\sum_{j=1}^{N_{k+1}}\left[err_j(k+1)\cdot w_{ij}(k)\right] \tag{5.33}$$

Cu aceste notaţii se poate demonstra că modificarea parametrilor în direcţia gradientului

$$\Delta w_{ij}(k) = -\eta\cdot\frac{\partial E}{\partial w_{ij}(k)} \tag{5.34}$$

devine, pentru toate straturile,

$$\Delta w_{ij}(k) = -\eta\cdot x_j(k)\cdot err_i(k+1) \tag{5.35}$$

Întotdeauna trebuie învăţate asocieri între mai mulţi vectori de intrare şi de ieşire. În acest caz, funcţia de eroare totală este suma funcţiilor de eroare corespunzătoare perechilor individuale intrare/ieşire. Aceasta eroare poate fi minimizată în două moduri:

1 *off-line* - se determină, pentru fiecare pereche intrare/ieşire, modificarea ce trebuie adusă coeficienţilor sinaptici. Aceste modificări se sumează şi se aplică numai după ce au fost prezentate toate perechile intrare/ieşire. Algoritmul realizează o optimizare deterministă după gradient a erorii totale.

2 *on-line* - modificarea coeficienţilor calculată pentru o pereche intrare/ieşire este aplicată imediat după prezentarea acestei perechi. Algoritmul realizează o optimizare după gradient pentru eroarea totală. Prezintă, în raport cu precedentul, avantajul că este în general mai rapid şi poate părăsi unele minime locale ale funcţiei de eroare totală.

În ceea ce priveşte parametrul η - mărimea pasului în direcţia gradientului - acesta determină viteza de convergenţă spre un minim al erorii E. În practică se începe cu un η relativ mare iar apoi, pe măsură ce reţeaua învaţă, această valoare se reduce treptat (analogie cu algoritmul Simulated Annealing).

5.3.8 **Algoritmul de învăţare Backpropagation**

Algoritmul Backpropagation învaţă ponderile pentru o reţea pe mai multe niveluri, dându-se o reţea cu o mulţime fixă de unităţi şi de interconexiuni. Utilizează panta gradientului pentru a încerca să minimizeze eroarea dintre valoarea ieşirii reţelei şi valoarea ţintă pentru acele ieşiri.

Problema învăţării în Backpropagation este de a căuta în spaţiul mare al ipotezelor, definit de toate valorile posibile ale ponderilor pentru toate unităţile din reţea. Algoritmul Backpropagation este prezentat în continuare. Algoritmul descris aici se aplică reţelelor feed-forward care conţin două nivele de unităţi, cu funcţia de activare sigmoidă, fiecare unitate de pe un nivel fiind conectată cu toate unităţile de pe nivelul anterior. Aceasta este o versiune a algoritmului backpropagation care calculează, incremental sau stohastic, panta gradientului.

Backpropagation (*exemple_antrenament*, η, n_{in}, n_{out}, n_{hidden})

Fiecare exemplu de antrenament este o pereche de forma $\langle \vec{x}, \vec{t} \rangle$, *unde* $\vec{x}$ *reprezintă valorile vectorului de intrare şi* $\vec{t}$ *reprezintă valorile ţintă ale vectorului de ieşire.*
η reprezintă rata de învăţare care este o valoare (0,1]
n_{in} *numărul de neuroni de pe stratul de intrare*
n_{hidden} *numărul de neuroni de pe stratul ascuns*
n_{out} *numărul de neuroni de pe stratul de ieşire*
intrarea pentru unitatea i de la unitatea j este notată cu x_{ji} *şi ponderea de la unitatea i la unitatea j este notată* w_{ji}

- Se creează o reţea feed-forward cu n_{in} intrări, n_{hidden} unităţi ascunse şi n_{out} unităţi de ieşire
- Se iniţializează toate ponderile din reţea cu valori aleatoare mici (de exemplu $\in \left[-\frac{2}{n_{in}}, \frac{2}{n_{in}} \right]$)[Vint07]
- Până când condiţia de terminare nu este îndeplinită execută (exemplu, eroarea....)
  - Pentru fiecare $\langle \vec{x}, \vec{t} \rangle$ *din exemple_antrenament* execută
    - Propagă semnalul forward prin reţea:
      1. se introduce instanţa $\vec{x}$ în reţea şi se calculează ieşirea *y* pentru fiecare neuron din reţea
    - Propagă eroarea înapoi prin reţea - backward
      2. pentru fiecare neuron de ieşire din reţea se calculează eroarea conform formulei (5.32)
      3. pentru fiecare neuron de pe stratul ascuns se calculează eroarea conform formulei (5.33)
      4. calculează $\Delta w_{ji} = \eta \cdot err_i \cdot x_{ji}$ pentru nivelele anterioare prin propagarea backward a erorii
      5. se actualizează ponderile reţelei $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$

Algoritmul este descris aici pentru o reţea feed-forward conţinând două straturi de unităţi cu funcţia sigmoidă de activare în cazul general (Fig. 5.6). Fiecare unitate de pe fiecare strat este conectată cu toate unităţile de pe stratul precedent. Unităţile de pe stratul de intrare sunt considerate unităţi repetoare care prezintă la ieşire valoarea primită la intrare. De asemenea, sunt prezentate formulele de calcul pentru această reţea, atât pentru pasul forward cât şi pentru pasul backward. Pentru pasul backward s-a luat în considerare formula de calculul a erorii prezentată în ecuaţia (5.39).

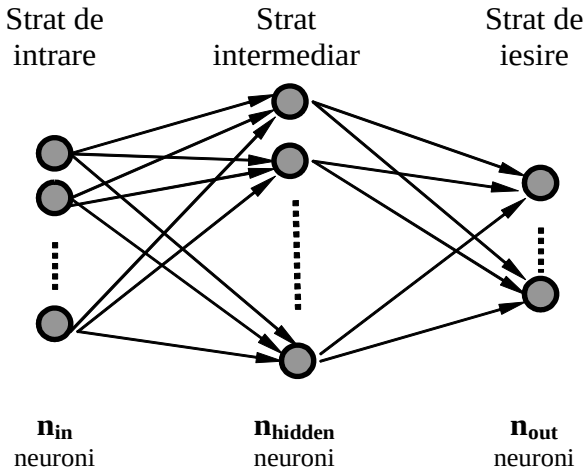


Fig. 5.6 - Arhitectura reţelei

Pentru acest caz formulele generale date de regula backpropagation devin, cu notaţiile următoare [Brea06]:

$w_{12}[h][i]$	ponderea legăturii neuronului **h** ("hidden") din stratul intermediar (2) cu neuronul **i** din stratul de intrare ("input") (1)
$\theta 2[h]$	valoarea de prag a neuronului h din stratul intermediar (2)
$w23[o][h]$	ponderea legăturii neuronului o ("output") din stratul de ieşire (3) cu neuronul h din stratul intermediar (2)
$\theta 3[o]$	valoarea de prag a neuronului o din stratul de ieşire (3)
$Out1[i]$	valoarea ieşirii neuronului i din stratul de intrare
$Out2[h]$	valoarea ieşirii neuronului h din stratul de intermediar
$Out3[o]$	valoarea ieşirii neuronului o din stratul de ieşire
$Scop[i]$	valoarea dorită la ieşire
$F(.)$	funcţia de activare a tuturor neuronilor
n_{in}, n_{hidden}, n_{out}	numărul de neuroni din stratul 1, 2 respectiv 3

5.3.8.1 Pasul forward

$$Out_2[h] = F\Big(\sum_{i=1}^{n_{in}} w_{12}[h][i] \cdot Out_1[i] + \theta_2[h]\Big) \tag{5.37}$$

$$Out_3[o] = F\Big(\sum_{h=1}^{n_{hidden}} w_{23}[o][h] \cdot Out_2[h] + \theta_3[o]\Big) \tag{5.38}$$

$$E = \sum_{o=1}^{n_{out}} \big(Out_3[o] - Scop[o]\big)^2 \tag{5.39}$$

5.3.8.2 Pasul backward

$\Delta w = -\eta \cdot \frac{\partial E}{\partial w}$, unde w poate fi $\theta 3[o]$, $w23[o][h]$, $\theta 2[h]$, $w12[h][i]$ (5.40)

$$\frac{\partial E}{\partial \theta_3[o]} = 2 \cdot (Out_3[o] - Scop[o]) \cdot F'(Out_3[o]) \tag{5.41}$$

$$\frac{\partial E}{\partial w_{23}[o][h]} = 2 \cdot (Out_3[o] - Scop[o]) \cdot F'(Out_3[o]) \cdot Out_2[h] \tag{5.42}$$

$$\frac{\partial E}{\partial \theta_2[h]} = 2 \cdot \sum_{o=1}^{n_{out}} (Out_3[o] - Scop[o]) \cdot F'(Out_3[o]) \cdot w_{23}[o][h] \cdot F'(Out_2[h]) \tag{5.43}$$

$$\frac{\partial E}{\partial w_{12}[h][i]} = 2 \cdot \sum_{o=1}^{n_{out}} (Out_3[o] - Scop[o]) \cdot F'(Out_3[o]) \cdot w_{23}[o][h] \cdot F'(Out_2[h]) \cdot Out_1[i] \tag{5.44}$$

Considerând pentru funcţia F funcţia sigmoidă clasică derivata se determină uşor din valoarea funcţiei conform relaţiei:

$$F'(x) = F(x) \bullet \big(1 - F(x)\big) \tag{5.45}$$

5.3.9 Cercetări privind evitarea saturării ieşirii neuronilor

Această problemă se pune în cazul în care neuronii din stratul de ieşire au funcţia de activare sigmoidală. Pentru a adapta spaţiul problemei la spaţiul reţelei neuronale cea mai simplă soluţie este translatarea domeniului valorilor componentelor vectorilor aplicaţi ieşirii reţelei [$V_{MIN} \div V_{MAX}$] printr-o funcţie liniară în domeniul [L ÷ H] cu L = 0.0 şi H = 1.0 (având în vedere domeniul de ieşire al neuronilor cu funcţii de activare sigmoidale clasice).

Am comparat această primă variantă cu o a doua variantă în care am realizat translatarea în domeniul [L ÷ H] cu L = 0.1 şi H = 0.9. Pentru comparaţie, am ales o reţea feed-forward cu doi neuroni de intrare, doi în stratul ascuns şi un singur neuron de ieşire, cu funcţii de activare sigmoidale pentru toţi neuronii, care să rezolve binecunoscuta problema XOR. Am folosit la antrenare metoda backpropagation clasică, învăţare off-line şi rata de învăţare constantă $\eta = 1$.

Evoluţia comparativă a erorii reţelei în primii 5000 de paşi este prezentată în Fig. 5.7.

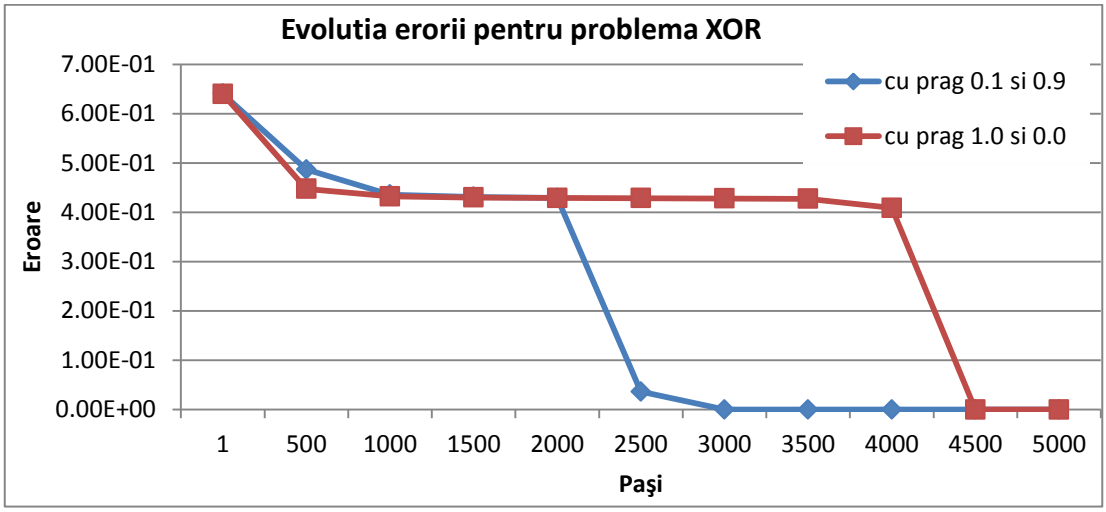


Fig. 5.7 Evoluţia erorii în cazul problemei XOR

În primele etape de antrenament, prima variantă duce la o învăţare mai rapidă deoarece eroarea determinată de reţea este mai mare. În următoarele însă, pe măsura apropierii valorii ieşirii de valorile de saturaţie ale funcţiei sigmoide, învăţarea devine foarte lentă. În varianta a doua, valorile dorite la ieşire se ating repede datorită evitării zonei de saturaţie a funcţiei sigmoide.

Rezultă deci, că scalarea domeniului semnalului de intrare la domeniul 0.1 ÷ 0.9 este benefică şi a fost aplicată în toate experimentele descrise în lucrare în care neuronii au funcţia de activare sigmoidală.

5.4 *Algoritmi evoluţionişti. Algoritmi genetici*

Algoritmii genetici fac parte din categoria algoritmilor de calcul evoluţionist şi sunt inspiraţi de teoria lui Darwin asupra evoluţiei. Ideea calculului evoluţionist a fost introdusă în 1960 de I. Rechenberg în lucrarea intitulată “Evolution strategies”.

Algoritmii genetici au fost aplicaţi cu succes într-o varietate de aplicaţii care necesită optimizarea globală a soluţiei. Algoritmii genetici se referă la un model introdus şi analizat de J. Holland în 1975, şi sunt proceduri adaptive care găsesc soluţia problemei pe baza unui mecanism de selecţie naturală şi evoluţie genetică. Algoritmul este des folosit pentru probleme în care găsirea soluţiei optime nu este uşoară sau cel puţin ineficientă (NP-hard), datorită

caracteristicilor căutării probabilistice. Algoritmii genetici codifică o soluţie posibilă la o problemă specifică într-o singură structură de date numită „**cromozom**" şi aplică operatori genetici la asemenea structuri astfel încât să menţină informaţiile critice.

Algoritmii genetici pornesc de la o mulţime iniţială de soluţii (de obicei aleasă aleator) numită în literatură „populaţie". În această populaţie, fiecare individ este numit „cromozom" şi reprezintă o soluţie posibilă a problemei. În aproape toate cazurile, cromozomul este un şir de simboluri numite gene (de obicei reprezentat ca un şir de biţi). Aceşti cromozomi evoluează pe durata iteraţiilor succesive numite generaţii. În fiecare generaţie, cromozomii sunt evaluaţi utilizând unele măsuri de evaluare (fitness). Pentru crearea următoarei populaţii, cei mai buni cromozomi din generaţia (populaţia) curentă sunt selectaţi, şi noii cromozomi sunt formaţi folosind unul dintre cei trei operatori genetici esenţiali: selecţia, crossover şi mutaţia [Ghos05].

Selecţia asigură că anumiţi cromozomi din generaţia curentă sunt copiaţi în acord cu valoarea funcţiei lor de evaluare în noua generaţie, ceea ce înseamnă că cromozomii cu o performanţă mare au o probabilitate mare să contribuie la formarea noii generaţii. Crossover este un alt operator genetic care reprezintă procesul prin care, pe baza a doi cromozomi din populaţia curentă sunt formaţi doi cromozomi pentru populaţia următoare. Mutaţia este procesul prin care un cromozom din populaţia curentă este modificat (o genă de obicei) şi salvat în noua populaţie.

5.4.1 Codificarea cromozomilor şi problema de optimizare

Algoritmii genetici au două componente principale care depind de problema abordată: codificarea problemei şi funcţia de evaluare. Cromozomii care reprezintă codificarea soluţiilor parţiale ale problemei trebuie într-o oarecare măsură să conţină informaţiile despre soluţia problemei şi, evident, depind foarte mult de natura acesteia. Există mai multe codificări, care au fost utilizate cu succes cum ar fi: codificarea binară (cromozomul este format din şiruri de 0 sau 1 care reprezintă binar soluţia problemei) sau codificarea prin valoare (cromozomul este format dintr-un şir / vector de valori întregi sau reale care, pe ansamblu, reprezintă soluţia problemei).

Funcţia de evaluare numită şi funcţia de „fitness" este funcţia care ne permite să dăm o încredere la fiecare cromozom din populaţie. Această funcţie este, de obicei, funcţia care reprezintă descrierea problemei.

Când trebuie să rezolvăm o problemă, de obicei ne uităm după anumite soluţii care sunt mai bune decât alte soluţii obţinute anterior. Spaţiul tuturor soluţiilor fezabile este numit spaţiul de căutare sau spaţiul stărilor. Problemele abordate folosind algoritmi genetici sunt de obicei probleme pentru care căutarea în spaţiul soluţiilor este o problemă complicată sau chiar NP-completă. De obicei nu ştim unde să căutăm o soluţie şi de unde să începem. Soluţiile obţinute folosind algoritmi genetici sunt de obicei considerate ca soluţii bune, deoarece nu este întotdeauna posibil să cunoaştem care este optimul real.

5.4.2 Metode de alegere a cromozomilor

Un alt pas important în algoritmul genetic este cum alegem părinţii din populaţia curentă care vor alcătui noua populaţie. Aceasta poate fi făcută în mai multe feluri, dar ideea de bază este de a selecta cei mai buni părinţi (în speranţa că aceştia vor produce cei mai buni copii). În acest pas poate apărea o problemă: generând noua populaţie doar pe baza noilor copii obţinuţi poate duce la pierderea celui mai bun cromozom obţinut până la acel pas. De obicei, această problemă este rezolvată prin utilizarea aşa numitei metode de „elitism". Adică, cel puţin un cromozom care produce cea mai bună soluţie conform cu funcţia de fitness este copiat fără nici o modificare în noua populaţie, astfel încât cea mai bună soluţie obţinută până la acel moment să nu se piardă.

Ca si metode de alegere a cromozomilor propunem două metode, fiecare dintre ele cu avantajele şi dezavantajele ei:

5.4.2.1 Metoda „Roulette Wheel" (ruleta)

În această metodă, fiecare individ din populaţia curentă este reprezentat printr-un spaţiu proporţional cu valoarea funcţiei lui de evaluare. Prin eşantionări aleatoare, succesive, din acest spaţiu de reprezentare a cromozomilor se asigură că cei mai buni cromozomi au şanse mai mari să fie selectaţi la un anumit pas decât cei cu mai slabi. Această metodă de selecţie va avea probleme în momentul în care valoarea funcţiei de evaluare diferă foarte mult de la un cromozom la altul. De exemplu, dacă cel mai bun cromozom are valoare funcţiei de evaluare mare (care va ocupa 90% din spaţiul de reprezentare) iar restul cromozomilor au valori ale funcţiilor de evaluare foarte mici, această metodă va selecta de foarte multe ori cromozomul cel mai bun ducând în final la degenerarea populaţiei printr-un fel de endogamie.

5.4.2.2 Alegerea utilizând metoda lui Gauss

Paşii propuşi pentru această metodă sunt:

1. Se alege un cromozom din populaţie curentă.
2. Utilizând formula lui Gauss calculăm probabilitatea ca acel cromozom să fie un cromozom bun (acela care obţine valoarea funcţiei de evaluare care depăşeşte un prag).

$$P(c_i) = e^{-\frac{(M - fitness(c_i))^2}{2\sigma^2}} \qquad (5.46)$$

unde $P(c_i)$ reprezintă probabilitatea calculată pentru cromozomul c_i, M reprezintă valoarea maximă care poate fi obţinută de către funcţia de evaluare şi σ care reprezintă dispersia sau panta cu care scade probabilitatea (viteza cu care scade probabilitatea), iar $fitness(c_i)$ reprezintă valoarea funcţiei de evaluare pentru acel cromozom. De exemplu, dacă valoarea maximă a funcţiei de evaluare este 1, M va fi 1 iar pentru dispersie propunem o valoare între 0.3-0.5.

3. În al treilea pas, această probabilitate calculată este comparată cu o probabilitate aleasă aleator în domeniul [0,1] (Probabilitatea lui Gauss întoarce valori în acest domeniu).
4. Se verifică dacă probabilitatea lui Gauss calculată pentru cromozomul ales este mai mare decât probabilitatea aleasă aleator:
    a. dacă da, cromozomul ales aleator se ia în considerare pentru a forma noua populaţie;
    b. dacă nu, se trece din nou la pasul 1.

Această metodă asigură posibilitatea luării în considerare şi a cromozomilor care nu au obţinut valori mari pentru funcţia de evaluare asigurând astfel diversitatea procesului (oferă şanse mai mari de evoluţie şi cromozomilor mai slabi).

5.4.3 Operatorii genetici utilizaţi

5.4.3.1 Selecţia

Pentru acest operator genetic, de obicei, se selectează doar un singur cromozom din populaţia curentă. Acest cromozom este copiat în noua populaţie fără nici o modificare. Această metodă se mai foloseşte şi pentru a nu pierde cromozomii care a obţinut cea mai bună valoare la funcţia de evaluare (elitism) din populaţia curentă. De asemenea, acest operator se aplică şi pentru alţi cromozomi selectaţi pe baza metodelor de selecţie propuse dar acest operator, în cazul meu, apare de un număr mic de ori la generarea noii populaţii.

5.4.3.2 Mutaţia

Mutaţia este un alt operator genetic important şi reprezintă un proces prin care cromozomul curent îşi modifică ocazional una sau mai multe valori (gene) într-un singur pas. Mutaţia depinde de asemenea de codificarea cromozomului. Mutaţia alege doar un singur candidat şi, aleator, modifică unele valori ale acestuia (modificând doar semnul acelei valori sau uneori se modifică şi valoarea – in cazul reprezentării prin valoare sau se schimbă doar valoarea în cazul reprezentării binare). Mutaţia funcţionează prin alegerea aleatoare a numărului de valori care vor fi schimbate şi a modului de modificare a acestora.

5.4.3.3 Crossover

Crossover poate fi văzut ca încrucişarea a 2 părinţi pentru a obţine copii, selectabili în generaţia următoare. Acest operator depinde foarte mult de tipul de codificare al cromozomilor. Metoda de crossover este aplicată la o pereche de părinţi aleşi folosind una din metodele prezentate. Cu o probabilitate p_c părinţii sunt recombinaţi pentru a forma doi noi copii care vor fi introduşi în noua populaţie. De exemplu luăm doi părinţi din populaţia curentă, îi împărţim şi încrucişăm componentele astfel încât să producem doi noi candidaţi. Aceşti candidaţi, în urma încrucişării, trebuie să reprezinte o soluţie posibilă pentru parametrii problemei noastre de optimizare; de aceea, de obicei, se inter-schimbă valori de pe aceleaşi poziţii. Utilizând un punct de recombinare, putem crea noii candidaţi prin combinarea primei părţi din primul părinte cu a doua parte din al doilea părinte.

5.5 *Algoritmi bazaţi pe nuclee. Support Vector Machine (SVM)*

Clasificarea Support Vector Machine (SVM) este o tehnică care se bazează pe teoria învăţării statistice prezentată în [Scho02],[Cris00]. Această tehnică a aplicat cu deosebit succes descoperiri matematice vechi de câteva secole, în probleme de optimizare peste mulţimi de date de dimensiuni mari, atât în ceea ce priveşte numărul de articole de clasificat cât şi în ceea ce priveşte numărul de caracteristici prin care se reprezintă acestea.

Algoritmul SVM trebuie să găsească un *hiperplan* (un plan de dimensiune *n-1* într-un spaţiu *n* dimensional) care să separe datele de intrare în două clase. De fapt, algoritmul constă în determinarea parametrilor ecuaţiei generale a planului (indicaţii practice de implementare se găsesc în [Hsu03]). Algoritmul original funcţionează doar în cazul a două clase, existând adaptări pentru mai multe clase. Privind astfel problema în plan, se doreşte de fapt găsirea unei linii care să împartă cât mai bine punctele aflate în clasa „pozitivă” (cele care se găsesc într-o

clasă) de cele din clasa „negativă” (restul punctelor). Pentru o mai bună ilustrare vom discuta în continuare pe un exemplu prezentat grafic în Figura 5.8.

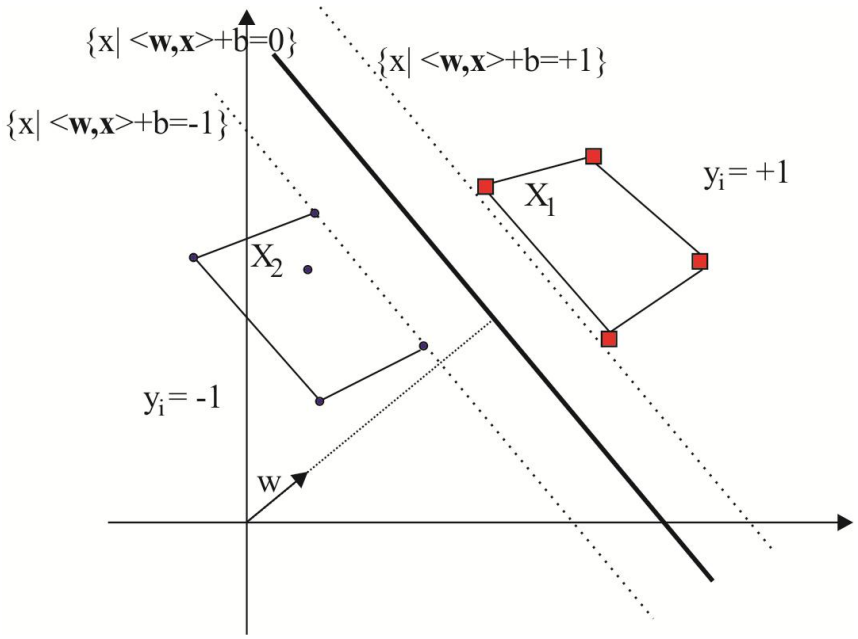


Fig. 5.8 – Determinarea hiperplanului optim de separare

În Fig. 5.8 considerăm că datele sunt împărţite în două clase: clasa „punct” şi clasa „pătrat”. În acest caz, problema ce se doreşte a fi rezolvată este trasarea unei linii optime de separare a celor două clase. Problema pare iniţial foarte simplă, trebuie însă să ţinem cont că o linie optimă de clasificare ar trebui să clasifice corect toate elementele ce pot fi generate cu aceeaşi distribuţie dată. Există o multitudine de hiperplane care îndeplinesc condiţiile de clasificare, dar algoritmul încearcă să determine hiperplanul optim. Adică orice puncte generate cu aceeaşi funcţie ca cele din grafic trebuie să fie clasificate corect. Hiperplanul ce va realiza această separare este determinat de o funcţie:

$$f(x) = \mathrm{sgn}\left(\langle w, \Phi(x)\rangle + b\right) \tag{5.47}$$

numită **funcţie de decizie**, unde *w* este vectorul pondere ortogonal pe hiperplan, ***b*** este un scalar ce reprezintă marginea hiperplanului, ***x*** este eşantionul curent şi ***Φ(x)*** este funcţia care trece pe ***x*** într-un spaţiu al caracteristicilor de dimensiune mai mare. În această funcţie am notat prin **<,>** produsul scalar a doi vectori ce va fi explicat ulterior, iar prin ***sgn*** am notat funcţia *signum* care returnează +1 dacă variabila de intrare este pozitivă, zero dacă este egală cu zero şi -1 altfel. În partea de antrenament a algoritmului trebuie să se determine vectorul normal ***w***, astfel încât marginea ***b*** să fie maximă.

Fie $x, y \in \mathbb{R}^n$ cu $x=(x_1, x_2, \ldots, x_n)$ şi $y=(y_1, y_2, \ldots, y_n)$ atunci spunem că produsul scalar a doi vectori este valoarea reală ce măsoară suprafaţa determinată de cei doi vectori; ea se notează $< x, y >$ şi se calculează:

$$< x, y >= x_1 \cdot y_1 + x_2 \cdot y_2 + \ldots + x_n \cdot y_n \tag{5.48}$$

şi spunem că **x** este ortogonal pe **y** dacă produsul lor scalar este egal cu zero, adică:

$$< x, y >= 0 \tag{5.49}$$

De asemenea, dacă *w* are lungimea unu, atunci <*w, Φ(x)*> este egal cu lungimea proiecţiei lui ***x*** pe direcţia lui *w*. În genere *w* va fi scalat prin $\| w \|$ pentru a obţine un vector de lungime

unitară. Prin $\| w \|$ se înţelege norma lui *w*, adică $\| w \| = \langle w , w \rangle$, numită şi normă euclidiană. În continuare se vor folosi vectori normaţi.

Determinarea hiperplanului de separare se bazează pe două trăsături fundamentale. Prima dintre ele precizează existenţa unui unic hiperplan optim de separarea a datelor. Hiperplanul optimal este acela care are cea mai mare valoare a lui b (margine maximă), adică distanţa de la cel mai apropiat punct de hiperplan la hiperplan este maximă pentru puncte aparţinând ambelor clase. A doua trăsătură precizează că, o dată cu creşterea marginii, scade puterea de separare a hiperplanului. Acestea sunt două trăsături antagonice care fac determinarea optimului un exerciţiu de compromis optimal (în condiţii restrictive).

Hiperplanul optim este soluţie a ecuaţiei:

$$\underset{w\in H, b\in\Re}{maximize} \min\left\{\left\|\mathbf{x}-\mathbf{x_i}\right\| \middle| \mathbf{x}\in\mathcal{H}, \left\langle \mathbf{w},\mathbf{x}\right\rangle + b = 0, i = 1,...,m\right\} \tag{5.50}$$

Acest algoritm se poate aplica numai într-un spaţiu vectorial. Pentru date separabile în spaţiul vectorial de intrare, hiperplanul se poate calcula direct din datele empirice. Pentru date ce nu sunt separabile în spaţiul de intrare, hiperplanul se va calcula într-un spaţiu de dimensiune mai mare, prin trecerea datelor într-un spaţiu al caracteristicilor de dimensiune mai mare utilizând funcţia Φ şi prin construirea unui hiperplan de separare în noul spaţiu.

Reprezentarea matematică a celor două trăsături, în scopul obţinerii hiperplanului optim, se face prin intermediul unei probleme de optimizare cu restricţii de forma:

$$\underset{w\in Hround, b\in\Re}{minimize}\ \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \tag{5.51}$$

cu restricţia $y_i(\langle w,x_i\rangle + b) \geq 1$ pentru toţi $i=1,...,m$, unde τ este numită funcţia obiectiv (prima trăsătură caracteristică), iar restricţia este numită inegalitatea de constrângeri (a doua trăsătură caracteristică). Pentru determinarea hiperplanului optim trebuie respectate amândouă condiţiile. Aceasta se numeşte problema de optimizare primară.

Observăm că restricţia asigură corectitudinea clasificării. Adică $f(x_i)$ va fi $+1$ pentru y_i egal cu $+1$ şi -1 pentru y_i egal cu -1. Să vedem intuitiv de ce trebuie să fie minimizată lungimea lui *w*. Dacă $||w||$ ar fi unu, atunci partea stângă din restricţie ar fi egală cu distanţa de la x_i la hiperplan. În general, trebuie să împărţim $y_i(\langle w,x_i\rangle + b)$ prin $||w||$, pentru a transforma aceasta în distanţă. De acum încolo, dacă putem satisface restricţia pentru toţi $i=1,..,m$ cu un *w* de lungime minimă, atunci marginea totală va fi maximizată.

Problema de optimizare primară este în genere dificil de rezolvat, datorită costurilor mari de calcul, fiind o problemă deschisă în teoria optimizării. Pentru rezolvarea acestei probleme există posibilitatea folosirii problemei duale, care, conform demonstraţiilor matematice, oferă aceeaşi soluţie. Pentru obţinerea problemei duale introducem Langrangian-ul şi multiplicatorii Lagrange:

$$L(\mathbf{w},b,\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m}\alpha_i(y_i(\langle \mathbf{x_i}, w\rangle + b) - 1) \tag{5.52}$$

cu multiplicatorii Lagrange α_i care trebuie să fie mai mari sau egali cu zero. Lagrangian ***L*** trebuie să fie minimizat în raport cu *variabilele primare w* şi *b* şi maximizat în raport cu *variabilele duale* $\boldsymbol{\alpha_i}$ (cu alte cuvinte, să găsească un punct de sprijin). Observăm că restricţiile sunt incorporate în al doilea termen al Lagrangian-ului, fără a mai fi nevoie să le aplicăm explicit.

Vom prezenta în continuare rezolvarea intuitivă a problemelor de optimizare cu restricţii. Dacă restricţia este violată, atunci $y_i(\langle w,x_i\rangle + b) - 1 < 0$, caz în care *L* poate fi incrementat prin incrementarea lui α_i corespunzător. În acelaşi timp, *w* şi *b* vor trebui modificate dacă *L* descreşte. Pentru ca $\alpha_i(y_i(\langle w,x_i\rangle + b) - 1)$ să nu devină un număr arbitrar negativ foarte mare, modificările

din *w* şi *b* vor asigura că, problema fiind separabilă, restricţiile vor fi eventual satisfăcute. Similar, putem înţelege că pentru toate constrângerile care nu se potrivesc exact cu egalitatea (adică, pentru care $y_i(\langle w,x_i\rangle + b) - 1 > 0$), α_i corespunzător trebuie să fie zero: aceasta este valoarea lui α_i care îl maximizează pe *L*. A doua este condiţia complementară a teoriei optimizării propusă de Karush-Kuhn-Tucker (*condiţiile KKT*). La punctul de sprijin derivatele Langrangian-ului *L*, în raport cu variabilele primare, trebuie să se anuleze:

$$\frac{\partial}{\partial b} L(w,b,\alpha) = 0 \quad şi \quad \frac{\partial}{\partial w} L(w,b,\alpha) = 0 \tag{5.53}$$

conducând la:

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \quad şi \quad w = \sum_{i=1}^{m} \alpha_i y_i x_i \tag{5.54}$$

Vectorul soluţie este o expansiune în raport cu submulţimea de pattern-uri de antrenament, adică acele pattern-uri cu α_i diferit de zero, numite *Support Vectors (SVs)*. Alături de condiţiile KKT:

$$\alpha_i \left[y_i \left(\langle x_i, w \rangle + b \right) - 1 \right] = 0 \text{ pentru toţi } i = \overline{1,m} \tag{5.55}$$

vectorii suport vor sta pe margine. Toate exemplele de antrenament rămase (x_i,y_i) sunt irelevante: restricţiile lor $y_i (\langle x_i,w \rangle + b) - 1 \geq 1$ pot fi foarte bine eliminate şi nu vor apărea în formula finală. Aceasta ne arată intuitiv că, deoarece hiperplanul este complet determinat de către pattern-urile cele mai apropiate de el, soluţia ar trebui să nu depindă de celelalte exemple.

Prin înlocuirea formulelor din (5.54) în Lagrangian, se elimină variabilele primare *w* şi *b*, obţinând astfel aşa-numita „*problemă de optimizare duală"*, care se rezolvă de obicei în practică:

$$\underset{\alpha \in \mathbb{R}^m}{maximize}\ W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left\langle x_i, x_j \right\rangle \tag{5.56}$$

cu condiţia $\alpha_i \geq 0$ pentru $i = \overline{1,m}$ şi

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{5.57}$$

Astfel, funcţia de decizie pentru hiperplan poate fi scrisă ca:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \left\langle x, x_i \right\rangle + b \right) \tag{5.58}$$

unde *b* este calculat folosind condiţiile KKT.

În rezolvarea problemei duale, este de asemenea frecvent cazul în care doar un subset de restricţii devine activ. Pentru exemplificare, dacă păstrăm o bilă într-o cutie, atunci, de obicei, se va rostogoli într-unul din colţuri. Restricţiile corespunzătoare zidurilor care nu sunt atinse de către bilă sunt irelevante şi aceste ziduri ar putea fi eliminate.

În practică, hiperplanul de separare poate să nu existe, de exemplu dacă clasele sunt suprapuse. Pentru a ţine seama de exemplele posibile de violare a restricţiilor, se pot introduce variabile discrete notate cu $\xi_i \geq 0$ *pentru toti* $i = 1,...,m$.

Folosind variabilele discrete, constrângerile devin: $y_i (\langle w,x_i \rangle + b) \geq 1 - \xi$ pentru toţi $i=1,...,m$. Hiperplanul optim găsit acum este determinat atât de controlul capacităţii de clasificare (via $||w||$), cât şi de suma de variabile discrete $\Sigma_i \xi_i$. Se observă că cea de-a doua parte furnizează o limită superioară a numărului de erori de antrenament. Aceasta se numeşte clasificarea cu margine soft. Funcţia obiectiv devine:

$$\tau(w,\xi)=\frac{1}{2}\|w\|^2+C\sum_{i=1}^{m}\xi_i \tag{5.59}$$

cu aceleaşi restricţii, unde constanta $C > 0$ determină schimbul dintre maximizarea marginii şi minimizarea erorii de antrenare. Dacă rescriem în termeni de multiplicator Lagrange, aceasta conduce la aceeaşi problemă de maximizare, în plus, apărând următoarea restricţie:

$0\leq \alpha_i \leq C$ pentru toţi i=1,...,m şi

$$\sum_{i=1}^{m}\alpha_i y_i = 0 \tag{5.60}$$

În cazul în care găsirea unui hiperplan optim în spaţiul datelor de intrare implică un cost mare, ideea SVM este de a mapa datele de intrare într-un nou spaţiu de dimensiune mai mare prin intermediul unei funcţii Φ şi încercarea de a găsi un hiperplan optim cu cost cât mai mic în acest nou spaţiu. Aceasta conduce în mod evident la obţinerea unei graniţe de decizie neliniare în spaţiul datelor de intrare. Deoarece transferarea tuturor datelor dintr-un spaţiu într-un spaţiu de dimensiune mai mare ar impune un efort de calcul mare, se foloseşte un mecanism numit în literatura de specialitate „trucul nucleu”. Acesta poate fi aplicat aici deoarece toate trăsăturile caracteristice ale vectorilor apar numai în produse scalare. Pentru calculul produsului scalar a doi vectori x şi x’ „trucul nucleu” substituie produsul scalar cu un nucleu, urmând ca doar acesta să se calculeze în noul spaţiu. In orice spaţiu, produsul scalar este un scalar iar „trucul nucleu” se poate aplica doar în cazul nucleelor pozitiv definite. Trucul nucleu este:

$$k\left(x,x'\right)=<x,x'> \tag{5.61}$$

Funcţia de decizie în noua formă devine:

$$f(x)=\operatorname{sgn}\left(\sum_{i=1}^{m}y_i\alpha_i\left\langle\Phi(x),\Phi(x_i)\right\rangle+b\right)=\operatorname{sgn}\left(\sum_{i=1}^{m}y_i\alpha_i k(x,x_i)+b\right) \tag{5.62}$$

Avantajul acestui algoritm este că nu implică transformarea tuturor datelor într-un spaţiu de dimensiune mai mare, deci nu necesită calcule costisitoare cum ar fi la reţelele neuronale. De asemenea se obţine o dimensiune redusă a setului de date de antrenament, având în vedere că în faza de testare se vor lua în calcul doar vectorii suport care, de obicei, sunt puţini. Alt avantaj al acestui algoritm este că acceptă o dimensiune a datelor de intrare oricât de mare, fără a creşte exponenţial timpul de calcul. Acest fapt nu este valabil în cazul reţelelor neuronal; de exemplu, algoritmul backpropagation (prezentată în [Mitc97]) funcţionează foarte greu sau chiar deloc pentru vectori de dimensiune mai mare de 35. Singura problemă a algoritmului bazat pe vectori suport este numărul de vectori suport rezultaţi. Cu cât acesta este mai mare, cu atât creşte timpul de răspuns.

Tot algoritmul anterior s-a bazat pe faptul că datele de intrare se găsesc doar în două clase, acesta fiind de fapt un algoritm de clasificare binară în care etichetele pot lua doar două valori. Majoritatea problemelor din lumea reală necesită mai mult de două clase la o problemă de clasificare. Există câteva metode pentru a putea lucra cu mai multe clase, folosind clasificarea binară. O metodă ar fi clasificarea „unu versus restul”, în care se aleg elementele care aparţin unei clase pentru a se diferenţia de restul elementelor. În acest caz, se calculează pentru fiecare clasă în parte un hiperplan optim care separă clasa respectivă de restul claselor. Pentru ieşirea algoritmului se alege valoarea maximă a tuturor hiperplanelor de separare:

$$\underset{j=1,\ldots,M}{argmax}\, g^j(x),\ unde\ g^j(x)=\sum_{i=1}^{m}y_i\alpha_i^j k(x,x_i)+b^j \tag{5.63}$$

Pentru M clase se calculează M hiperplane, această metodă având o complexitate liniară. O altă abordare a clasificării la mai multe clase este „clasificarea perechilor”, în care sunt alese perechi de câte două clase şi se calculează câte un hiperplan de separare pentru fiecare pereche.

Pentru *M* clase se calculează $C_M^2 = (M-1)*M/2$ hiperplane, această metodă având o complexitate polinomială. Avantajul acesteia este că, de obicei, hiperplanul optim este de dimensiune mai mică (are mai puţini vectori suport).

5.6 *Clasificatori hibrizi. Metaclasificatori*

Metaclasificarea sau clasificarea hibridă se bazează pe alegerea clasificatorului (algoritmului) corect pentru o problemă specifică, pe baza caracteristicilor vectorului de intrare şi a clasificărilor anterioare. O problemă principală, care apare când se utilizează mai mulţi clasificatori clasificatori, este de a determina dacă clasificatorul ales este util şi pentru noi instanţe. Utilizarea unor metaclasificatori sau ansambluri de clasificatori este o soluţie simplă de rezolvare a acestei probleme. Având mai mulţi clasificatori de bază, ideea este de a învăţa un metaclasificator care prezice gradul de corectitudine pentru fiecare dintre clasificatorii de bază.

Selectarea unui clasificator pentru a eticheta o instanţă se face în funcţie de încrederea acordată acelui clasificator, încredere dobândită în urma clasificărilor corecte realizate de acesta pentru instanţe de tipul respectiv.

Metaclasificatorului cere fiecărui clasificator de bază inclus să atribuie o clasă la instanţa curentă cu o anumită probabilitate şi apoi metaclasificatorul să decidă care clasificare este cea mai demnă de încredere. Pe lângă creşterea acurateţei de clasificare prin exploatarea sinergismului mai multor clasificatoare, un alt avantaj al metaclasificării constă în posibilitatea de exploatare a paralelismelor funcţionale (utilizând sisteme multiprocesor, inclusiv în implementări comerciale de tipul High Performance Computing).

Studii experimentale efectuate în domeniul învăţării automate au demonstrat că dacă se combină ieşirile mai multor clasificatori se poate reduce eroarea generală [Domi96], [Quin96] [Opit99], [Jaeg08].

Metaclasificatorii pot face uz de diversitatea clasificatorilor utilizaţi pentru a reduce eroarea de varianţă [Turn99] fără a mări eroarea de prag (bias error). În anumite situaţii un metaclasificator poate reduce eroarea de prag, cum se arată în teoria pentru clasificatori cu margini largi (large margin classifiers) [Bart98]. Utilizarea ansamblurilor de clasificatori este utilă în diverse domenii: clasificare de text, finanţe [Leig02], bioinformatică [Tan03], medicină [Mang04], industrie [Maim04], geografie [Bruz04].

Având în vedere utilitatea metaclasificatorilor, prezint în continuare metodele semnificative din acest domeniu. Există câţiva factori care diferenţiază tipurile de ansambluri. Aceşti factori sunt după [Roka05]:

1. interacţionarea clasificatorilor în cadrul metaclasificatorului: din acest punct de vedere există două mari categorii: metaclasificatori secvenţiali sau concurenţi;
2. combinarea clasificatorilor: strategia combinării este generată de către un algoritm inductiv. Cel mai simplu mod de a combina clasificatorii este de a determina o ieşire doar pe baza ieşirilor din clasificatorii individuali. În [Ali96] şi [Mora06_b] sunt prezentate câteva metode de combinare: votul majoritar, combinare Bayesiana, însumarea distribuţiilor sau combinarea probabilităţilor. În [Turn99] se face o analiză teoretică pentru a estima îmbunătăţirea clasificării într-un ansamblu de clasificatori;
3. generarea diversităţii: pentru a face ca un ansamblu de clasificatori să lucreze eficient trebuie ca între clasificatori să existe o anumită diversitate. Această diversitate poate fi obţinută prin diferite forme de prezentare a datelor de intrare.

Great things are done by a series of small things brought together.
Vincent Van Gogh

6 Cercetări privind proiectarea sistemelor complexe de clasificare

În acest capitol voi prezenta cercetările mele privind proiectarea, realizarea şi testarea unor metaclasificatori complecşi. Am pornit de la premisa că există în principiu două categorii importante de metaclasificatori, şi anume o categorie care nu are o etapă de învăţare şi o altă categorie care poate să înveţe pe baza exemplelor anterioare. Din cei care învaţă, există metaclasificatori care, după învăţare, nu mai îşi modifică ponderile (nu se mai adaptează conform noilor instanţe pe care le-a clasificat) şi metaclasificatori care se adaptează continuu. Astfel, am proiectat şi realizat metaclasificatori din cele două categorii, numiţi în continuare metaclasificatori neadaptivi (cei fără învăţare şi cei cu învăţare dar care nu se mai adaptează în timpul clasificării) şi respectiv metaclasificatori adaptivi (care se adaptează în timpul clasificării). Scopul construirii unor metaclasificatori se bazează pe **ipoteza ştiinţifică** că mai mulţi clasificatori simpli pot obţine împreună, exploatând sinergismul lor, rezultate mai bune decât un singur clasificator complex.

Metaclasificatorii propuşi în continuare au avut două direcţii majore de dezvoltare. Prima direcţie a fost de a modifica modul cum se alege clasa învingătoare. Într-o primă etapă, în metaclasificatorii realizaţi am combinat mai mulţi clasificatori de tip SVM şi un clasificator de tip Bayes, pentru a crea un metaclasificatori care să îmbunătăţească acurateţea clasificării. Am calculat limita teoretică maximă a acurateţei de clasificare, la care se poate ajunge folosind această combinaţie de clasificatori, şi ea este de 98.63% (adică procentajul de documente care pot fi clasificate corect de cel puţin un clasificator). În acest context mi-am propus îmbunătăţirea schemei de clasificare prezentată în [Mora08]. Rezultatele au fost obţinute pe seturile A1 şi T1. În această secţiune, calitatea clasificării realizată de algoritm este prezentată ca şi acurateţe de clasificare, calculată ca raport de documente corect clasificate din numărul total de documente care trebuiau clasificate în faza de testare.

Au fost implementaţi 3 tipuri de metaclasificatori: unul neadaptiv bazat pe votul majoritar şi doi adaptivi, bazaţi pe cozi de erori, unul folosind distanţa euclidiană şi unul bazat pe distanţa cosinus. Dintre toţi aceştia, metaclasificatorul bazat pe distanţa cosinus a îmbunătăţit acurateţea de clasificare, ajungând la valoarea de **93.10%** [Cret08].

A doua direcţie de dezvoltare a fost combinarea tuturor ieşirilor clasificatorilor incluşi în metaclasificator şi ponderarea poziţiilor claselor din ieşirile clasificatorilor. Aceste ponderi pot fi date sau calculate pe baza unui algoritm genetic în cazul metaclasificatorilor neadaptivi.

De asemenea voi prezenta realizarea unui nou metaclasificator hibrid. Acesta este realizat dintr-un metaclasificator neadaptiv care va folosi o sumă ponderată pentru stabilirea clasei finale, urmat de un metaclasificator neuronal adaptiv. Acest metaclasificator neuronal utilizează un metaclasificator neadaptiv cu rol de preclasificare, şi o reţea neurală cu rol de postclasificare.

6.1 *Evaluarea clasificatorilor de tip SVM*

În [Mora07] este prezentat un metaclasificator bazat pe 8 clasificatoare de tip SVM care era folosit pentru îmbunătăţirea acurateţei de clasificare a documentelor de tip text. Maximul acurateţei de clasificare obţinut de către un singur clasificatorul de tip SVM a fost de 87.11% şi a fost obţinut de clasificatorul SVM de tip polinomial de grad 2 cu reprezentare Cornell Smart. În [Mora07] sunt prezentaţi şi testaţi mai mulţi clasificatori de tip SVM, bazaţi atât pe nucleul polinomial cât şi pe cel Gaussian, cu diferite forme de reprezentare. Formulele pentru modurile de reprezentare utilizate au fost prezentate în secţiunea 4.1.1.2. Dintre toţi clasificatorii testaţi şi prezentaţi, s-au inclus în metaclasificator 8 clasificatori SVM distincţi. Alegerea celor 8 clasificatori s-a făcut pe baza acurateţei de clasificare obţinută de aceştia pe diferite seturi de documente. Pentru metaclasificatorul din [Mora07] s-au ales clasificatorii de tip SVM cu cea mai bună acurateţe de clasificare astfel:

NR. CRT.	TIPUL NUCLEULUI	GRAD	REPREZENTAREA DATELOR
1	Polinomial	1	Nominal
2	Polinomial	2	Binar
3	Polinomial	2	Cornell Smart
4	Polinomial	3	Cornell Smart
5	Gaussian	1.8	Cornell Smart
6	Gaussian	2.1	Cornell Smart
7	Gaussian	2.8	Cornell Smart
8	Gaussian	3.0	Cornell Smart

Tabel 6.1 - Clasificatorii de tip SVM aleşi în metaclasificatorul din [Mora07]

Utilizând aceşti clasificatori, s-a ajuns la o acurateţe maximă de clasificare de 92.04% în cazul metaclasificatorului bazat pe distanţa euclidiană şi după un număr de 14 paşi de învăţare. Tot în [Mora07] s-a prezentat şi o analiză în care se calcula şi limita maximă la care ar putea să ajungă metaclasificatorul astfel creat (cu cei 8 clasificatori selectaţi). Limita calculată era de 94.21%. Această limită a clasificării s-a obţinut deoarece, din 2351 de documente de test, 136 de documente nu au putut fi clasificate corect de nici un clasificator selectat în cadrul metaclasificatorului.

În prima fază am încercat găsirea unui nou clasificator care să reuşească să clasifice corect documentele ce s-au dovedit imposibil de clasificat de către toţi clasificatorii selectaţi în metaclasificator din [Mora07]. În Fig. 6.1 am reprezentat schematic cercetările efectuate.

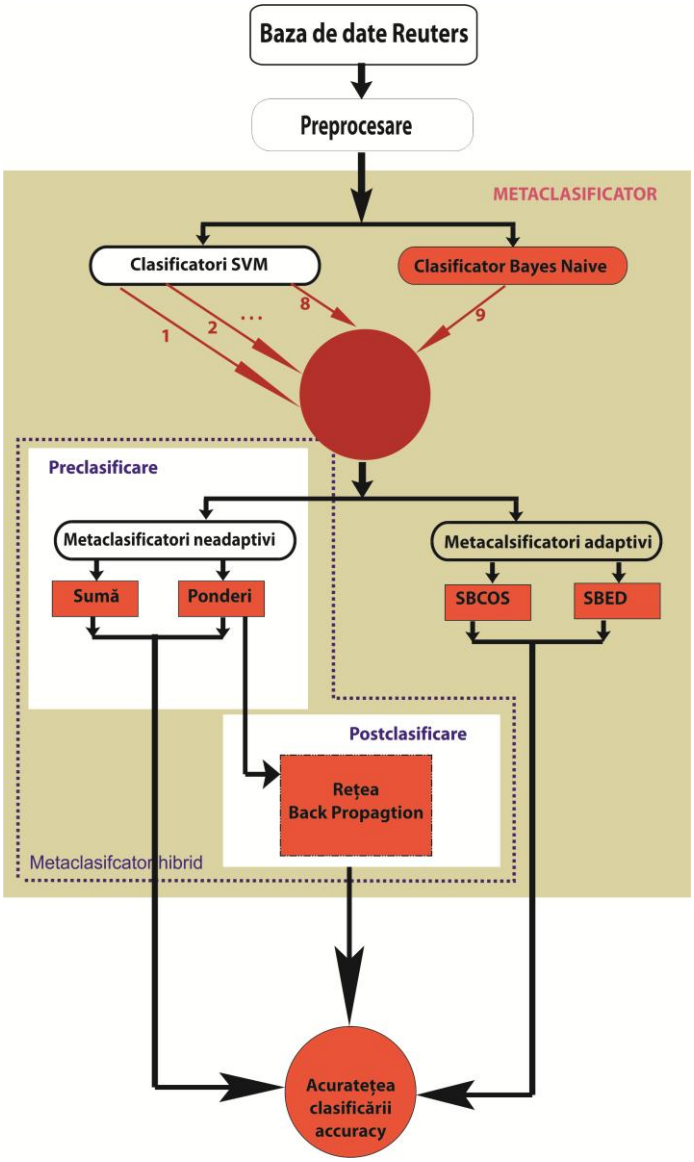


Fig. 6.1 Procesul de realizare a metaclasificatorilor

6.1.1 **Problema limitării metaclasificatorului cu clasificatori de tip SVM**

După cum am mai arătat, o parte din documentele de testare nu pot fi clasificate corect de nici un clasificator selectat în cadrul metaclasificatorului, fapt ce duce la o acurateţe a clasificării - maximum obtenabilă - de 94,21%.

Pentru a verifica clasificatorii în condiţiile prezentate în [Mora07], am antrenat clasificatorii SVM pe setul de date A1 (4.702 exemple şi 1.309 atribute) şi am utilizat ca date de testare setul de date T2 (136 de exemple - cele cu probleme - şi 1.309 atribute). Având în vedere faptul că documentele din setul T2 nu au putut fi clasificate corect, în urma efectuării testelor, după cum ne aşteptam, rezultatul clasificării corecte este aproape de 0%. Totuşi există clasificatori SVM care nu au fost selectaţi în cadrul metaclasificatorului, dar care reuşesc să clasifice corect o parte din cele 136 documente. Aceşti clasificatori nu au fost selectaţi, deoarece au avut o acurateţe de clasificare pe setul T1 mai slabă, dar se pare că pe setul T2 dau rezultate mai bune. În Fig. 6.2 şi Fig. 6.3 prezint rezultatele pentru acurateţea clasificării obţinute de clasificatorii de tip SVM, utilizând nucleul gaussian (notat în figurile următoare cu RBF) respectiv nucleul polinomial (notat în figurile următoare POL). În ceea ce priveşte nucleul polinomial, pentru gradul nucleului s-au luat valori numere întregi între 1 şi 5 iar în ceea ce

priveşte nucleul gaussian, pentru parametrul *C* s-au luat valori reale cuprinse între 1.0 şi 2.8. Ambele nuclee au fost testate pentru toate 3 tipurile de reprezentări (binară, nominală şi Cornell Smart) prezentate în secţiunea 4.1.1.2.

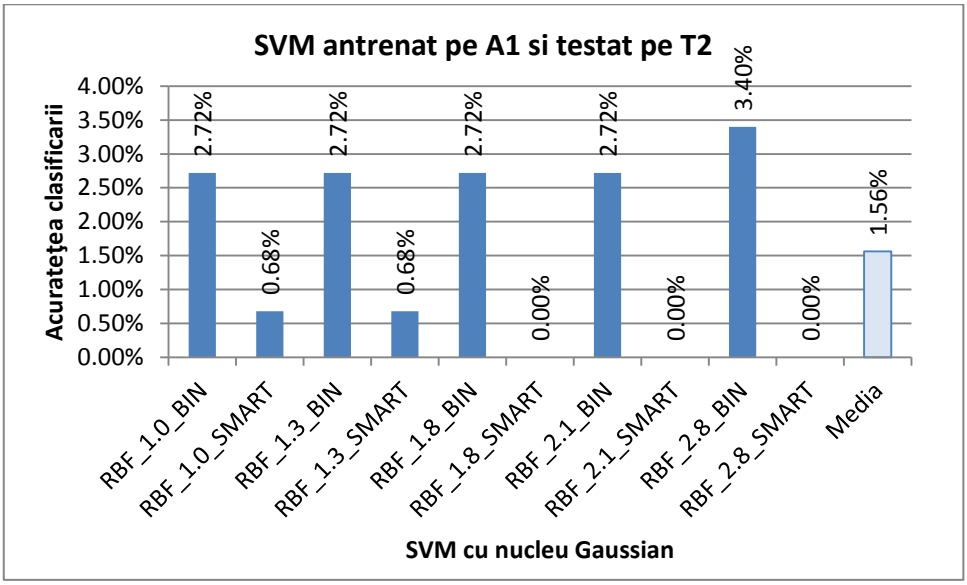


Fig. 6.2 Rezultate obţinute de clasificatorii SVM pe setul de date de antrenament A1 şi testat pe T2 - nucleu gaussian

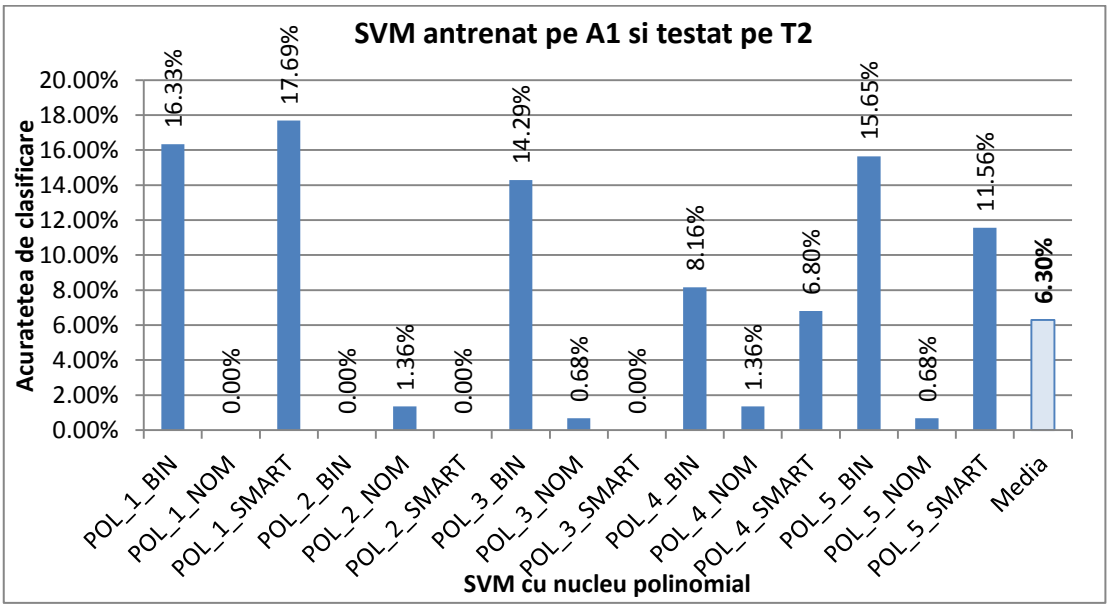


Fig. 6.3 Rezultate obţinute de clasificatorii SVM pe setul de date de antrenament A1 şi testat pe T2 - nucleu polinomial

Doar clasificatorul cu nucleu polinomial de grad 1 şi reprezentare Cornell Smart a reuşit clasificarea corectă a 24 de documente din cele 136. Avem două explicaţii:

a. documentele care nu pot fi clasificate, apar în prea puţine clase sau sunt cazuri particulare ale unei clase;
b. au fost greşit clasificate în baza de date Reuters.

6.1.2 O primă tatonare a problemei

Într-o prima etapă, pentru a testa dacă documentele ar putea fi clasificate corect, am ales ca set de antrenament pentru clasificatorii de tip SVM (selectaţi în metaclasificatorul prezentat) setul de date T1, care au fost utilizate în [Mora07] ca şi set de test. În cadrul acestui set de date se regăsesc şi cele 136 de exemple care nu au putut fi clasificate corect de către metaclasificator. Cu alte cuvinte, clasificatorii de tip SVM au fost antrenaţi pe acel set de date, care conţine şi documentele considerate cu probleme. Precizez aici că, antrenând SVM-urile doar pe documentele cu probleme – cele 136 documente (setul T2), majoritatea, după antrenare, au reuşit să clasifice corect toate documentele cu probleme, ceea ce era de aşteptat de altfel. În Fig. 6.4 şi Fig. 6.5 sunt prezentate rezultatele clasificării obţinute pe un setul de testare T2 (doar 136 documente), dar antrenate pe setul T1.

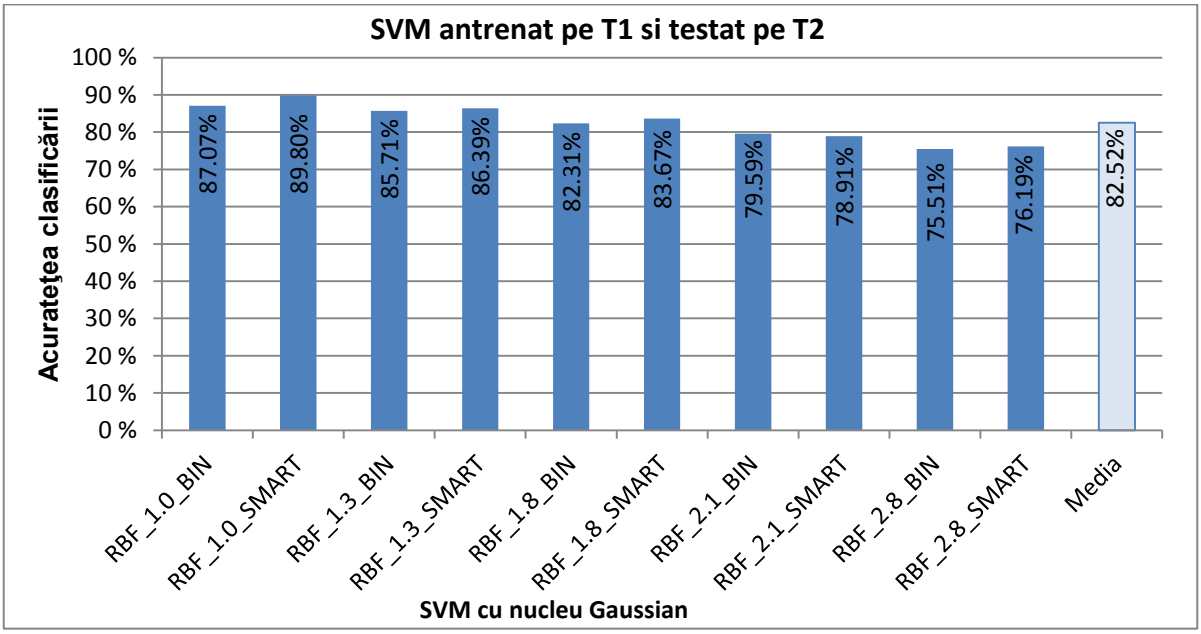


Fig. 6.4 Rezultate obţinute de clasificatorii SVM, utilizând diferite tipuri de reprezentare a datelor (binar şi Cornell-Smart), cu nucleu de tip Gaussian

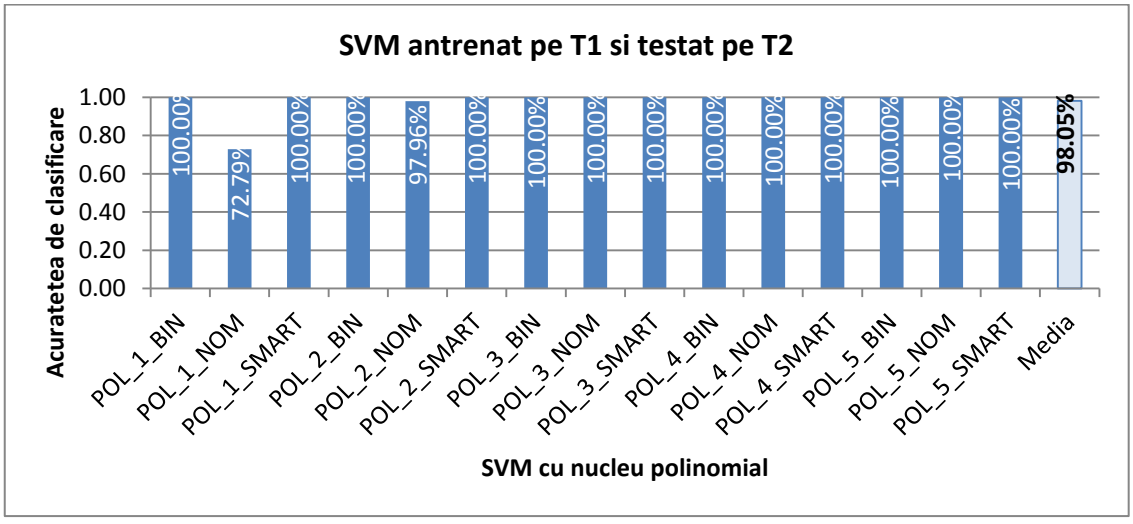


Fig. 6.5 Rezultate obţinute aplicând clasificatori SVM, utilizând diferite tipuri de reprezentare a datelor (binar, nominal şi Cornell-Smart), cu nucleu de tip polinomial

Observăm că rezultatele obţinute cu clasificatori SVM, care utilizează nucleul polinomial, au o acurateţe a clasificării mai mare (media este de 98,05%) decât cei care utilizează nucleul Gaussian (media este de 82,52%). Remarcăm faptul că din 15 clasificatori cu nucleu polinomial,

doar doi clasificatori - polinomial de grad 1 şi 2 cu reprezentarea datelor de tip nominal - nu au reuşit o acurateţe a clasificării de 100% a documentelor.

Rezultatele de mai sus pot fi explicate prin faptul că utilizarea la clasificatorii de tip SVM a unui nucleu polinomial (liniar) duce la rezultate mai bune, deoarece s-a constatat că documentele text în reprezentarea ca şi vectori de termeni sunt liniar separabile. Transformarea acestor vectori într-un nou spaţiu, folosind funcţii neliniare (nucleu Gaussian), îngreunează găsirea unui hiperplan optim de separare, fapt confirmat şi în [Gabr04].

6.2 *Soluţii explorate pentru îmbunătăţirea metaclasificatorului bazat pe clasificatoare de tip SVM*

6.2.1 Soluţia introducerii unor noi clasificatori SVM

O soluţie a fost introducerea dinamică de noi clasificatori în metaclasificator, în cazul în care în faza de antrenare/testare ar apărea un anumit număr (de ex. minim 50) de documente care nu pot fi clasificate corect de niciunul dintre clasificatorii deja existenţi. Astfel, s-a introdus un nou clasificator SVM de tip polinomial, antrenat special pe acele (50) documente. Această soluţie a fost ulterior abandonată, deoarece nu a produs rezultate mai bune.

6.2.2 Soluţia alegerii altei clase

O altă soluţie ar consta în alegerea unei alte categorii pentru un document dificil clasificabil, pentru care, de asemenea, clasificatorul întoarce un răspuns pozitiv mare. Dacă nici un clasificator nu va fi selectat pentru clasificarea unui document, conform regulilor prezentate în [Mora07], atunci se va alege clasificatorul cu cea mai mare probabilitate de reuşită (distanţa între documentul curent şi toate documentele din coada clasificatorului este maximă, chiar dacă este mai mică decât pragul stabilit). De la clasificatorul astfel selectat nu se va mai alege prima clasă propusă, ci următoarea, dacă ea este suficient de aproape faţă de prima clasă. Această abordare se va prezenta mai bine pe un exemplu.

Pentru exemplificare presupunem că în metaclasificator avem 4 clasificatoare diferite, fiecare având în coada de erori un număr de documente. Când avem un nou document d_n care trebuie clasificat, se ia pe rând fiecare clasificator şi se calculează distanta între d_n şi fiecare document din coada de erori a clasificatorului respectiv. Dacă cel puţin o distanţă calculată este mai mică decât pragul stabilit, metaclasificatorul nu va folosi acel clasificator pentru a clasifica documentul d_n. În cazul în care se rejectează astfel toţi clasificatorii, metaclasificatorul totuşi va alege pe cel care are distanţa cea mai mare obţinută (chiar dacă este mai mică decât pragul stabilit). Actualmente, metaclasificatorul va prezice clasa specificată de acest clasificator, chiar dacă se ştie cu o probabilitate mare că acesta va clasifica prost documentul d_n. Modificarea ar fi ca, în acest caz, clasificatorul ales să nu mai selecteze clasa pentru care se obţine valoarea cea mai mare (pentru că oricum va da greş, deoarece clasifică prost tipul respectiv de documente), ci să aleagă clasa imediat următoare din lista de clase pe care le prezice. Se va alege următoarea clasă prezisă doar dacă valoarea pentru aceasta este suficient de apropiată de valoarea maximă obţinută de clasificator (în limita unui prag ε). În acest caz, clasificatorul ar specifica o altă clasă pentru documentul curent d_n.

Rezultatele obţinute utilizând această ipoteză le vom prezenta în secţiunea 6.3

6.2.3 Soluţia adăugării unui clasificator de alt tip

O altă soluţie pentru îmbunătăţirea metaclasificatorului ar fi includerea unui alt clasificator, nu neapărat de tip SVM, care să reuşească să clasifice documentele cu problemă (cele 136), evident fără să fie antrenat pe acele documente. Pentru aceasta am făcut câteva experimente cu un clasificator de tip Naive Bayes. În acest caz, am folosit clasificatorul Bayes din pachetul IR, pus la dispoziţie de Universitatea din Texas [WEB09]. Într-o primă etapă, l-am folosit aşa cum este implementat în [WEB09], apoi i-am adus anumite modificări pentru a putea fi integrat ulterior în metaclasificator proiectat în [Mora07]. Modificările aduse se referă la modul de funcţionare în cazul clasificării în mai multe clase, la comportamentul acestuia când există documente clasificate iniţial în mai multe clase şi la modul de selecţie a datelor de antrenare şi testare.

Clasificatorul Naive Bayes nemodificat (BNN) testat primeşte, în cazul de faţă, ca date de intrare, toate fişierele care urmează a fi clasificate, urmând ca alegerea datelor de antrenament şi a datelor de test să se facă după metoda "*n-fold crossvalidation*" prezentată în secţiunea 5.2. Reamintim ideea acestei metode este de a împărţi un set de date în *n* subseturi, urmând ca *n-1* subseturi să se folosească la antrenare iar testarea să se facă pe subsetul nefolosit la antrenament, adică antrenarea şi testarea să se execute pe seturi de date disjuncte. Algoritmul se va executa de *n* ori, astfel încât fiecare subset de date va fi o singură dată un subset de test pentru verificarea antrenării. Din păcate, în acest caz nu mai pot fi selectate exact seturile prezentate la început, fapt ce a dus la modificarea modului de lucru al clasificatorului iniţial din clasa IR şi transformarea acestuia în alt algoritm pe care îl notăm BNA (Bayes Naive adaptat).

6.2.3.1 Adaptarea clasificatorului Bayes pentru utilizarea în metaclasificator

Pentru a putea integra şi clasificatorul Bayes în metaclasificatorul prezentat în [Mora07], au trebuit făcute câteva modificări. Clasificatorului Bayes primeşte la intrare un set de date din care el alege aleator subseturi de antrenament şi de test, după metoda propusă în [WEB09]. În prima fază, am modificat modul de alegere a setului de antrenament şi a celui de testare astfel încât acestea să fie identice cu seturile de antrenare şi testare folosite pentru celelalte clasificatoare din metaclasificator (SVM polinomial, SVM gaussian).

În cazul clasificatorului Bayes, acesta se va antrena întotdeauna pe acelaşi set de antrenament (A1) şi se va testa pe alt set de date (T1). Astfel, datele de antrenare şi de testare devin identice pentru toate clasificatoarele din metaclasificator.

În prima fază am încercat o clasificare la mai multe clase, de genul „*one class versus the rest*”. În cazul în care un document aparţinea mai multor clase (ceea ce este obişnuit la baza de date Reuters), acel document a fost trecut în setul de antrenare de mai multe ori, în funcţie de numărul de clase specificate de Reuters pentru acel document. Aceasta face ca multe clase să fie suprapuse (există multe clase care sunt subcategorii ale unei clase de bază – atunci toate documentele dintr-o subclasă pot să fie şi într-o altă clasă de bază).

Am luat în considerare prima dată această opţiune, deoarece în clasificatorii de tip SVM această metodă („one class versus the rest”) este folosită pentru antrenarea la mai mult de două clase.

Am modificat antrenarea BNA pe cele două seturi fixe (A1 şi T1), eliminând clasele în cazul în care un document aparţinea mai multor clase. De exemplu, dacă fişierul *file134.xml* era clasificat în Reuters ca aparţinând clasei *C15* şi clasei *C151,* am eliminat clasa *C151*. Astfel, un document se consideră că aparţine doar primei categorii specificate de Reuters, celelalte categorii fiind ignorate. Rezultă astfel doar 16 categorii distincte care vor fi folosite pentru clasificatorul Bayes (cel modificat).

O altă modificare adusă clasificatorului BNA, pentru a funcţiona pe aceleaşi set de date şi a furniza rezultatul în acelaşi mod ca şi clasificatorii SVM, a fost schimbarea modului în care se validează rezultatul clasificării. Deoarece în baza de date Reuters documentele erau clasificate în două sau mai multe categorii, unele din acestea fiind însă subcategorii ale unei categorii de bază, am validat rezultatul ca fiind corect dacă clasificatorul Bayes specifică corect una din clasele precizate de Reuters. De exemplu, dacă clasificatorul stabileşte pentru un document clasa *C151*, iar în Reuters el apare în *C15* şi apoi în *C151*, am considerat rezultatul clasificării ca fiind corect.

Clasificarea cu BNA s-a îmbunătăţit considerabil, ajungând la o acurateţe de 72,14% aproape identică cu cea atinsă de clasificatorul BNN din [WEB09]. Diferenţa apare deoarece în primul caz clasificatorul primeşte un set de date pe care îl împarte el automat în subseturi de antrenare şi testare, prezentând la sfârşit o medie a rezultatelor obţinute pe aceste subseturi, iar în al doilea caz clasificatorul primeşte un set fix de antrenare şi un set fix de testare, iar rezultatul de 72,14% este cel obţinut pe aceste seturi fixe.

Deşi prezentarea mediei acurateţei de clasificare este o măsură mai bună a performanţelor clasificatorului, deoarece acesta este testat şi antrenat pe mai multe submulţimi disjuncte, în cazul metaclasificatorului acest lucru nu se pretează, deoarece seturile de date ar trebui schimbate la nivel de metaclasificator, nu la nivel de clasificator. Acesta este motivul pentru care clasificatorii se antrenează şi se testează pe seturi prestabilite.

În acest moment, putem afirma că cele 3 categorii de clasificatoare - SVM cu nucleu polinomial, SVM cu nucleu gaussian şi Bayes (adaptat - BNA) rulează în acelaşi context. Singura diferenţă între clasificatorii de tip SVM şi cel de tip Bayes este că pentru SVM se vor folosi 24 de clase, iar pentru Bayes doar 16 clase.

6.2.3.2 Compararea clasificatorului Bayes adaptat (BNA) cu clasificatorii de tip SVM

În această secţiune voi prezenta rezultate comparative între clasificatorii de tip SVM şi clasificatorul Bayes adaptat. Ideea este de a determina dacă sunt şanse de îmbunătăţire a acurateţei de clasificare a metaclasificatorului prezentat în [Mora07] în cazul introducerii în metaclasificator şi a unui clasificator de tip Bayes adaptat.

6.2.3.3 Antrenarea clasificatorilor pe setul A1 şi testarea pe setul T1

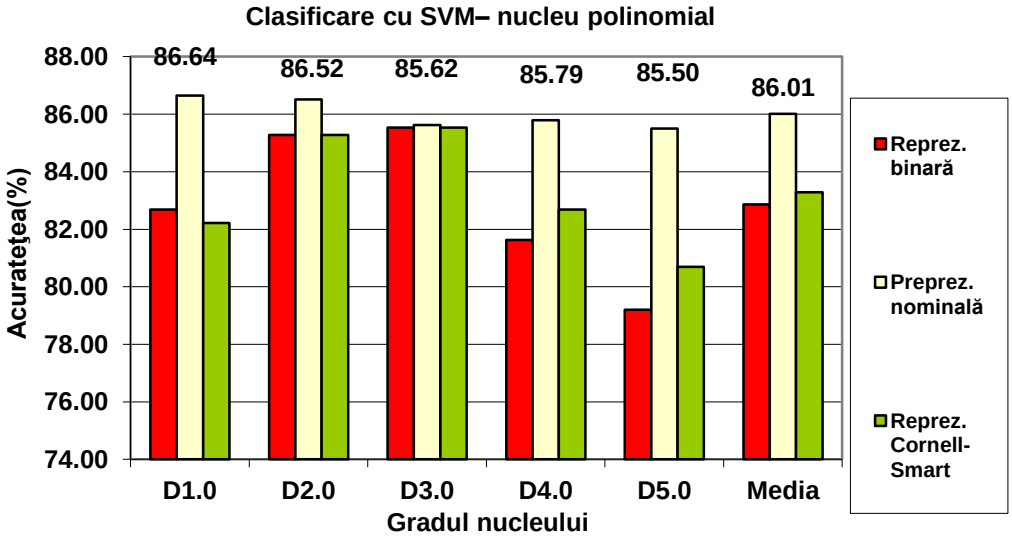


Fig. 6.6 Rezultatele clasificării cu SVM nucleu polinomial (preluat din [Mora07])

Rezultatele prezentate în Fig. 6.6 s-au obţinut de către clasificatorii SVM cu nucleu polinomial, pentru diferite grade ale nucleului şi pentru diferite reprezentări ale vectorilor de documente, antrenaţi pe setul A1 şi testaţi pe setul T1. Graficul din Fig. 6.6 indică faptul că rezultatele cele mai bune (86,64%), s-au obţinut utilizând clasificatorul SVM cu nucleu polinomial de grad 1 şi reprezentare nominală. Ca şi medie a acurateţei de clasificare, acest tip de clasificator a obţinut cel mai bun rezultat pentru reprezentarea nominală (86,01%).

Singurul parametru care ne permite reglarea acurateţei de clasificare la clasificatorul Bayes este procentajul de documente de antrenament (după cum s-a arătat în Fig 5.1). În cazul acesta, setul de antrenare respectiv de testare sunt prestabilite. Din acest motiv, avem doar un singur rezultat pentru Bayes, rezultat pe care îl comparăm atât cu cel mai bun rezultat obţinut de SVM polinomial, cât şi cu media obţinută de acesta. După cum se poate observa în Fig. 6.7 cu clasificatorul Bayes antrenat pe setul A1 şi testat pe setul T1, fiecare având 1.309 atribute, s-a obţinut o acurateţe a clasificării de 81,32%. Trebuie specificat că clasificatorul Bayes foloseşte doar metoda nominală de reprezentare a vectorului de documente.

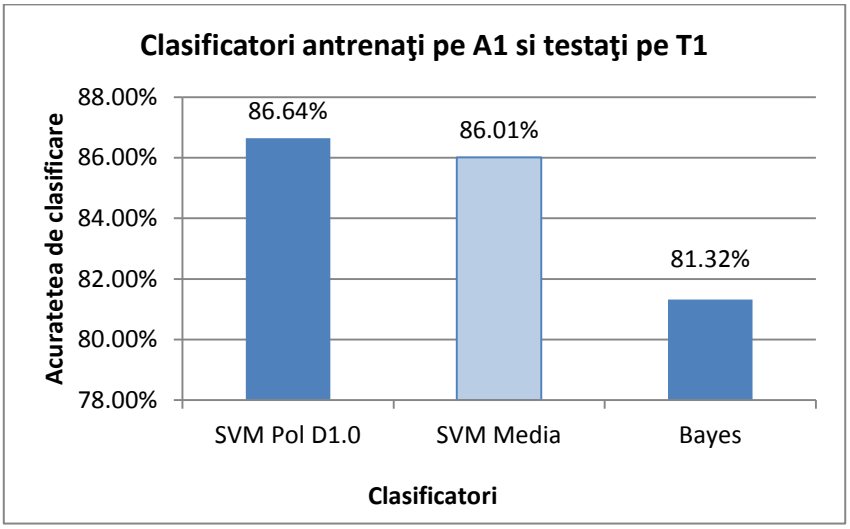


Fig. 6.7 Compararea rezultatelor obţinute cu clasificatori SVM şi Bayes

În Fig.6.8 se prezintă timpii de antrenare obţinuţi de fiecare clasificator în parte. Timpii de antrenare sunt obţinuţi pe un calculator P IV la f=3.2GHz cu 1GB memorie DRAM.

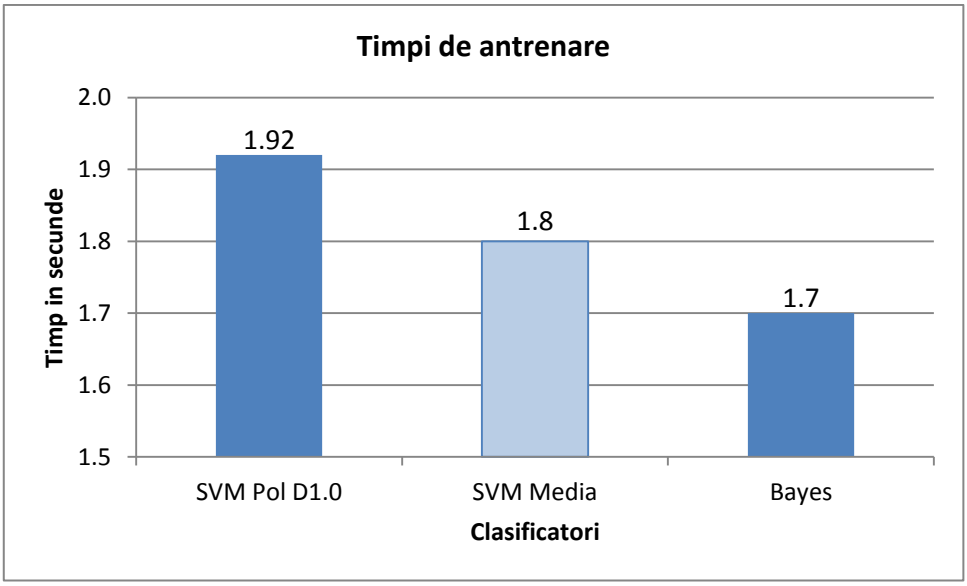


Fig. 6.8 Compararea timpilor de antrenare obţinuţi cu clasificatori SVM şi Bayes

Ca şi timpi de antrenare, clasificatorul Bayes oferă timpi mai mici de antrenare, deoarece calculează doar rapoarte între atribute şi clase.

6.2.3.4 Antrenarea pe setul A1 şi testarea pe setul T2

Reamintesc faptul că în setul de test T2 avem doar acele documente care nu au putut fi clasificate corect de către nici un clasificator SVM din cadrul metaclasificatorului prezentat în [Mora07]. Antrenarea pentru fiecare clasificator s-a făcut pe setul A1.

După cum s-a observat din Figurile 6.3 (SVM polinomial) şi 6.2 (SVM Gaussian) clasificatorii de tip SVM obţin rezultate diferite de 0 pe acest set (T2), dar cu alţi parametri de intrare decât cei folosiţi în metaclasificator. În Fig. 6.9 prezint cele mai bune rezultate obţinute de clasificatorul SVM polinomial şi SVM Gaussian pe acest set, precum şi mediile obţinute pentru toate testele (cele din Figurile 6.2 şi 6.3). Pe ultima bară se prezintă rezultatul obţinut de clasificatorul Bayes pe acest set de test. În [Mora07], „regula" de selecţie a clasificatorilor care au fost introduşi în metaclasificator a fost să obţină cele mai bune rezultate pe setul T1 (cu 2.351 vectori de documente). După cum s-a observat şi din Figurile 6.2 şi 6.3, există clasificatori SVM pentru care documentele din setul T2 (cel cu 136 vectori de documente) se clasifică mai bine (oricum, nemulţumitor), dar aceştia au obţinut rezultate nesatisfăcătoare pe setul mare (T1)(de exemplu: *SVM polinomial, grad 1, reprezentarea Cornell SMART* -80.99%, *SVM polinomial, grad 1, reprezentarea Binară* -81.45%, *SVM polinomial, grad 3, reprezentarea BIN* -85.79% - din Tabel 6.1 [Mora07]) şi de aceea nu au fost selectaţi în metaclasificator. Introducerea în metaclasificator a unui alt clasificator SVM care clasifică ceva mai bine setul T2 (de exemplu cel polinomial de grad 1 cu reprezentare Cornell Smart), alături de cele selectate în [Mora07], care obţin 0% pe acest set, ar modifica limita maximă la care poate ajunge metaclasificatorul (în sus sau în jos).

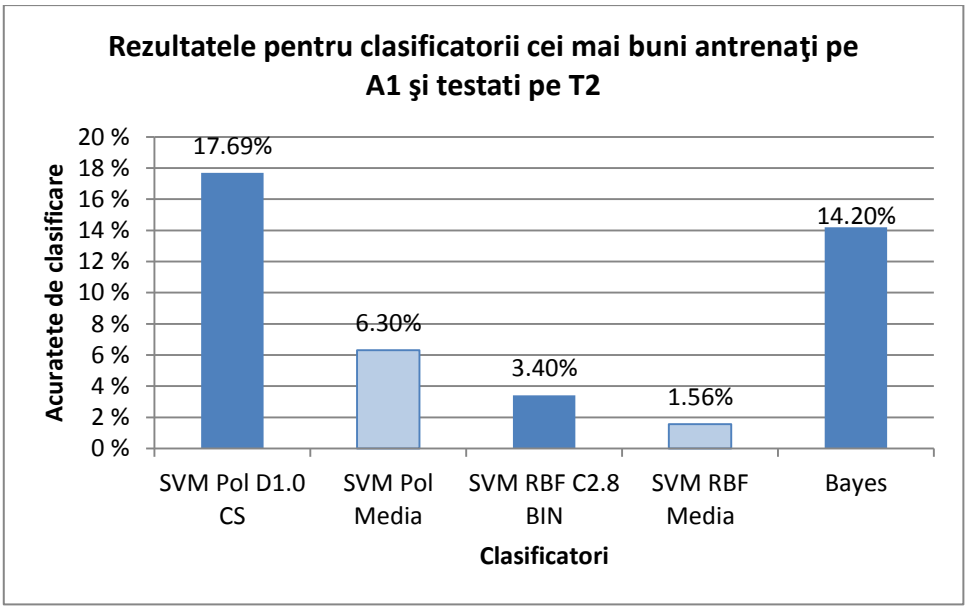


Fig. 6.9 Rezultate obţinute de clasificatorii SVM şi Bayes în clasificarea setului T2 şi antrenat pe A1

În urma testelor efectuate am observat că, deşi clasificatorului Bayes (BNA) generează rezultate mai slabe din punct de vedere al acurateţei totale decât SVM; el totuşi clasifică în mod corect 104 documente din cele 136 documente cu probleme din setul T2; chiar dacă este antrenat pe setul A1, iar acest fapt ar creşte limita maximă obtenabilă la valoarea 98.63%.

6.2.3.5 Antrenarea şi testarea pe setul T2

În Fig. 6.4 şi 6.5 am prezentat rezultatele obţinute de clasificatorii de tip SVM cu nucleu gaussian şi SVM cu nucleu polinomial în cazul în care s-au antrenat pe setul T2 (cu doar 136 documente) şi s-au testat pe acelaşi set. În acest caz artificios clasificatorii SVM de tip polinomial au reuşit o acurateţe a clasificării chiar de 100%, în medie ei obţinând o valoare de

98.05%. Clasificatorul Bayes a obţinut o acurateţe a clasificării de 97.95%, puţin mai mică faţă de SVM. Ceea ce ne interesează este dacă acest clasificator va reuşi în metaclasificator să clasifice corect documentele din setul T2.

În urma rezultatelor favorabile din punct de vedere al setului T2 (a se vedea secţiunea 5.2.4), am decis includerea clasificatorului de tip Bayes în metaclasificatorul prezentat în [Mora07]. Astfel pe lângă cei 8 clasificatori selectaţi, am mai introdus unul (Bayes), metaclasificatorul având acum nouă clasificatori.

Acum metaclasificatorul conţine 8 clasificatori SVM şi un clasificator Bayes, astfel:

NR. CRT.	TIP CLASIFICATOR	PARAMETRUL NUCLEULUI	REPREZENTAREA DATELOR DE INTRARE	ACURATEŢEA OBŢINUTĂ (%)
1	SVM-Polinomial	1	Nominal	86.69
2	SVM-Polinomial	2	Binary	86.64
3	SVM-Polinomial	2	Cornell Smart	87.11
4	SVM-Polinomial	3	Cornell Smart	86.51
5	SVM-Gaussian	1.8	Cornell Smart	84.30
6	SVM-Gaussian	2.1	Cornell Smart	83.83
7	SVM-Gaussian	2.8	Cornell Smart	83.66
8	SVM-Gaussian	3.0	Cornell Smart	83.41
9	Bayes	-	Nominal	81.32

Tabel 6.2 Clasificatorii incluşi în metaclasificator

De asemenea, am calculat limita maximă la care ar putea ajunge noul metaclasificator. Astfel, în urma introducerii clasificatorului Bayes, limita maximă a metaclasificatorului creşte la **98.63%** (faţă de 94.21% cât era fără Bayes), ceea ce oferă o posibilitatea obţinerii unei acurateţi reale a clasificării mult mai bune decât cea obţinută în [Mora07].

6.3 *Metode de selecţie a clasificatorilor*

6.3.1 **Selecţia bazată pe vot majoritar (MV). Rezultate**

Ideea acestei metode este de a utiliza toţi clasificatorii din metaclasificator pentru a clasifica documentul curent. Fiecare clasificator va specifica pentru documentul curent o anumită clasă căreia îi acord încrederea maximă. Sarcina metaclasificatorului va fi să aleagă clasa învingătoare pe baza categoriilor propuse de fiecare clasificator. Practic metaclasificatorul va contoriza pentru fiecare clasă voturile clasificatorilor. Clasa care obţine cele mai multe voturi este declarată clasă învingătoare. pentru o anumită categorie pentru documentul curent. Metaclasificatorul va păstra pentru Dacă două clase obţin acelaşi punctaj atunci metaclasificatorul va alege ambele clase ca şi clase învingătoare. Problema majoră a acestui metaclasificator este faptul că este un model neadaptiv, neavând şi un pas în care rezultatul final să influenţeze apoi funcţionarea metaclasificatorului în continuare

Utilizând votul majoritar acurateţea clasificării obţinute este de 86.09%. În cazul introducerii unui nou clasificator în metaclasificator, acurateţea clasificării a scăzut faţă de valoarea obţinută cu 8 clasificatori (86.38%), având deci o scădere de 0.29%. Aceasta poate apărea deoarece pe tot setul de test clasificatorul Bayes obţine o acurateţe de doar 81.32%, clasificând destul de multe documente (439) incorect, ceea ce se pare că „ajută”, în metaclasificator, la selectarea în 7 cazuri a unor categorii greşite, deoarece Bayes a întărit votul greşit.

6.3.2 **Selecţia bazată pe distanţa euclidiană (SBED). Rezultate**

În [Mora07] a fost dezvoltat un metaclasificator adaptiv care îşi modifică modul de funcţionare încercând să se adapteze la datele de intrare. Sarcina acestui tip de metaclasificator este de a alege un clasificator căruia metaclasificatorul îi acordă încrederea de a clasifica documentul curent. Rezultatul întors de clasificatorul ales va fi rezultatul întors de metaclasificator. Practic metaclasificatorul are o metodă euristică de a învăţa în funcţie de datele de intrare şi se bazează de fapt pe exemplele greşit clasificate. Astfel fiecare din cei 9 clasificatori din metaclasificator va avea o coadă proprie în care se salvează documentele greşit clasificate de acel clasificator.

Modul de funcţionare al metaclasificatorului este următorul: când un document nou este prezentat metaclasificatorului acesta va alege aleator un clasificator care îl va clasifica. Rezultatul returnat de către clasificator care este şi totodată rezultatul furnizat de metaclasificator va fi comparat cu clasa propusă de clasificarea Reuters. În cazul în care clasele coincid se trece la următorul document. Dacă clasele nu coincid documentul va fi salvat în coada clasificatorului care l-a clasificat greşit. Pentru a evita utilizarea aceluiaşi clasificator pentru un document asemănător se va calcula distanţa euclidiană (prezentată iniţial în ecuaţia 2.11, şi adaptată în ecuaţia 6.1) dintre documentul curent şi toate documentele din coada clasificatorului selectat. În cazul în care se obţine o distanţă mai mică decât un prag prestabilit înseamnă că acel clasificator clasificând greşit un exemplu asemănător, va fi rejectat. Altfel va fi folosit în clasificare documentului curent. Se întâmplă ca toţi clasificatorii să fie rejectaţi şi atunci totuşi metaclasificatorul va alege un clasificator pentru care distanţa este maximă chiar dacă aceasta este mai mică decât pragul stabilit. În acest caz probabilitatea ca metaclasificatorul să clasifice greşit este mare.

$$Eucl(x, x') = \sqrt{\sum_{i=1}^{n} (x_i - x_i')^2} \tag{6.1}$$

unde x_i este valoarea de la poziţia i a vectorului x, iar x şi x' sunt vectorii de intrare, unul fiind documentul curent iar celălalt fiind vectorul din coada clasificatorului.

Funcţionarea metaclasificatorului se face în două etape. În prima etapă se realizează de fapt etapa de învăţare în care metaclasificatorul primeşte informaţii asupra calităţii clasificării punând exemplele greşit clasificate în coada clasificatorilor respectivi. Această etapă se realizează cu ajutorul setului de antrenament. În etapa de testare se verifică acurateţea de clasificare iar caracteristicile metaclasificatorului nu se mai schimbă. Având în vedere faptul că după fiecare pas de antrenare metaclasificatorul îşi modifică caracteristicile, în experimente cei doi paşi au fost repetaţi.

Prezint în tabelul 6.3 rezultatele obţinute de noul metaclasificator cu 9 clasificatori comparativ cu metaclasificatorul din [Mora07]. În tabelul 6.3 se prezintă doar primii 14 paşi deoarece, după acest număr de paşi, am observat că acurateţea clasificării nu se mai modifică substanţial. La fel ca în [Mora07], pragul pentru primii 7 paşi s-a ales egal cu 2,5 şi pragul pentru ultimii 7 paşi s-a ales egal cu 1,5.

Pas	Distanţă euclidiană SBED -8 clasificatori	Distanţă euclidiană SBED -9 clasificatori
1	84.77	83.79
2	85.71	85.07
3	86.35	83.28
4	86.60	83.54
5	86.81	85.67
6	87.32	84.43
7	88.43	82.01
8	89.83	87.75

9	90.05	88.60
10	90.56	88.73
11	91.24	88.35
12	91.58	88.60
13	**92.05**	89.88
14	**92.05**	90.39

Tabelul 6.3 Rezultate obţinute de metaclasificator utilizând distanţa euclidiană

În Fig. 6.10, prezint grafic aceste rezultate şi rezultatele obţinute cu metoda vot majoritar atât pentru metaclasificatorul cu 8 clasificatoare cât şi pentru cel nou.

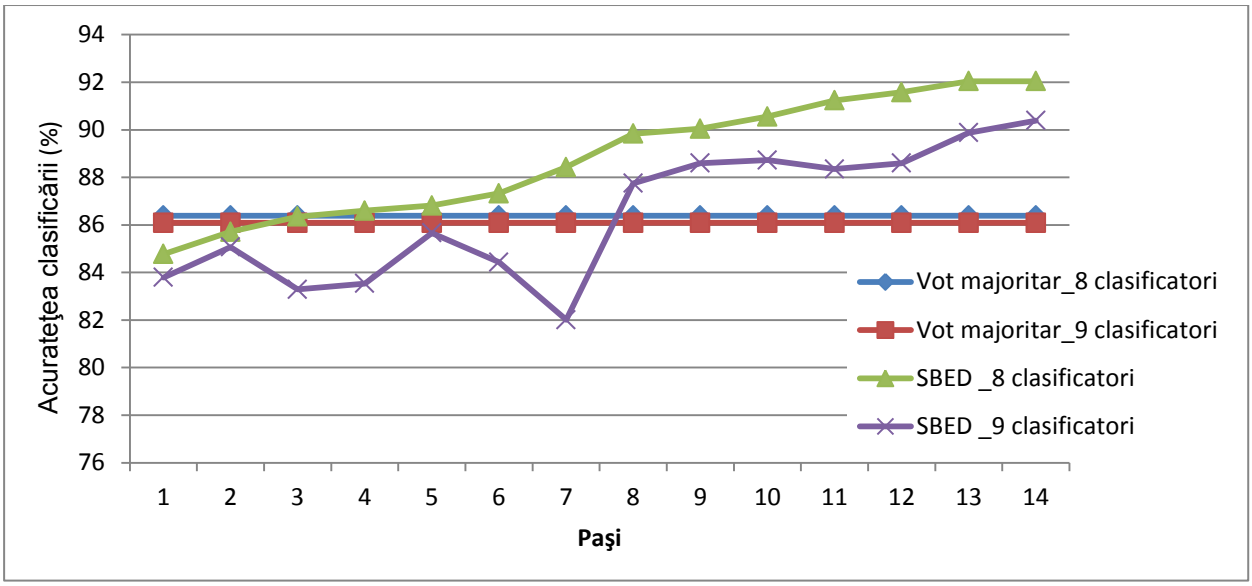


Fig. 6.10 Acurateţea clasificării - vot majoritar şi distanţă euclidiană (metaclasificator cu 8 şi 9 clasificatori)

În cazul metaclasificatorului cu 9 clasificatoare, rezultatele (90.38%) obţinute sunt mai slabe decât cele realizate de metaclasificatorul cu 8 clasificatoare (92.05%). Se pare că acurateţea de clasificare proastă a clasificatorul Bayes (81.32%) comparativ cu cel SVM, reduce, în acest caz, acurateţea globală de clasificare a metaclasificatorului.

Observăm că acurateţea clasificării în cazul metaclasificatorul cu 9 clasificatoare are şi tendinţe descrescătoare. Aceasta se poate datora faptului că un clasificator poate clasifica corect un document d_1 şi clasifica incorect un alt document d_2, apropiat ca distanţă de documentul d_1. Din acest motiv, la o nouă parcurgere a setului de test, clasificatorul respectiv nu a mai fost selectat pentru clasificarea lui d_1 (deoarece a dat rezultate proaste pentru d_2), atunci căutându-se un alt clasificator (care poate, la rândul său, să clasifice greşit).

6.3.3 Selecţia bazată pe distanţa cosinus (SBCOS). Rezultate

Metaclasificatorul care este prezentat în această secţiune este asemănător în funcţionare celui prezentat în secţiunea 6.3.2 singura modificare fiind modul de calculare a similarităţii dintre documentul curent şi cele din coada clasificatorului ales. Astfel acest metaclasificator foloseşte metrica cosinus. Reamintim formula (prezentată în ecuaţia 2.14) utilizată în calculul cosinusului unghiului θ dintre doi vectori x şi x' este:

$$\cos\theta = \frac{\langle x, x' \rangle}{\|x\| \cdot \|x'\|} = \frac{\sum_{i=1}^{n} x_i x'_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} x'^2_i}} \qquad (6.2)$$

unde x_i este valoarea de la poziţia i a vectorului x iar x şi x' sunt vectori de intrare (documentele).

Acest metaclasificator va accepta clasificatorul ales dacă toate valorile obţinute pentru cosinus calculate între documentul curent şi documentele din coada clasificatorului respectiv sunt mai mici decât un prag prestabilit.

Prezint în tabelul 6.4 rezultatele obţinute de noul metaclasificator cu 9 clasificatori, comparativ cu metaclasificatorul cu 8 clasificatori. Se prezintă doar primii 14 paşi, deoarece după acest număr de paşi, acurateţea clasificării nu se mai modifică substanţial. La fel ca în [Mora07], pragul pentru primii 7 paşi s-a ales egal cu 0.8 iar pragul pentru ultimii 7 paşi s-a ales egal cu 0.9.

Pas	Distanţă cosinus SBCOS_ 8 clasificatori	Distanţă cosinus SBCOS_ 9 clasificatori
1	85.33	90.98
2	85.33	91.03
3	86.60	91.20
4	86.09	92.00
5	86.47	92.64
6	87.32	92.43
7	87.66	92.47
8	88.09	92.64
9	88.30	92.90
10	89.66	92.47
11	88.90	92.34
12	89.58	92.68
13	89.24	**93.11**
14	89.75	**93.11**

Tabelul 6.4 Rezultate obţinute de metaclasificator utilizând distanţa cosinus

De asemenea prezint grafic, în Fig. 6.11, rezultatele la care am adăugat şi limita maximă obtenabilă a noului metaclasificator.

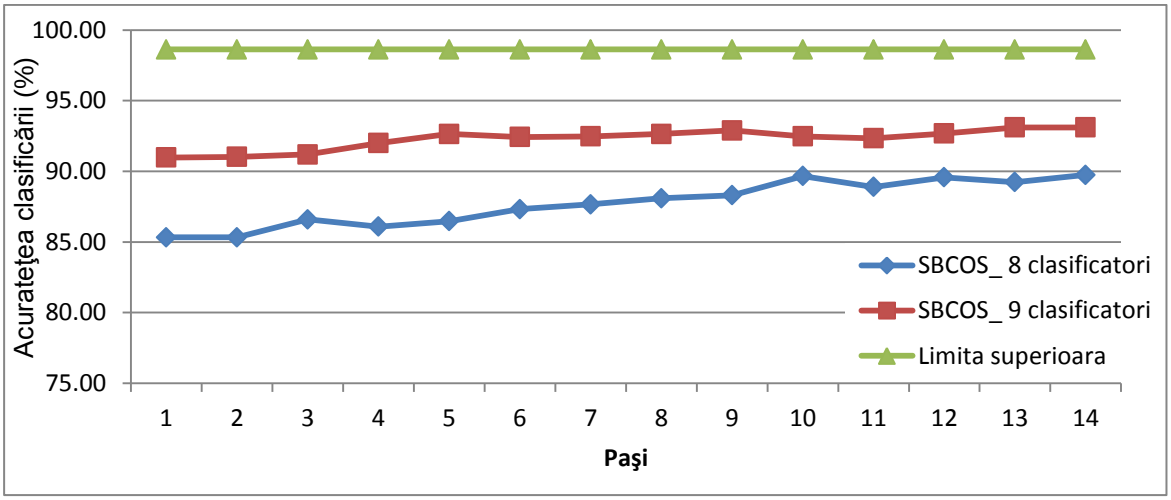


Fig. 6.11 Rezultatele obţinute cu metaclasificatorul cu 8 şi cu 9 clasificatori utilizând cosinusul

În cazul acestui metaclasificator, rezultatele obţinute arată că acurateţea de clasificare s-a îmbunătăţit de la 89.74% la **93.11%**, în momentul adăugării clasificatorului Bayes (BNA).

6.4 *Arhitecturi neadaptive propuse şi dezvoltate*

6.4.1 Metaclasificator cu ponderi predefinite. Evaluare de tip Eurovision. Rezultate obţinute.

Metaclasificatorul propus în continuare conţine cei 9 clasificatori utilizaţi în secţiunea anterioară şi pleacă de la premisa că ar conta şi numărul şi locul pe care apare fiecare clasă în parte. De exemplu, în cazul a doi clasificatori şi 3 clase, dacă o clasă apare o dată pe locul 1 şi o dată pe locul 4 şi o altă clasă apare de 2 ori pe locul 2, este posibil ca cea de-a doua clasă să fie mai valoroasă, chiar dacă nu a obţinut niciodată locul 1.

6.4.1.1 Metaclasificator neadaptiv bazat pe sumă

În metaclasificator sunt incluşi 8 clasificatori de tip SVM şi unul de tip Bayes. Fiecare dintre aceştia produce un vector care conţine 16 scalari, vezi Fig. 6.12. Fiecare scalar reprezintă valoarea funcţiei de decizie a clasificatorului pentru clasa respectivă. Până acum, în [Mora08] se alegea întotdeauna valoarea cea mai mare şi clasa corespunzătoare a acestei valori era considerată clasa pe care o prezice clasificatorul respectiv. Pentru fiecare document în parte vom obţine acum 9 astfel de vectori fiecare cu 16 valori.

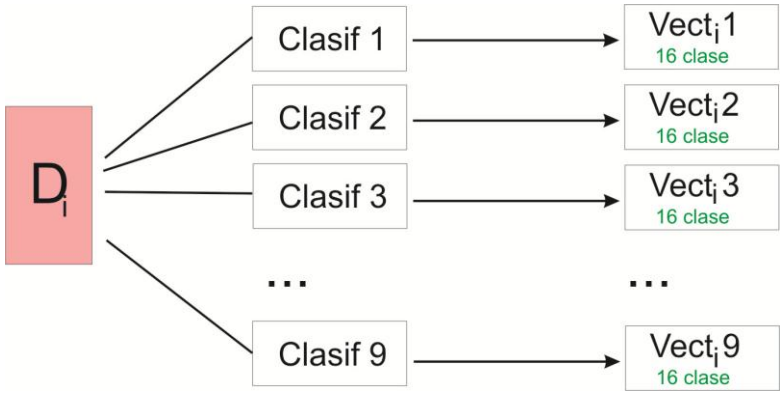


Fig. 6.12 Metaclasificator neadaptiv bazat pe sumă

Valorile funcţiilor de decizie pentru clasificatorii de tip SVM se află în intervalul (-∞,∞) dar în apropierea valorii 0, iar pentru clasificatorul de tip Bayes valorile se află în intervalul (-∞, 0). Având în vedere aceste diferenţe şi pentru a putea realiza însumarea valorilor vectorilor, am transpus valorile vectorilor în intervalul [1, ∞).

$$V_i = V_i + \left|\min(\vec{V})\right| + 1 \qquad (6.3)$$

Astfel, pentru fiecare vector, diferenţele între valorile vectorilor de ieşire ai clasificatorilor de tip SVM se păstrează. La fel şi pentru clasificatorul de tip Bayes. Pentru a putea realiza însumarea acestor vectori, în următorul pas am normalizat vectorii, aducând valorile acestora în intervalul (0,1].

$$V_i = \frac{V_i}{\max(\vec{V})} \qquad (6.4)$$

În cazul metaclasificatorului care realizează doar sumele (numit in continuare M-SUM), am însumat cele 16 valori ale acestor 9 vectori, vezi Fig. 6.13, clasa câştigătoare fiind clasa cu valoarea cea mai mare obţinută.

$$Class = \max_{c_i, i=1,16} \sum_{k=1}^{9} V_i[k] \qquad (6.5)$$

Acest metaclasificator, fiind unul neadaptiv, va obţine întotdeauna acelaşi rezultat pentru o anumită instanţă de intrare. În cazul rulării pe cele 2351 documente de test (setul T1), am obţinut un număr de **313** documente clasificate eronat, care reprezintă o acurateţe a clasificării de **86.68%,** cu **0.59%** mai mare decât valoarea obţinută, folosind votul majoritar şi toţi cei 9 clasificatori. Astfel putem concluziona că metoda bazată pe luarea în considerare doar a clasei învingătoare (*majority vote*), are dezavantaje faţă de metoda prezentată mai sus. În acest caz există şansa ca o clasă care poate niciodată nu a obţinut locul 1, dar a obţinut valori apropiate de maxim, să fie în final clasa corectă.

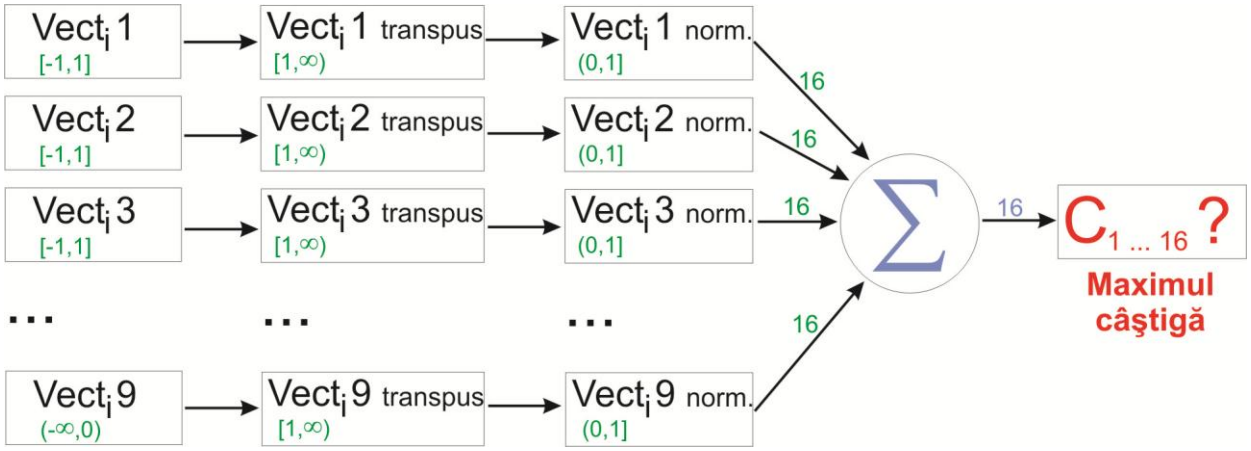


Fig. 6.13 Metaclasificator neadaptiv bazat pe sumă (M-SUM)

6.4.1.2 Metaclasificator neadaptiv bazat pe sumă normalizată

Această metodă are la bază metoda prezentată în secţiunea 6.4.1.1, doar că, înainte de însumarea celor 16 valori din cei 9 vectori, se realizează o modificare a acestor valori. Astfel, în cazul acestei ponderări vom asigna în fiecare vector pentru clasa de pe locul 1, valoarea 12, pentru clasa de pe locul 2, valoarea 10, iar în continuare, pentru fiecare clasă de pe următoarele locuri, valori descrescătoare până la valoarea 1 (abordarea implementează soluţia consacrată în concursul european de muzică uşoară EUROVISION). Astfel, în fiecare vector clasele de pe primele 11 locuri vor avea valori diferite iar celelalte clase, până la 16, vor avea valoarea 1. Domeniul de reprezentare a valorilor vectorilor este {1, 2, 3, ...,10,12}, vezi Fig. 6.14. După această etapă se va realiza însumarea vectorilor şi selectarea clasei care obţine valoarea cea mai mare, analog cu metoda anterior prezentată în secţiunea 6.4.1.1.

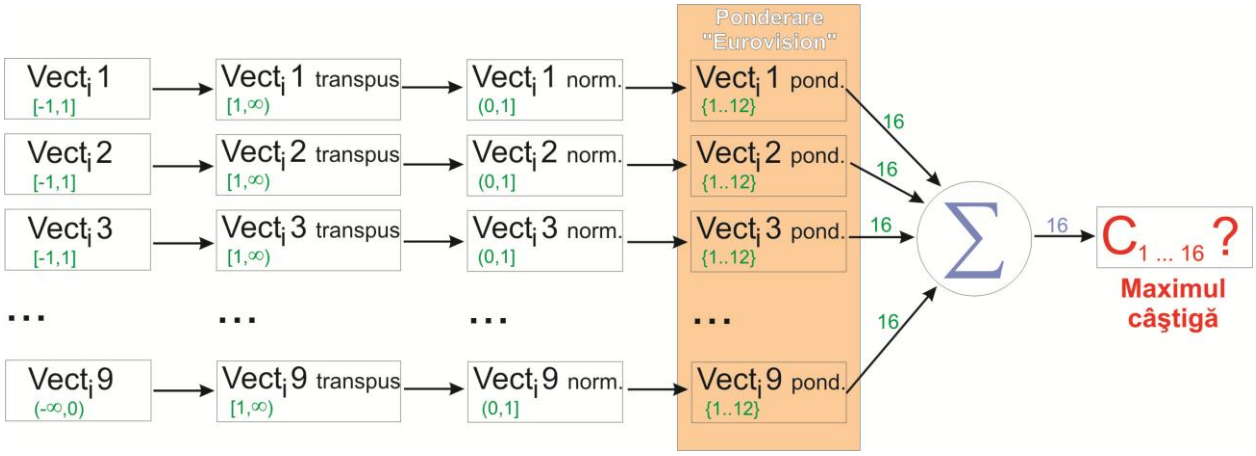


Fig. 6.14 Metaclasificator neadaptiv bazat pe sumă ponderată

În urma acestei ponderări am obţinut un număr de 316 erori de clasificare, ceea ce reprezintă o acurateţe a clasificării pe setul T1 de **86,55%** pentru această metodă. Rezultatele obţinute sunt cu 0,12% mai slabe decât cele obţinute direct pe sumă.

6.4.1.3 Metaclasificator neadaptiv bazat pe sumă ponderată

În această secţiune introducem un nou metaclasificator neadaptiv bazat pe sumă ponderata (notat în continuare M-SPON). Pentru acest metaclasificator, în pasul în care se realizează ponderarea, am decis ca în fiecare vector reprezentat ca în secţiunea 6.4.1, pentru clasa de pe primul loc, noua valoare să fie vechea valoare **înmulţită** cu 12. Pentru clasa de pe locul 2 va fi vechea valoare înmulţită cu 10, pentru locul 3 valoarea veche înmulţită cu 9 ş.a.m.d. pentru toate celelalte clase; valoarea minimă cu care înmulţim va fi 1. În cazul acesta, domeniul de reprezentare al valorilor vectorilor este (0,12], vezi Fig. 6.15. Ca şi mai sus, în continuare se vor însuma valorile celor 9 vectori şi se va alege ca şi clasă învingătoare clasa care obţine valoare maximă.

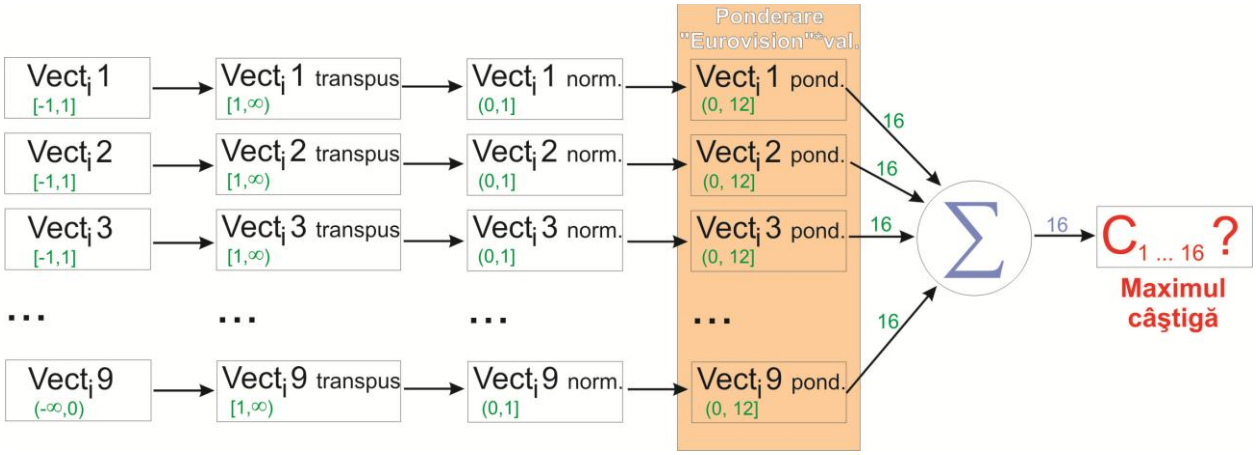


Fig. 6.15 - Metaclasificator neadaptiv bazat pe sumă ponderată

În acest caz am obţinut un număr de **305** documente clasificate eronat pe setul T1, acurateţea clasificării pentru acest metaclasificator fiind de **87.02%.** Această acurateţe de clasificare obţinută este cea mai mare obţinută prin utilizarea unui metaclasificator neadaptiv, dar evident mai mică decât limita maximă de 98.63%, la care poate ajunge teoretic metaclasificatorul.

6.4.1.4 Cercetări privind alte variante de ponderare a elementelor vectorilor

În această secţiune prezint câteva experimente efectuate asupra valorilor de ponderare pentru elementele celor 9 vectori rezultaţi în urma folosirii celor 9 clasificatori. Aceste valori reprezintă ponderea care este înmulţită cu valoarea obţinută de o clasă în cadrul clasificatorului, înainte de a realiza însumarea celor 9 rezultate.

6.4.1.4.1 Înjumătăţirea ponderii

În primul experiment am ales ca valoarea de ponderare să se înjumătăţească pentru fiecare clasă, clasele fiind în prealabil ordonate descrescător. Astfel, valoarea clasei de pe prima poziţie se înmulţeşte cu constanta „16”, valoarea clasei de pe a doua poziţie se înmulţeşte cu constanta „8”, cea de pe a treia poziţie cu constanta „4” şi aşa mai departe, valorile claselor de pe ultimele 12 poziţii se înmulţesc cu constanta „1”. În acest caz doar clasele de pe primele 4 poziţii vor avea ponderi distincte, celelalte rămânând cu valoarea iniţială. Ideea este de a favoriza foarte mult primele locuri. Rezultatul obţinut de metaclasificator este de 324 documente incorect clasificate,

ceea ce reprezintă o acurateţe a clasificării de 86.22%. Conform rezultatelor obţinute, se observă că clasa corectă nu este întotdeauna prima clasă returnată de fiecare clasificator în parte. Această concluzie am formulat-o şi în cazul votului majoritar care a obţinut o acurateţe de clasificare de 86.09%, adică 327 documente clasificate incorect.

6.4.1.4.2 Ponderi mici descrescătoare linear

Pas 0.1

În acest experiment, pentru ponderi am ales valori mici iar diferenţa dintre ele să fie de 0.1. Astfel, ponderea valorii clasei celei cu probabilitatea cea mai mare va fi 2.5 iar a celei cu probabilitate mai mică va fi 1. Ideea este de a nu face o diferenţă foarte mare între clasele de pe diferite poziţii, dar totuşi să favorizăm puţin clasa situată pe o poziţie superioară. În acest caz metaclasificatorul a avut un număr de 304 documente clasificate incorect ajungându-se astfel la o acurateţe de clasificare de 87.07%. Alegerea ponderării distincte pentru fiecare loc cu valori apropiate este benefică în acest context.

Pas 1.0

De aceea, în următorul experiment am ponderat clasele cu valori descrescătoare distincte cu pasul 1. Astfel, pentru prima poziţie valoarea ponderii este16, pentru a doua poziţie valoarea ponderii este 15 ş.a.m.d. până la ultima poziţie, la care valoarea ponderii este 1. În acest caz, numărul de documente incorect clasificate de către metaclasificator a scăzut la 303, ceea ce reprezintă o acurateţe a clasificării de 87.11%.

Pas 0.5

Totuşi, cele mai bune rezultate le-am obţinut în cazul în care valorile ponderilor scad liniar cu un pas egal cu valoarea 0,5. Valoarea de prima poziţie va fi ponderată cu valoarea 8.5 ş.a.m.d., descrescător până la ultima poziţie, unde valoare ponderii este 1.0. Astfel, numărul de documente incorect clasificate de către metaclasificator a scăzut la 301, rezultând o acurateţe a clasificării de **87.20%.**

În Fig. 6.16 prezint comparativ rezultatele obţinute în secţiunea 6.3.1.3.

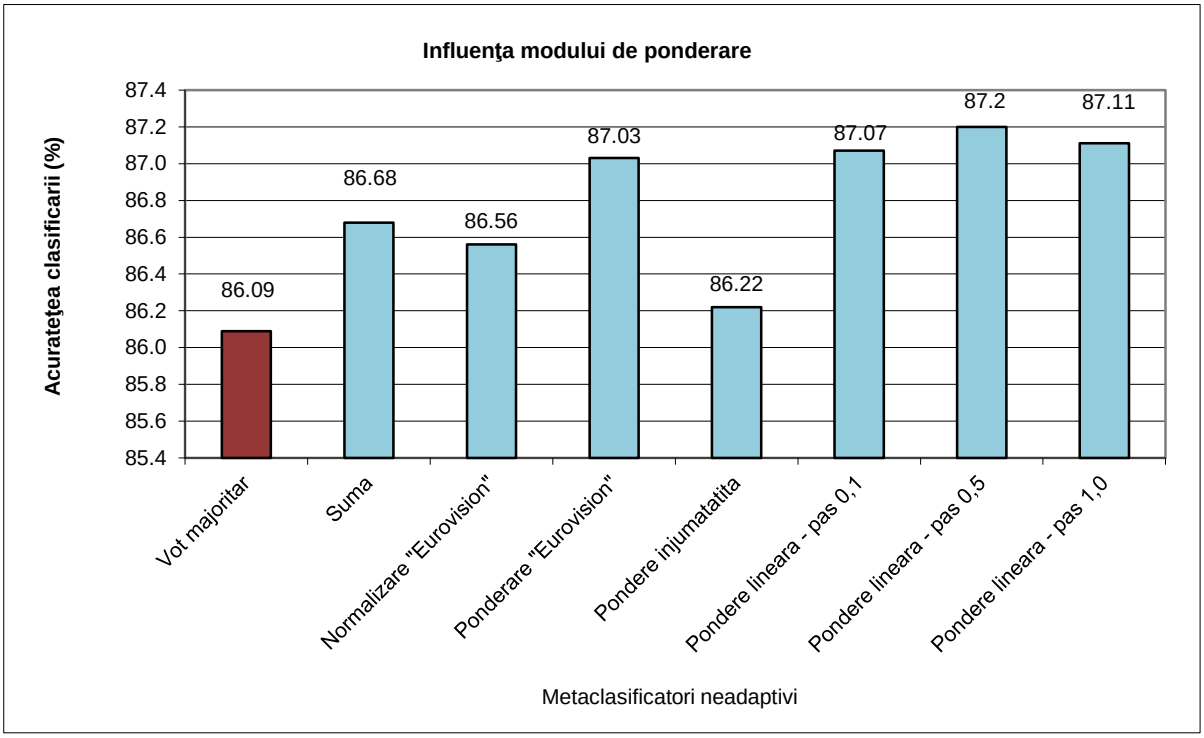


Fig. 6.16 - Comparaţie rezultate metaclasificatori neadaptivi

Rezultatele prezentate în această secţiune au fost de asemenea publicate în [Cret09_a].

6.4.2 Metaclasificator cu ponderi calculate. Design Space Exploration cu algoritmi genetici. Rezultate obţinute.

În această secţiune voi introduce un nou metaclasificator neadaptiv (dar cu pas de învăţare), bazat pe suma ponderată (îl voi numi în continuare M-WSUM). Acest metaclasificator se bazează tot pe 8 clasificatori de tip SVM şi pe unul de tip Bayes. În acest metaclasificator, valorile rezultate corespunzătoare pentru clasele calculate vor fi înmulţite cu o valoare numită pondere. În secţiunea 6.3.1 am prezentat câţiva metaclasificatori asemănători, folosind diverse valori pentru ponderarea poziţiilor pentru clase. Întrucât alegerea valorilor optime pentru ponderarea fiecărei poziţii de clasă este o sarcină dificilă, în această secţiune prezint o metodă bazată pe un algoritm genetic de calcul a ponderilor. Pornind de la setul de date de test, am creat un fişier de testare adecvat, care este folosit pentru algoritmul genetic. Astfel, pentru fiecare document din setul de testare am salvat toate ieşirile din toţi clasificatorii separat. Astfel, pentru un document, am obţinut 9 vectori de ieşire (pentru că avem 9 clasificatori), fiecare vector cu un număr de 16 scalari (pentru ca avem 16 clase). Elementele fiecăruia dintre cei 9 vectori sunt în prealabil ordonate descrescător. În algoritmul genetic folosit în experimente, un cromozom reprezintă valoarea ponderilor pentru fiecare clasă separat, în funcţie de poziţia clasei. Prima valoare din vector reprezintă ponderea clasei de pe prima poziţie din clasificator, a doua valoare din vector reprezintă ponderea clasei de pe poziţia următoare, şi aşa mai departe. Astfel cromozomul va avea forma:

$$c = \left(w_1, w_2, ..., w_j\right), \text{ cu } j = \overline{1,16} \tag{6.7}$$

unde w_j reprezintă ponderea pentru fiecare poziţie a clasei, returnată de fiecare clasificator utilizat în metaclasificator. Setul de antrenament este de forma $\left\{\left\langle \vec{x}_{ij}, y\right\rangle, i = \overline{1,m} \text{ and } j = \overline{1,n}\right\}$, unde y reprezintă clasa corectă pentru documentul $\vec{x}_{ij}$, n reprezintă numărul claselor iar m reprezintă numărul clasificatoarelor utilizate în metaclasificator. Pentru algoritmul genetic, setul de antrenament este alcătuit din ieşirile tuturor clasificatorilor utilizaţi în metaclasificator. Pentru a simplifica calcularea valorii funcţiei de fitness, voi folosi o reprezentare a cromozomului care utilizează toate ieşirile clasificatorilor utilizaţi în metaclasificator. Prin această abordare se va ţine cont şi de poziţia fiecărei clase în ieşirea fiecărui clasificator. În cromozom sunt salvate, pentru fiecare poziţie a clasei, ponderile care se vor utiliza în calcularea acurateţei metaclasificatorului. Astfel, decizia metaclasificatorului include şi ponderile pentru fiecare clasă.

În fiecare experiment s-a pornit cu o generaţie de 100 de cromozomi, fiecare dintre ei având iniţial valori aleatoare cuprinse în intervalul [-1, 1]. Pentru a genera o populaţie nouă, am folosit operatorii genetici ca selecţia, crossover şi mutaţia, descrişi în secţiunea 5.4.3.

Procesul se opreşte după un număr predefinit de 1000 de generaţii, sau dacă nu apar modificări ale celui mai bun cromozom în ultimele 20 de generaţii.

Pentru fiecare cromozom, funcţia de fitness se va calcula ca fiind acurateţea clasificării obţinută de metaclasificator pe setul de test.

Deoarece ordinea claselor este importantă, în cromozom am respectat condiţia $w_1 > w_2 > ... w_{16}$ pentru formula (6.7). Pentru calcularea funcţiei de fitness, am efectuat următorii paşi:

1. pentru fiecare document din setul de test, algoritmul obţine un vector de ieşire corespunzător pentru fiecare clasificator inclus în metaclasificator;
2. fiecare vector de ieşire este ponderat cu valorile corespunzătoare din cromozom;
3. clasa învingătoare se stabileşte însumând rezultatele pentru fiecare clasă în parte şi alegând-o pe cea cu valoare maximă conform formulei (6.8).

$$Class_j = \sum_{i=1}^{9} w_j c_{ij}, \text{ for } j = \overline{1,16} \quad si \quad WinClass = \arg\max_{j=\overline{1,16}}(Class_j) \tag{6.8}$$

unde w_j reprezintă valoarea cromozomului pentru ponderea clasei situată pe poziţia j, c_{ij} reprezintă valoarea clasei calculată de clasificatorul i de pe poziţia j, $Class_j$ reprezintă valoarea calculată pentru clasa de pe poziţia j.

4. clasa declarată învingătoare (WinClass) de către metaclasificator va fi comparată cu clasa reală propusă de Reuters iar dacă clasele pentru acelaşi document sunt identice, atunci am considerat că documentul a fost clasificat corect;
5. după procesarea tuturor documentelor, valoarea funcţiei de fitness se calculează ca fiind acurateţea obţinută pe setul de testare.

Am considerat ca fiind cel mai bun cromozom acela cu care s-a obținut cea mai mare valoare a funcţiei de fitness. Ca şi metode de selecţie a cromozomilor din cadrul unei populaţii, am folosit două metode: metoda „Roulette Wheel" prezentată în secţiunea 5.4.2.1 şi metoda lui Gauss prezentată în secţiunea 5.4.2.2. Pentru crearea unei populaţii am folosit toţi cei 3 operatori genetici prezentaţi în secţiunea 5.4.3 astfel: în 30% din cazuri am folosit operatorul de selecţie (selectând întotdeauna cel mai bun cromozom), în 40% din cazuri am folosit operatorul de mutaţie iar operatorul de crossover l-am folosit în 30% din cazuri. Totodată am respectat condiţia iniţială pentru valoarea ponderilor ca $w_1 > w_2 > \ldots w_{16}$. Pentru operatorul de crossover am căutat punctul de secţionare (s), astfel încât condiţia $w_s^{primul\_părinte} > w_{s+1}^{al\_doilea\_părinte}$ este satisfăcută. Pentru operatorul de mutaţie, după alegerea aleatoare a punctului de mutaţie (m), noua valoare este aleasă aleator din intervalul $w \in (w_{m-1}, w_{m+1})$. Aceste condiţii au fost impuse în algoritmul genetic, deoarece considerăm că un cromozom care nu le respectă nu reprezintă un cromozom valid şi astfel el nu ar trebui să fie generat.

Testele au fost efectuate pe seturile de date utilizate şi de către metaclasificatorii propuşi în secţiunea 6.4.1. Pentru fiecare metodă de selecţie a cromozomilor, am efectuat câte 4 teste, având în vedere faptul că rezultatul metaclasificatorului depinde de valorile cu care se iniţializează ponderile pentru cele 16 clase.

Rezultatele obţinute pentru metoda „Roulette Wheel" de alegere a cromozomilor le prezint în tabelul 6.5. În acest tabel prezint selectiv, din 50 în 50 de generaţii, valorile acurateţei obţinute de metaclasificator.

Generaţii	Roulette 1	Roulette 2	Roulette 3	Roulette 4	Media
1	0.76733305	0.777541472	0.778392174	0.794980859	0.779561889
50	0.868566567	0.873245427	0.870267971	0.859208847	0.867822203
100	0.871969375	0.873670778	0.871544024	0.876648235	0.873458103
150	0.873670778	0.874096129	0.874096129	0.877498937	0.874840493
200	0.873670778	0.87920034	0.876222884	0.877498937	0.876648235
250	0.875372182	0.879625691	0.878774989	0.87920034	0.878243301
300	0.875372182	0.880051042	0.878774989	0.880051042	0.878562314
350	0.875797533	0.880901744	0.878774989	0.880051042	0.878881327
400	0.877498937	0.880901744	0.87920034	0.880051042	0.879413016
450	0.87920034	0.880901744	0.882177797	0.880051042	0.880582731
500	0.87920034	0.881327095	0.882177797	0.880476393	0.880795406
550	0.880901744	0.881327095	0.882603148	0.880476393	0.881327095
600	0.880901744	0.881752446	0.8838792	0.880476393	0.881752446
650	0.880901744	0.882177797	0.8838792	0.881752446	0.882177797
700	0.882177797	0.882177797	0.8838792	0.881752446	0.88249681
750	0.882177797	0.882177797	0.8838792	0.882177797	0.882603148
800	0.882177797	0.882177797	0.8838792	0.883453849	0.882922161
850	0.883028499	0.882177797	0.8838792	0.883453849	0.883134836
900	**0.884304551**	0.882177797	0.8838792	0.883453849	0.883453849
950	**0.884304551**	0.882603148	0.8838792	0.883453849	0.883560187
1000	**0.884304551**	0.883028499	0.8838792	0.883453849	**0.883666525**

Tabelul 6.5 Acurateţea obţinută de metaclasificator utilizând metoda de selecţie „Roulette Whele" a cromozomilor.

În Fig.6.17 am reprezentat grafic rezultatele prezentate în tabelul 6.5.

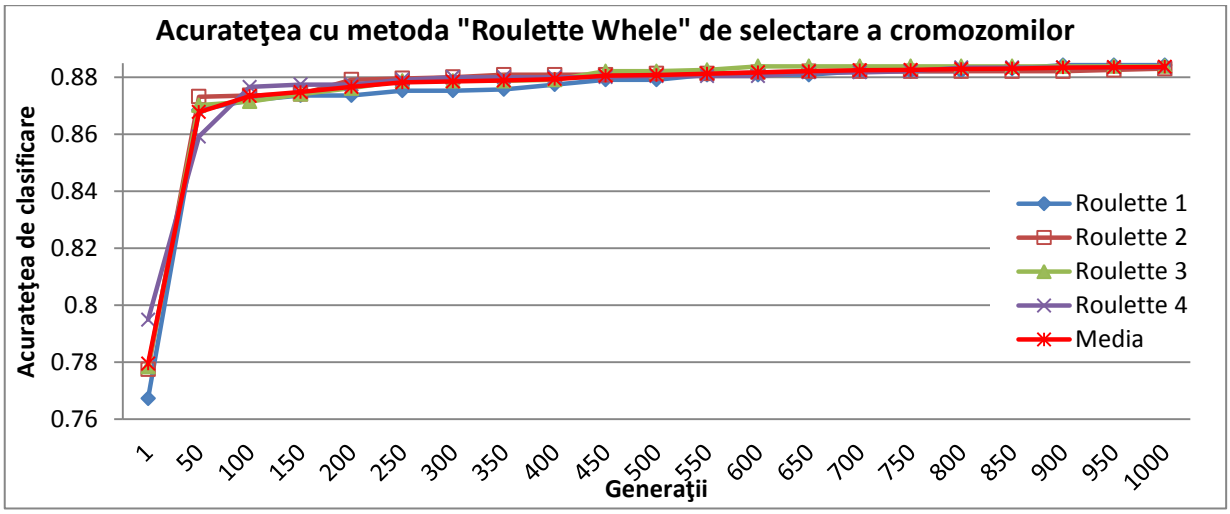


Fig. 6.17 Acurateţea obţinută de metaclasificator utilizând metoda ruletei de selectare a cromozomilor

Rezultatele obţinute pentru metoda Gauss de alegere a cromozomilor sunt prezentate în tabelul 6.6. În acest tabel am prezentat selectiv, din 50 în 50 de generaţii, valorile acurateţei obţinute de metaclasificator.

Generaţia	Gauss 1	Gauss 2	Gauss 3	Gauss 4	Media
1	0.806465	0.818801	0.78775	0.801786	0.803701
50	0.871969	0.849851	0.861761	0.854955	0.859634
100	0.876223	0.872395	0.869843	0.870693	0.872288
150	0.877924	0.874947	0.870693	0.873245	0.874202
200	0.87835	0.880476	0.870693	0.873245	0.875691
250	0.878775	0.880476	0.870693	0.874096	0.87601
300	0.878775	0.880902	0.870693	0.876223	0.876648
350	0.880902	0.880902	0.870693	0.878775	0.877818
400	0.882603	0.880902	0.874947	0.881327	0.879945
450	0.882603	0.881327	0.874947	0.881327	0.880051
500	0.882603	0.881752	0.877074	0.881327	0.880689
550	0.882603	0.882603	0.877074	0.881327	0.880902
600	0.882603	0.883028	0.8792	0.881752	0.881646
650	0.882603	0.884305	0.880051	0.882178	0.882284
700	0.882603	0.884305	0.880051	0.882178	0.882284
750	0.882603	0.884305	0.880051	0.883028	0.882497
800	0.882603	0.885155	0.880476	0.883028	0.882816
850	0.882603	0.885155	0.880476	0.883028	0.882816
900	0.882603	**0.885581**	0.882178	0.883028	0.883348
950	0.882603	**0.885581**	0.883028	0.883454	0.883667
1000	0.882603	**0.885581**	0.883454	0.883454	**0.883773**

Tabelul 6.6 Acurateţea obţinută de metaclasificator utilizând metoda de selecţie Gauss a cromozomilor.

În Fig.6.18 am reprezentat grafic rezultatele din tabelul 6.6.

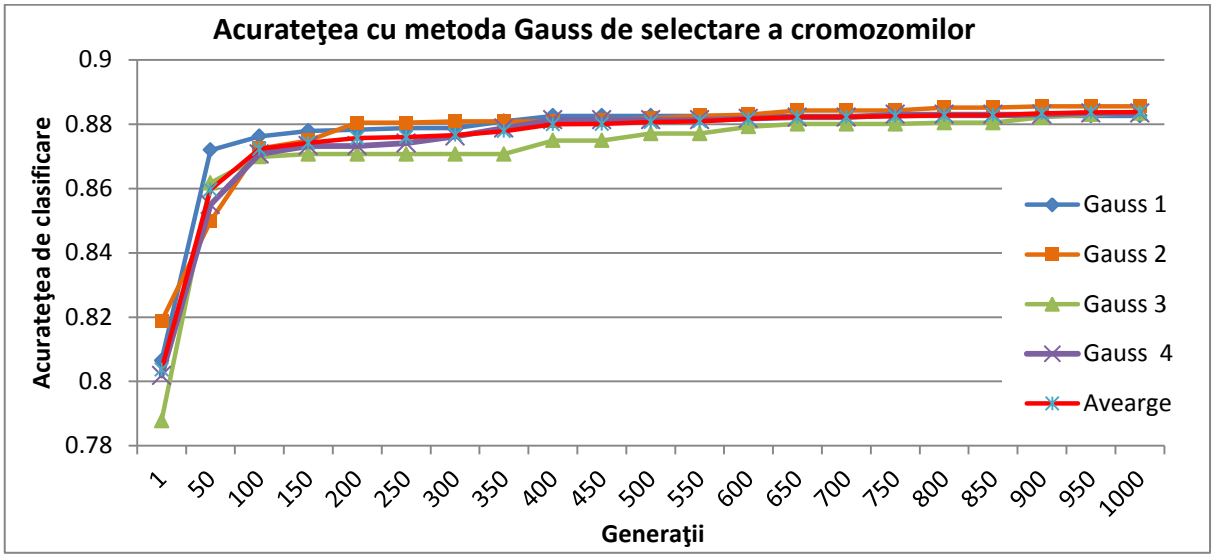


Fig. 6.18 Acurateţea obţinută de metaclasificator utilizând metoda lui Gauss de selectare a cromozomilor

Pentru cel mai bun rezultat (Gauss 2) pentru acurateţe obţinut de metaclasificator în cazul utilizării metodei Gauss de selectare a cromozomilor, ponderile *w* calculate pentru poziţiile claselor sunt următoarele:

Weights for Testing best cromosome

0.82337, 0.820739, 0.797028, 0.796625, 0.782079, 0.776642, 0.771496, 0.738956, 0.737421, 0.733313, 0.731112, 0.715836, 0.683754, 0.438121, 0.214071, 0.110157

Rezultatele obţinute de metaclasificatorul cu ponderi calculate s-au îmbunătăţit în medie faţă de rezultatele metaclasificatorului neadaptiv prezentat în secţiunea 6.3.1 cu 1.16%; în cazul utilizării metodei „Roulette Wheele" de selecţie a cromozomilor, acurateţea de clasificare ajunge la 88.36% iar în cazul utilizării metodei de selecţie Gauss, acurateţea de clasificare creşte cu 1.17% şi ajungând la 88.37%. Cel mai bun rezultat al acestui metaclasificator a fost obţinut în cazul metodei Gauss de selecţie a cromozomilor, acurateţea de clasificare ajungând la **88.55%** în acest caz.

Aceste rezultate au fost prezentate şi în [Cret11_a].

6.5 *Arhitecturi adaptive propuse şi dezvoltate*

6.5.1 Metaclasificatoare bazate pe similaritate

În secţiunea 6.2 am propus posibile soluţii pentru îmbunătăţirea acurateţei de clasificare a metaclasificatorului. În secţiunea 6.2.2 am afirmat că, în cazul unui nou document d_n care trebuie clasificat, se ia pe rând fiecare clasificator şi se calculează distanta între d_n şi fiecare document din coada de erori a clasificatorului respectiv. Dacă cel puţin o distanţă calculată este mai mică decât pragul stabilit, metaclasificatorul nu va folosi acel clasificator pentru a clasifica documentul d_n. În cazul în care se rejectează astfel toţi clasificatorii, metaclasificatorul va alege totuşi pe cel care are distanţa cea mai mare obţinută (chiar dacă este mai mică decât pragul stabilit). Actualmente, metaclasificatorul va prezice clasa specificată de acest clasificator. Modificarea adusă în acest caz metaclasificatorului ar fi că, în acest caz, clasificatorul ales să nu mai selecteze clasa cu valoarea cea mai mare (pentru că oricum va da greş, deoarece este predispus să clasifice eronat tipul respectiv de documente), ci să selecteze clasa imediat următoare din lista de clase pe care le prezice. Se va alege următoarea clasă prezisă doar dacă valoarea pentru aceasta este suficient de apropiată de valoarea maxim obţinută de clasificator (cu

un ε=0.5 ales în experimentele efectuate). În acest caz clasificatorul ar specifica o altă clasă pentru documentul curent d_n. Efectuând aceste modificări rezultatele metaclasificatorului cu 9 clasificatori s-au îmbunătăţit. În continuare vom numi acest metaclasificator „metaclasificator cu 9 clasificatori modificat".

Prezentăm comparativ rezultatele obţinute de metaclasificatorul cu 9 clasificatori modificat în cazul selecţiei bazate pe distanţa euclidiană şi selecţiei bazate pe distanţa cosinus. Pentru selecţia bazată pe votul majoritar, nefiind vorba de învăţare, modificările aduse metaclasificatorul nu au nici o influenţă asupra rezultatului final. De asemenea, am prezentat în fiecare grafic şi limita maximă obtenabilă a metaclasificatorului cu 9 clasificatoare.

6.5.1.1 Rezultate obţinute în cazul selecţiei bazate pe distanţa euclidiană

Metaclasificatorul cu 9 clasificatori modificat care se bazează pe distanţa euclidiană şi-a îmbunătăţit clasificarea, obţinând o acurateţe a clasificării de **93.32%** faţă de cel cu 9 clasificatori nemodificat care a obţinut doar 90.38%. Reamintim faptul că, în aceleaşi condiţii, metaclasificatorul cu 8 clasificatori de tip SVM din [Mora07] a obţinut o acurateţe de clasificare de 92.04%, maximul obţinut pe 8 clasificatoare. În primii 7 paşi, acurateţea clasificării pentru metaclasificatorul cu 9 clasificatoare modificat este aproape identică cu cea a metaclasificatorului cu 9 clasificatoare nemodificat, deoarece în primii paşi nu apare niciodată cazul în care trebuie aleasă ce-a de-a doua clasă. În primii paşi, rezultatele între cele două metaclasificatoare cu 9 clasificatori, cel modificat şi nemodificat sunt puţin diferite, deoarece întotdeauna se alege aleator un clasificator din cei 9 existenţi.

Rezultatele obţinute le prezint comparativ în tabelul 6.7.

Paşi	Distanţă euclidiană SBED cu 8 clasificatori	Distanţă euclidiană SBED cu 9 clasificatori	Distanţă euclidiană SBED cu 9 clasificatori şi selecţie modificată
1	84.77244	83.79413	84.77244
2	85.70821	85.07018	83.07103
3	86.34624	83.28371	83.49638
4	86.60145	83.53892	84.64483
5	86.81412	85.66567	85.32539
6	87.32454	84.43216	86.51638
7	88.43046	82.00766	85.70821
8	89.83411	87.74989	88.55806
9	90.04679	88.60060	92.72650
10	90.55721	88.72820	93.06678
11	91.23777	88.34538	**93.32199**
12	91.57805	88.60060	**93.32199**
13	92.04594	89.87665	**93.32199**
14	92.04594	90.38707	**93.32199**

Tabelul 6.7 Acurateţea obţinută de metaclasificator utilizând distanţa euclidiană şi selecţia clasei modificată.

În Fig. 6.18 prezint grafic rezultatele, la care am adăugat şi limita teoretică maximă 98.36% la care poate ajunge metaclasificatorul format din 9 clasificatoare.

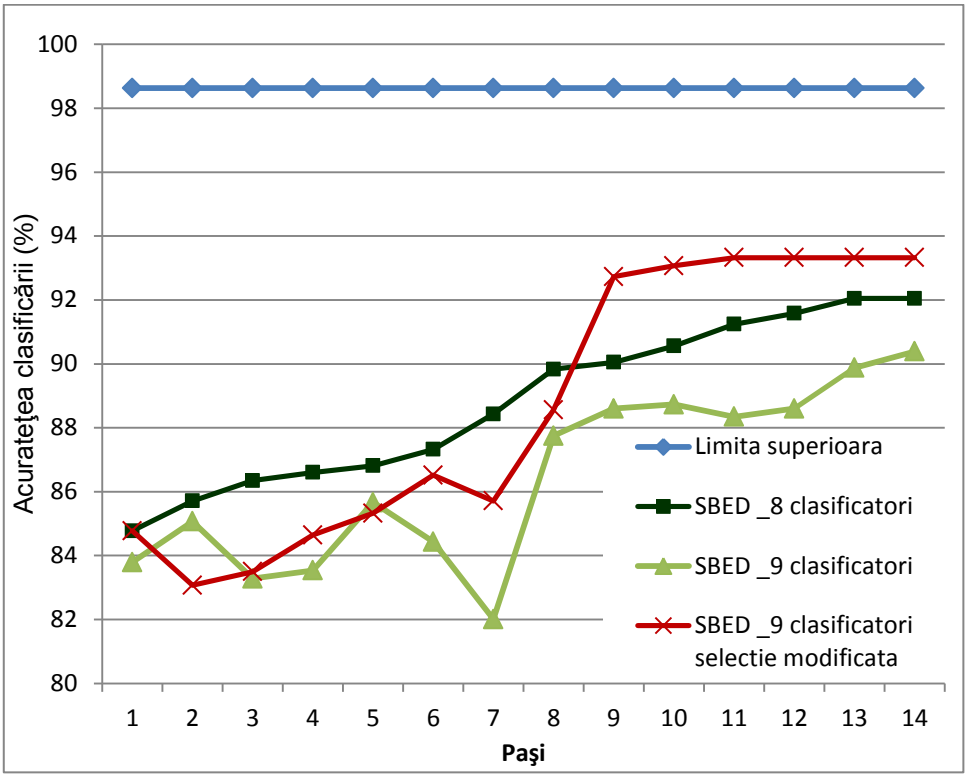


Fig. 6.18 Acurateţea de clasificare obţinută de metaclasificatorul cu 9 clasificatori modificat bazat pe distanţa euclidiană

6.5.1.2 Rezultatele obţinute în cazul selecţiei bazate pe distanţa cosinus.

În acest caz, acurateţea de clasificare a metaclasificatorului s-a îmbunătăţit de la 93.10% la **93.87%.** Reamintim că metaclasificatorul cu 8 clasificatori de tip SVM în aceleaşi condiţii a obţinut o acurateţe de clasificare de 89.74%. Rezultatele obţinute le prezint comparativ în tabelul 6.8, adăugând şi valorile obţinute de acelaşi metaclasificator.

Paşi	Distanţă cosinus SBCOS 8 clasificatori	Distanţă cosinus SBCOS 9 clasificatori	Distanţă cosinus SBCOS 9 clasificatori şi selecţie modificată
1	85.32539345	90.98256061	91.32284134
2	85.32539345	91.02509570	92.25861336
3	86.60144619	91.19523607	92.89663973
4	86.09102510	92.00340281	93.44959592
5	86.47384092	92.64142918	93.15185028
6	87.32454275	92.42875372	92.74819226
7	87.66482348	92.47128881	93.36452573
8	88.09017439	92.64142918	93.74734156
9	88.30284985	92.89663973	93.02424500
10	89.66397278	92.47128881	93.36452573
11	88.89834113	92.34368354	92.93917482
12	89.57890259	92.68396427	93.32199064
13	89.23862186	93.10931519	93.66227137
14	89.74904296	93.10931519	**93.87494683**

Tabelul 6.8 Acurateţea obţinută de metaclasificator, utilizând distanţa cosinus şi selecţia clasei modificată

În Fig. 6.19 prezint grafic rezultatele, la care am adăugat şi limita superioară la care poate ajunge teoretic metaclasificatorul propus.

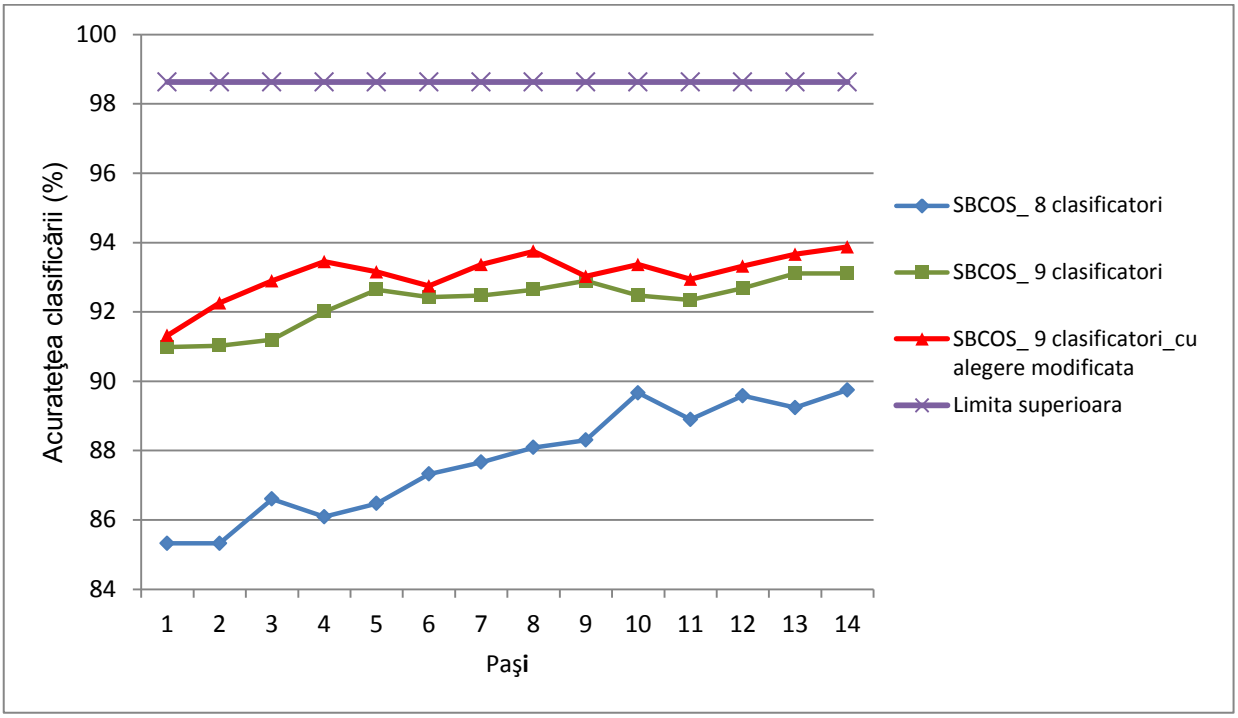


Fig. 6.19 Acurateţea de clasificare a metaclasificatorului cu 9 clasificatori modificat, bazat pe cosinus

Acurateţea de clasificare de **93.87%** reprezintă cel mai bun rezultat obţinut de către metaclasificator, din toate experimentele prezentate până în acest moment.

Rezultatele din această secţiune au fost prezentate şi în [Mora10].

6.5.2 Metaclasificator bazat pe algoritmul Backpropagation

Metaclasificatorul dezvoltat în continuare se bazează pe o preclasificare de documente prezentată în secţiunea 6.4.1.2 şi o reţea neuronală de tip feed-forward cu învăţare online. Am dorit să includem în metaclasificatorul numit în continuare **MC-BP (Metaclasificator cu reţea Backpropagation)** o reţea neuronală, deoarece am considerat că un metaclasificator adaptiv poate va reuşi să se „adapteze" şi la datele cu probleme, care există în setul de antrenare/testare. Reţelele neuronale sunt sisteme care se adaptează la schimbările survenite în seturile de date, astfel că metaclasificatorul MC-BP devine unul mult mai adaptiv decât metodele SBDE şi SBCOS dezvoltate şi prezentate în [Mora06_a].

Cel mai bun rezultat de până acum, prezentat în lucrare, s-a obţinut cu ajutorul metaclasificatorului bazat pe cosinus, unde acurateţea de clasificare a atins valoarea de 93,87% pe setul de test T1 cu 2351 documente.

Pentru antrenarea şi testarea reţelei neuronale cu învăţare Backpropagation, am plecat de la setul de vectori obţinut de metaclasificatorul neadaptiv prezentat în secţiunea 6.4.1.2. Am antrenat acel metaclasificator cu setul de antrenament A1 (4702 documente) şi l-am testat folosind setul de test T1 (2351 documente) [Cret08]. Ca şi intrare în acest metaclasificator, avem setul de date iar la ieşire obţinem un set de vectori, câte un vector pentru fiecare document de intrare, de 16 elemente fiecare. Setul de vectori obţinut, pornind de la setul de documente de antrenare A1, pe care îl vom numi în continuare setul AV1 este un vector agregat conţinând 16 elemente (scalari). Acesta va fi folosit în etapa de antrenare a reţelei. Setul de vectori obţinut pornind de la setul de documente de testare T1, numit în continuare TV1, va fi folosit atât in etapa de testare cât şi în etapa de determinare a configuraţiei reţelei.

În ceea ce priveşte arhitectura reţelei backpropagation, am ales una care conţine două straturi de unităţi cu funcţia sigmoidală de activare, iar fiecare unitate de pe fiecare strat este

conectată cu toate unităţile de pe stratul precedent. Deoarece la intrare avem la dispoziţie vectori agregaţi de 16 elemente, reţeaua va avea pe stratul de intrare 16 neuroni. La ieşire, metaclasificatorul trebuie să „prezică" clasa în care se găseşte documentul curent. Atunci reţeaua Backpropagation va avea la ieşire tot un număr de 16 neuroni, deoarece avem 16 clase distincte, iar la un moment dat va fi activ doar un neuron. În stratul ascuns avem un număr variabil de neuroni, alegerea acestui număr va fi făcut în funcţie de rezultatele simulărilor care vor fi prezentate în secţiunea următoare (6.4.2.1).

În faza de antrenare, deoarece reţeaua este una cu învăţare supervizată, pentru setul de antrenare am creat un set cu răspunsurile corecte pentru fiecare document în parte. Un astfel de răspuns conţine valoarea „1" pe poziţia clasei corecte şi valoarea „0" în rest. Structura metaclasificatorului adaptiv M-BP este prezentată în Fig. 6.20.

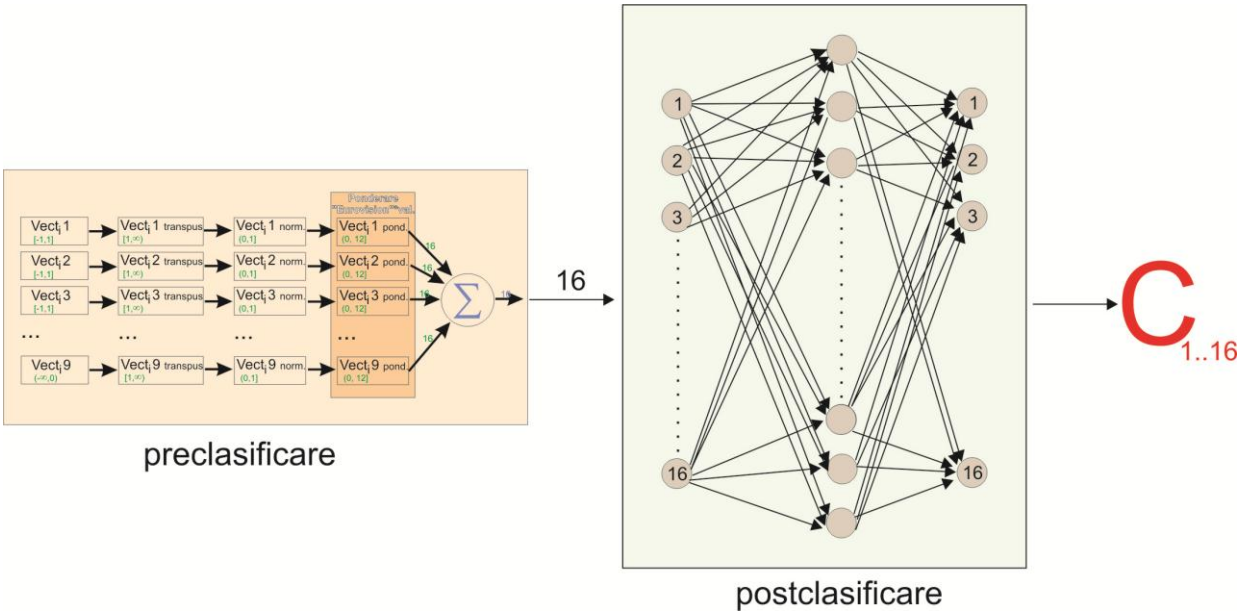


Fig. 6.20 Metaclasificator adaptiv M-BP

6.5.2.1 Influenţa numărului de neuroni de pe stratul ascuns

Deoarece nu există o formulă matematică pentru calcularea numărului optim de neuroni necesari pe stratul ascuns, în această secţiune prezentăm experimentele realizate în vederea determinării numărului optim de neuroni pentru reţeaua prezentată în Fig. 6.20. Experimentele prezentate în această secţiune sunt efectuate pe setul TV1, atât antrenarea cât şi testarea [Cret09_b].

Ca şi metodă de evaluare, am oprit antrenarea reţelei după ce aceasta a ajuns din punct de vedere al erorii de antrenare la anumită valoare, am evaluat reţeaua pe întreg setul, din punct de vedere al numărului de documente incorect clasificate, după care am continuat antrenarea. Valorile erorii la care am oprit antrenarea reţelei sunt calculate ca fiind sumă a tuturor erorilor obţinute pentru fiecare exemplu în parte din setul TV1. Pentru calculul erorii, am folosit formula (5.39). Evaluarea reţelei în acel punct se face prin numărul de documente incorect clasificate de către reţea. Având în vedere că setul TV1 conţine 2351 vectori şi că eroarea pe fiecare vector reprezintă o sumă de 16 elemente, eroarea totală va avea valori supraunitare. Vom începe testarea pornind de la o valoare a erorii totale egală cu 500, ceea ce înseamnă o eroare medie pe fiecare vector de 0.21. Ideea este de ajunge cu eroarea de antrenare la o valoare cât mai mică, într-un timp cât mai scurt.

În Fig. 6. 21 prezint evoluţia acurateţei de clasificare a MC-BP, în funcţie de numărul de neuroni de pe stratul ascuns. Iniţial am pornit de la un număr de 17 neuroni pe stratul ascuns. Toate experimentele prezentate în această secţiune au coeficientul de învăţare fix η=1. În momentul în care timpul necesar reţelei pentru a reduce eroarea de antrenare devine mare, am oprit antrenarea reţelei pentru acea configuraţie. Din acest motiv, în figurile prezentate în continuare, unele grafice nu coboară cu eroarea de antrenare până la valoarea minimă obţinută de cea mai bună configuraţie testată.

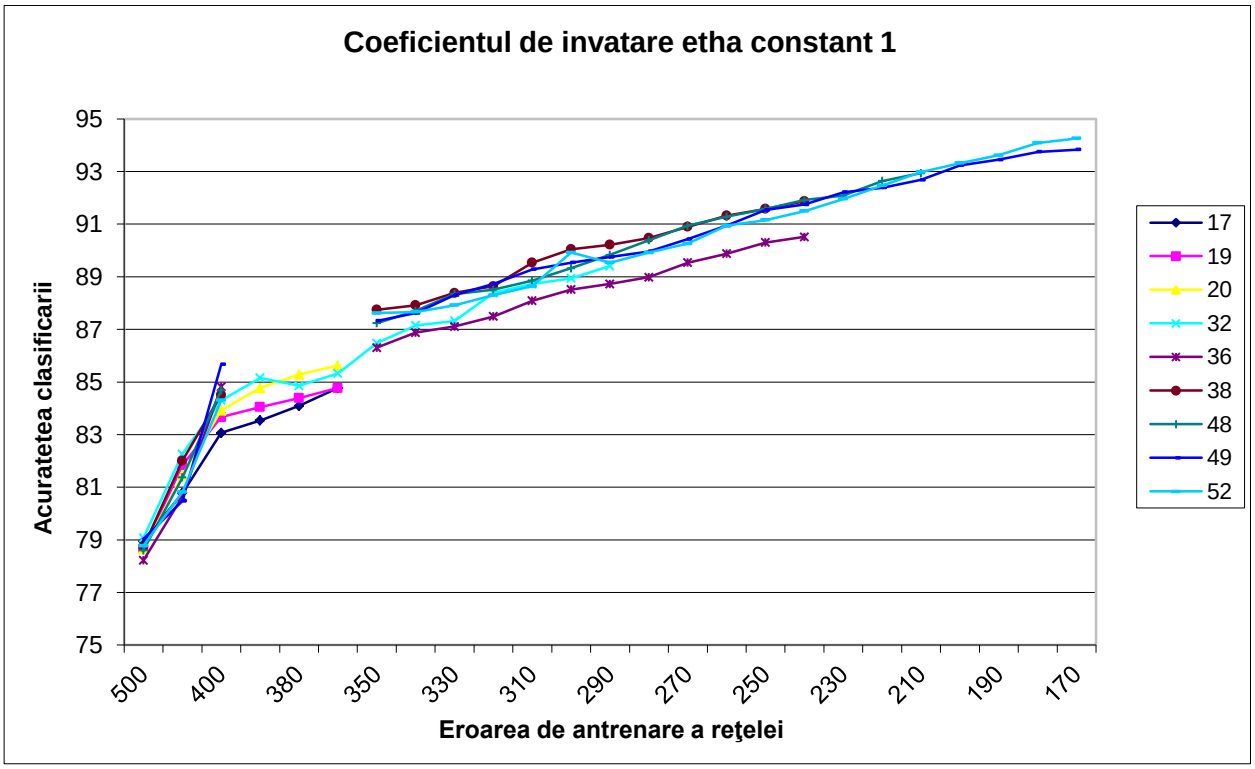


Fig. 6.21 Influenţa numărului de neuroni de pe stratul ascuns asupra acurateţei de clasificare.

Coeficient de învăţare η=1

În acest grafic am început cu un număr mic de neuroni pe stratul ascuns, pentru a avea un timp de învăţare relativ mic (din punct de vedere al calculelor efectuate), dar în momentul în care am ajuns la valori totale ale erorii de antrenare în jur de 200, timpul de antrenare creşte, iar datorită coeficientului de învăţare mare, reţeaua începe să fluctueze în jurul unei valori a erorii totale. Din acest motiv am oprit evaluarea reţelei la un număr de 52 de neuroni pe stratul ascuns, chiar dacă acurateţea de clasificare creştea în continuare.

6.5.2.2 Influenţa coeficientului de învăţare

În această secţiune prezint experimente în care modific şi pasul de învăţare. În cazurile în care reţeaua este mai simplă (are un număr mai mic de neuroni pe stratul ascuns), de la un moment dat eroarea totală a început să scadă foarte încet, moment în care am oprit antrenarea reţelei. De aceea, din acel punct nu vor mai fi valori în graficele pe care le prezint. Am modificat numărul de neuroni utilizând multipli ai lui 16 (Fig. 6.22 şi 6.23). Pe măsură ce am crescut numărul neuronilor de pe stratul ascuns păstrând pas de învăţare η=1, eroarea a scăzut de la 0.2 la 0.04 per exemplu. Cea mai bună valoare a acurateţei de clasificare obţinută până în acest moment este de 94.26%, fiind deja superioară celei mai bune valori obţinute cu metaclasificatorul de tip SBCOS cu 9 clasificatoare (93.32%).

Totuşi, odată cu creşterea numărului de neuroni de pe stratul ascuns, am observat că timpul de antrenare pentru reţea scade, chiar dacă numărul de calcule care trebuie efectuate

creşte. În cazul în care avem mai mulţi neuroni pe stratul ascuns, reţeaua ajunge mai repede la o eroare mai mică, iar fluctuaţiile apar la valori mici ale erorii. Numărul mai mare de neuroni pe stratul ascuns duce la o micşorare mai rapidă a erorii datorită unei distribuţii mai adecvate. Această convergenţă superioară compensează timpul necesar efectuării unui număr mult mai mare de calcule. De asemenea şi acurateţea clasificării creşte semnificativ.

În continuare voi prezenta experimente efectuate pe acelaşi număr de neuroni pe stratul ascuns, dar cu un coeficient de învăţare care scade în timp. Practic, oprim antrenarea reţelei la anumite valori ale erorii totale de antrenare, efectuăm testarea, şi continuăm antrenarea reţelei cu un coeficient de învăţare micşorat. De exemplu, până la o valoare a erorii totale egală cu 350, coeficientul de învăţare este „1”, între 340 şi 320 este „0.9”, între 320 şi 300 este „0.8” şi scade până la valoarea de „0.1” la o valoare a erorii de 150.

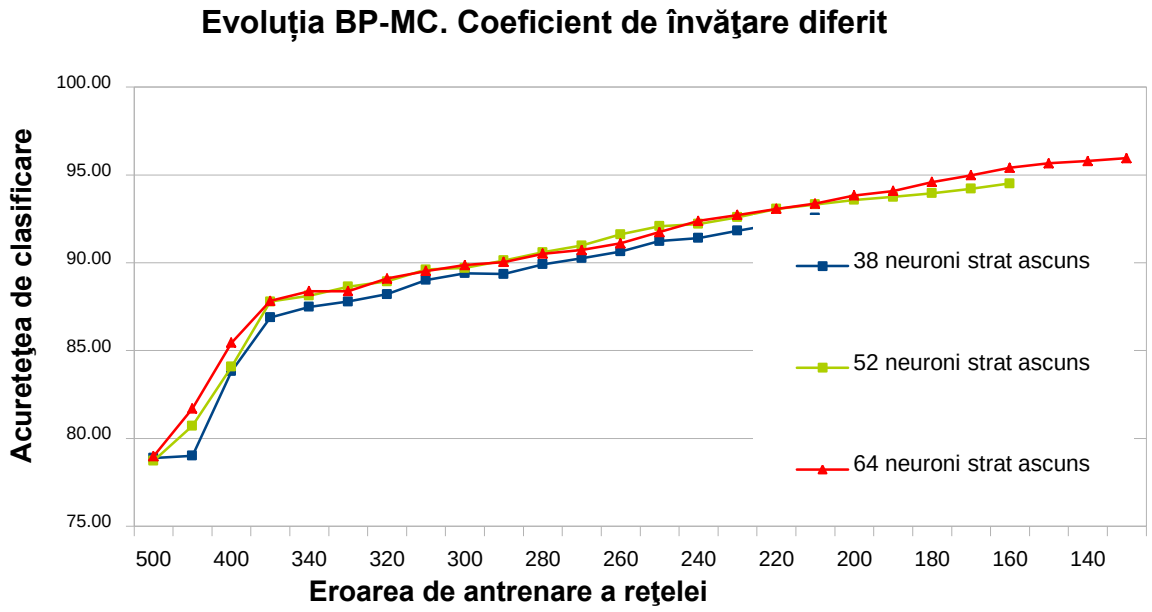


Fig. 6.22 Influenţa numărului de neuroni de pe stratul ascuns asupra acurateţei de clasificare. Coeficient de învăţare η diferit

În Fig. 6.22 am prezentat doar evoluţia reţelei pentru un număr de neuroni pe stratul ascuns egal cu 38, 52 şi 64, deoarece acestea au obţinut rezultatele cele mai bune în cazul coeficientului de învăţare constant. În acest grafic am coborât cu eroarea de învăţare până la valoarea de 130 (coeficientul de învăţare a ajuns la 0.01), deoarece reţeaua nu a mai fluctuat mult în jurul erorii şi astfel timpul de antrenare a fost redus. În acest caz, cu un număr de 52 de neuroni pe stratul ascuns, numărul de vectori incorect clasificaţi pentru o eroare totală de antrenare egală cu 170 este de 136. În acest grafic am prezentat şi rezultate obţinute pe o reţea cu 64 de neuroni pe stratul ascuns, caz în care eroarea totală de antrenare a scăzut la valoarea de „130” iar numărul de documente incorect clasificate s-a redus la 95, ceea ce reprezintă o acurateţe de clasificare de 95.96%.

Am observat că, odată cu creşterea numărului de neuroni de pe stratul ascuns, se îmbunătăţeşte acurateţea clasificării, deoarece putem ajunge la o eroare de antrenare mult mai mică. Am efectuat şi unele experimente în care numărul de neuroni de pe stratul ascuns este mai mare, regula de alegere a numărului de neuroni de pe stratul ascuns fiind multipli ai lui 16 prezentate în Fig. 6.23.

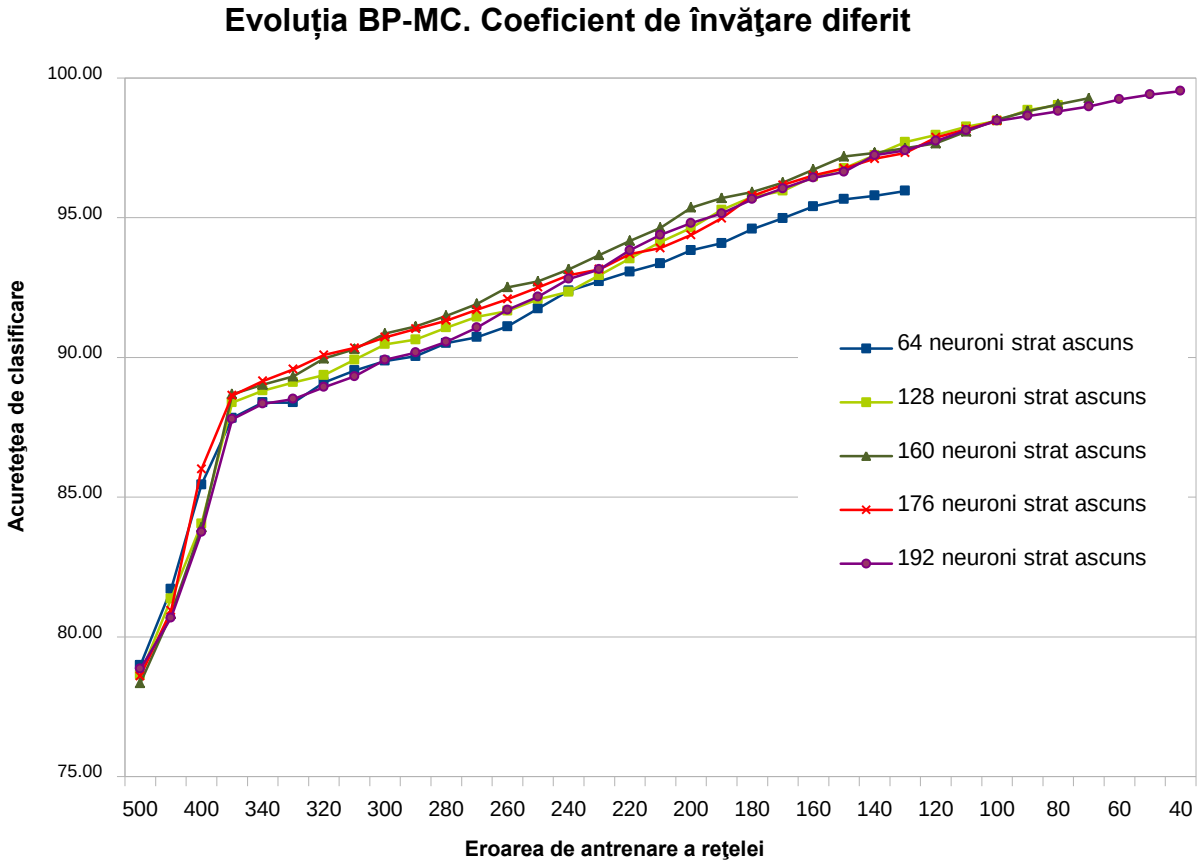


Fig. 6.23 Influenţa numărului de neuroni de pe stratul ascuns asupra acurateţei de clasificare. Coeficient de învăţare η diferit

În acest caz, arhitectura cu 160 de neuroni pe stratul ascuns a obţinut cele mai multe rezultate bune, dar, în momentul în care eroarea totală de antrenare a scăzut până la valoarea 70, timpul de antrenare pentru ca eroarea să scadă la valoarea 60 a depăşit 24 ore! De aceea am realizat o arhitectură a reţelei şi cu 192 de neuroni pe stratul ascuns, care a reuşit să coboare la o eroare de antrenare egală cu 40, caz în care numărul de documente incorect clasificate este de doar **11**. Acest număr reprezintă o acurateţe a clasificării pentru metaclasificatorul M-BP de **99.53%.** O eroare totală de antrenare egală cu 40 înseamnă o eroare medie per exemplu egală cu 0.017.

Experimentele prezentate au fost rulate pe un calculator P-IV Dual core la f=1.9GHz cu 2GB DRAM şi sistem de operare Windows Vista. Prezentăm în Fig. 6.24 rezultatele comparative între arhitectura reţelei cu 52 neuroni pe stratul ascuns şi coeficient de învăţare 1 şi respectiv aceeaşi arhitectură dar coeficient de învăţare descrescător în timp. Pentru a ajunge la prima oprire (eroare 500), reţeaua are nevoie de mai mult timp, deoarece porneşte de la o eroare mare dar care scade foarte repede. Timpii pentru următoarele opriri ale reţelei sunt timpii necesari reţelei pentru a ajunge de la valoarea erorii de la pasul curent la valoarea erorii de la următoarea oprire.

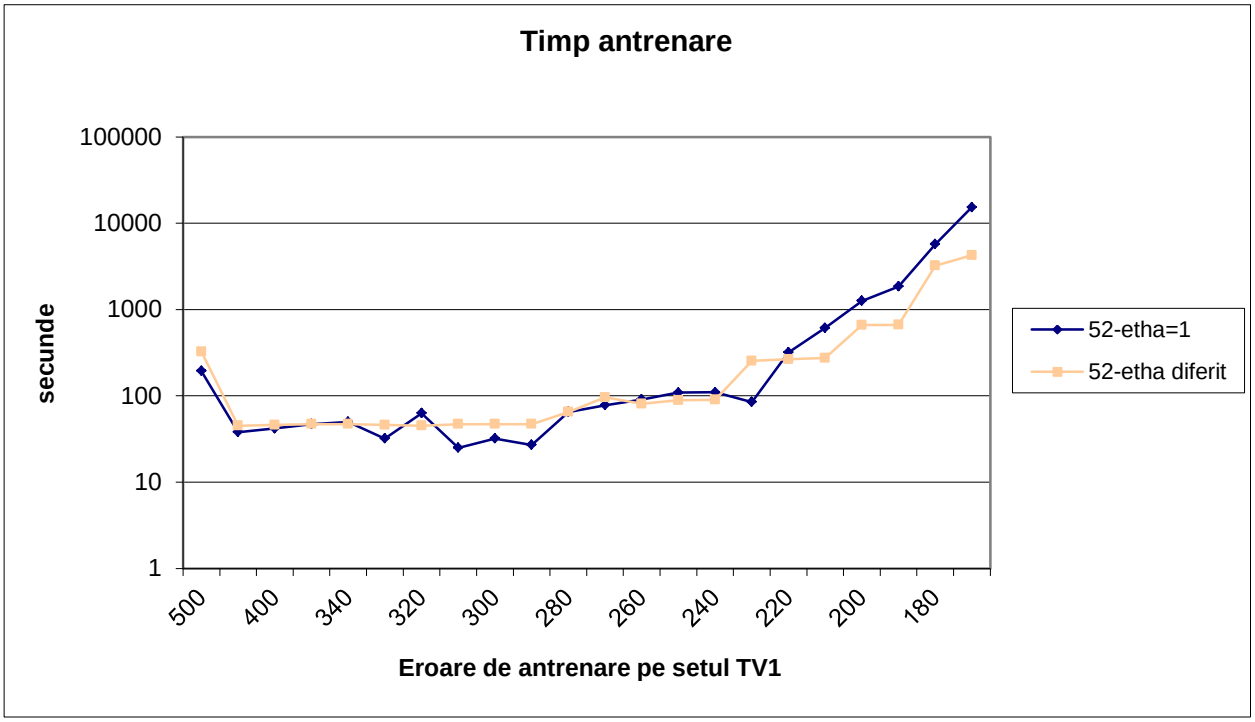


Fig. 6.24 Timpul de antrenare - comparaţie între 2 arhitecturi cu 52 neuroni pe stratul ascuns.

Rezultatele prezentate anterior au fost obţinute antrenând şi testând reţeaua Backpropagation pe setul TV1, care conţine 2351 vectori (deci pe acelaşi set ceea ce reduce valoarea practică). În cele ce urmează prezentăm rezultatele obţinute în cazul antrenării pe setul AV1 (4702 vectori) şi a testării pe setul TV1.

Rezultate obţinute în cazul antrenării pe setul AV1 şi a testării pe TV1 sunt prezentate în Fig. 6.25. Prezint rezultate doar pentru arhitecturi ale reţelei cu un număr de neuroni mai mare de 96 pe stratul ascuns şi un coeficient de învăţare descrescător în timp. Şi în acest caz, pentru testare, oprim reţeaua în momentul în care atinge un anumit prag al erorii de antrenare, o testăm pentru a obţine numărul de documente incorect clasificate, după care continuăm cu antrenarea. În acest caz, eroarea totală de antrenare este obţinută ca o sumă a tuturor celor 4702 erori, ceea ce reprezintă o medie a erorii per exemplu de 0.11 în cazul erorii totale egale cu 500. În acest experiment am ajuns la o eroare totală egală cu 80, ceea ce înseamnă o eroare medie de 0.017 per exemplu.

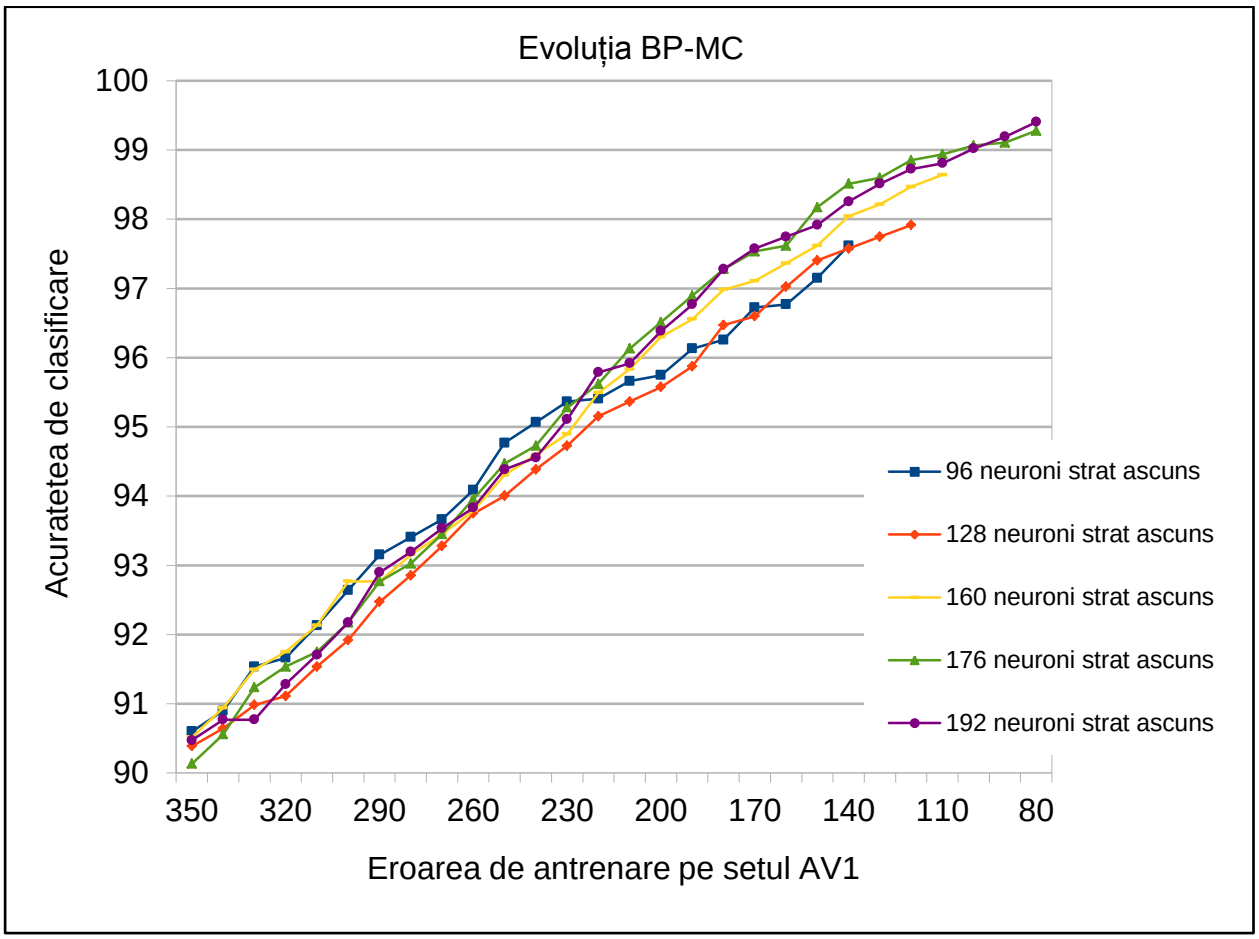


Fig. 6.25 Acurateţea de clasificare în cazul antrenării pe setul AV1 şi a testării pe setul TV1

Arhitectura cu 176 de neuroni pe stratul ascuns a obţinut cele mai multe valori minime pentru numărul de documente incorect clasificate, dar în momentul în care eroarea totală de antrenare a scăzut sub valoarea 100, rezultatele cele mai bune au fost obţinute de arhitectura cu 192 de neuroni pe stratul ascuns. În acest caz am obţinut un număr de 14 documente incorect clasificate, ceea ce reprezintă o acurateţe de clasificare a metaclasificatorului de **99.40%.** Diferenţa faţă de cea mai bună valoare faţă de cea cu 176 de neuroni pe stratul ascuns este de doar **3** documente incorect clasificate.

Coeficientul de învăţare	Eroarea totală de antrenare	Număr de neuroni pe stratul ascuns				
		96	128	160	176	192
1	500	381	369	**368**	376	374
1	450	345	325	334	324	**321**
1	400	278	282	**268**	282	288
1	350	**221**	226	223	232	224
0,9	340	214	220	**213**	222	217
0,9	330	**199**	212	200	206	217
0,8	320	196	209	**194**	199	205
0,8	310	186	199	**185**	194	195
0,7	300	173	190	**170**	184	184
0,7	290	**161**	177	170	170	167
0,6	280	**155**	168	161	164	160
0,6	270	**149**	158	154	154	152
0,6	260	**139**	147	146	142	145
0,6	250	**123**	141	134	130	132
0,5	240	**116**	132	127	124	128
0,5	230	112	124	120	**111**	115
0,5	220	108	114	106	103	**99**
0,4	210	102	109	98	**91**	96

0,4	200	100	104	87	**82**	85
0,3	190	91	97	81	**73**	76
0,3	180	88	83	71	**64**	**64**
0,2	170	77	80	68	58	**57**
0,2	160	76	70	62	56	**53**
0,1	150	67	61	56	**43**	49
0,1	140	56	57	46	**35**	41
0,1	130		53	42	**33**	35
0,1	120		49	36	**27**	30
0,1	110			32	**25**	28
0,1	100				**22**	23
0,01	90				21	**19**
0,01	80				17	**14**

Tabel 6.3 Număr de documente incorect clasificate

În Tabelul 6.3 am prezentat numărul de documente incorect clasificate obţinut de arhitecturile testate. Pentru fiecare arhitectură am prezentat valoarea obţinută pentru toate testele efectuate în timpul antrenării reţelei. Astfel, în coloana a doua se prezintă valorile erorii totale de antrenament la care reţeaua a fost oprită şi testată. În prima coloană se prezintă valorile coeficientului de învăţare care a fost folosit pentru reţea, astfel încât eroarea de antrenare a reţelei să coboare la valoarea precizată.

Acest nou metaclasificator cu o reţea neuronală cu număr suficient de mare de neuroni pe stratul ascuns a reuşit să depăşească şi limita maximă de 98.63% la care ar fi putut ajunge „teoretic" clasificatorii incluşi în cadrul metaclasificatorului.

Foarte interesant, acest metaclasificator neuronal cu învăţare supervizată a demonstrat faptul că acurateţea de 98.63% nu este de fapt limita maximă a metaclasificării, aşa cum eu considerasem. Datorită procesului de învăţare supervizată, această limită poate fi depăşită. Spre exemplu, în cazul unui vector de intrare în reţea al cărui element maxim nu se află situat pe poziţia clasei corecte, acesta poate activa la ieşire celula corectă tocmai datorită unui proces de învăţare adecvat (în care reţelei i s-au mai livrat exemple asemănătoare). Această limită maximă a acurateţei clasificării este corectă doar pentru metode de agregare neadaptive (postclasificare triviala) ale clasei optimale. Cu acest adaus, limita teoretică rămâne corectă şi devine acum clar de ce algoritmul neuronal (adaptiv) o poate depăși și o chiar depăşeşte!

Rezultatul obţinut de acest metaclasificator pentru acurateţea clasificării de **99.40%** este cel mai bun rezultat obţinut în toate experimentele efectuate în prezenta lucrare (pentru cazul antrenării şi testării pe seturi diferite).

Rezultatele prezentate în secţiunea 6.4.2 au fost prezentate şi în [Cret10_b].

Partea IV-a. Concluzii

"An expert is a man who has made all the mistakes which can be made, in a narrow field."

N. Bohr

7 Concluzii

7.1 *Contribuţii originale ale autorului*

Lucrarea prezintă munca autorului în domeniul clusteringului şi clasificării de documente, în special în clasificarea / clusteringul documentelor text şi a celor web. Din contribuţiile autorului în acest domeniu se pot remarca următoarele:

În **Capitolul 1** autorul realizează o trecere în revistă a acestei teze. Autorul evidenţiază procesul de clasificare automată a documentelor cu toate părţile componente şi evidenţiază părţile în care autorul şi-a adus contribuţii.

În **Capitolul 2** autorul realizează o sinteză a metodelor consacrate în domeniul extragerii de cunoştinţe din baze de date. Se pune accentul pe metodele aplicate documentelor text şi documentelor web. Autorul prezintă critic, comparativ aspecte generale referitoare la învăţarea supervizată şi cea nesupervizată şi implicaţiile pentru analiza clasificării şi cea a clusteringului. Se prezintă metricile utilizate de către algoritmi pentru calcularea similarităţii sau disimilarităţii dintre documente. Pentru evaluarea rezultatului algoritmilor de clasificare/clustering sunt prezentate şi măsurile de evaluare internă şi externă; de asemenea se prezintă seturile de date utilizate pentru algoritmii de clasificare în etapele de antrenare şi respectiv testare precum şi seturile de date RSS utilizate în algoritmii de clustering.

Capitolul 3 conţine o prezentare originală a unei posibile taxonomii a algoritmilor de clustering. Sunt prezentate diferite tipuri de algoritmi de clustering pentru fiecare categorie în parte, fiind prezentaţi detaliat algoritmii cei mai reprezentativi; algoritmii HAC (Hierarchical Agglomerative Clustering) şi K-Medoids au fost prezentaţi în detaliu, cei doi fiind aleşi pentru efectuarea experimentelor în clusteringul documentelor text; este exemplificat într-un spaţiu bidimensional modul de rulare al acestor algoritmi.

În **Capitolul 4** sunt prezentate cercetările autorului şi contribuţiile originale în domeniul clusteringului. În acest capitol autorul a realizat o cercetare care a avut ca obiectiv compararea celor două modele de reprezentare a datelor text: modelul VSM (Vector Space Model) şi respectiv modelul STDM (Suffix Tree Document Model), mai puţin utilizat în reprezentarea documentelor text. Ipoteza ştiinţifică porneşte de la ideea că reprezentarea STDM în ceea ce priveşte reprezentarea documentelor de tip text include şi ordinea cuvintelor din fraze nu doar cuvintele din document. Astfel, în mod implicit, acest model conţine şi câteva elemente de „semantică" a documentului. iar prin reprezentarea arborescent-ierarhica a documentului, se apropie cumva implicit de o reprezentare (mai) semantică, ceea a adus îmbunătăţiri algoritmilor de clustering. Autorul a decis din raţiuni de complexitate a calculului reprezentarea tot a câte două documente, calcularea distanţei dintre cele două documente pe baza unei metrici şi apoi obţinerea unei matrice de distanţe. Această matrice de distanţe a fost apoi utilizată în cadrul celor doi algoritmi de clustering aleşi – HAC şi *k*-Medoids.

De asemenea autorul a propus pentru modelul STDM o metrică originală NEWST care s-a dovedit a fi superioară celor existente în literatura de specialitate. Media acurateţei obţinută pe toate seturile de date S01-S07 obţinută de către metrica NEWST a fost de **87.23%**.

În **Capitolul 5** se face o prezentare a unei taxonomii pentru algoritmii de clasificare. Sunt prezentaţi algoritmi de clasificare caracteristici pentru fiecare categorie şi unele rezultate din faza de testare a acestora. Clasificatorul de tip Bayes este ales de autor pentru a fi folosit pe seturile de date Reuters şi de aceea se prezintă partea matematică aferentă acestui clasificator. Se prezintă rezultatele experimentelor realizate cu acest clasificator pe baza de date Reuters 2000, pentru a arăta că adaptarea acestui tip de clasificator în sensul de a utiliza uniformizarea (normalizarea) lui Laplace, îl face fezabil să fie utilizat la clasificarea de vectori de dimensiune foarte mare. Din păcate nu s-a reuşit modificarea acestuia, astfel încât să lucreze cu documente clasificate în mai multe categorii, cum sunt cele din baza de date Reuters. Astfel se consideră că un document poate să aparţină la mai multe clase, iar clasele astfel create nu trebuie să fie strict disjuncte, ele putând să fie suprapuse. Dacă s-a eliminat posibilitatea existenţei claselor suprapuse acurateţea de clasificare este de **75.89%.** De asemenea, la finalul capitolului cinci sunt prezentate noţiuni generale legate de metaclasificatori şi utilizarea acestora în îmbunătăţirea rezultatelor finale ale clasificării.

În **Capitolul 6** se prezintă rezultatele cercetării privind proiectarea metaclasificatorilor şi posibilităţi de îmbunătăţire a acurateţei de clasificare realizată de metaclasificatorul prezentat iniţial în [Mora07]. Sunt propuse unele soluţii de îmbunătăţire a rezultatelor şi se prezintă 3 metaclasificatori adaptivi şi neadaptivi originali care se dovedesc superiori.

Astfel metaclasificatorii neadaptivi creaţi pornesc de la ideea că în loc să contorizeze clasa prezisă de fiecare clasificator în parte, cum ar fi în cazul „vot majoritar" (MV), însumează simplu toate valorile generate de către clasificatoare pentru fiecare clasă în parte. În diferite experimente acele valori au fost apoi ponderate cu valori alese experimental. Cel mai bun rezultat a fost obţinut când s-a utilizat ponderarea liniară cu pasul de „0,5" acurateţea de clasificare fiind de **87.20%.** Fiind o problemă de „Design Space Exploration" pentru găsirea ponderilor optime s-a utilizat într-o nouă abordare un algoritm genetic. Astfel acurateţea de clasificare a metaclasificatorului s-a îmbunătăţit ajungând în medie la **88.37%.** Cel mai bun rezultat al acestui metaclasificator a fost obţinut într-un experiment folosind metoda Gauss de selecţie a cromozomilor, acurateţea de clasificare ajungând la **88.55%.**

Pentru metaclasificatorii adaptivi introducerea în cadrul metaclasificatorului prezentat în [Mora07] a clasificatorului Bayes a dus la creşterea limitei maxime a acurateţei aceasta ajungând la **98.63%.** Contribuţia autorului a fost modificarea adusă alegerii clasificatorului care va fi utilizat de metaclasificator pentru a clasifica un anumit document şi a clasei pe care o prezice acesta. Astfel în cazul SBED s-a obţinut o acurateţe a clasificării de **93.32%,** iar în cazul SBCOS s-a obţinut o îmbunătăţire de la 93.10% la **93.87%.**

În cazul metaclasificatorului hibrid contribuţiile autorului s-au concretizat prin realizarea unei componente, considerată ca fiind etapa de preclasificare, realizată din metaclasificatorul (selector) neadaptiv şi o componentă nouă, considerată ca fiind etapa de postclasificare, adaptivă, realizată dintr-o reţea neuronală de tip backpropagation. Cele mai bune rezultate s-au obţinut folosind o reţea neuronală cu **192** de neuroni pe stratul ascuns, acurateţe de clasificare ajungând la **99.40%** pe setul de antrenare.

Prezenta teză se încheie cu **Capitolul 7** care este dedicat prezentării ideilor care se desprind din aspectele teoretice şi practice ale cercetărilor efectuate şi care sintetizează contribuţiile personale aduse în această lucrare, precum şi perspectivele de cercetare.

7.1.1 Contribuţii originale în problema metaclasificatorilor

Contribuţii în realizarea metaclasificatorilor neadaptivi

Autorul a creat o serie de metaclasificatori neadaptivi care folosesc diferite procedee pentru ponderarea valorilor generate de către fiecare clasificator în parte, cu scopul de a îmbunătăţi acurateţea finală a clasificării. Astfel s-a creat un metaclasificator care, în loc să contorizeze clasa prezisă de fiecare clasificator în parte, cum ar fi în cazul „vot majoritar" (MV), va însuma simplu toate valorile generate de către clasificatoare pentru fiecare clasă în parte. Autorul a recurs la această abordare deoarece a observat că clasa care apare uneori pe poziţia a doua la majoritatea clasificatoarelor, dar foarte aproape ca si valoare de clasa de prima poziţie, este, în realitate, clasa corectă (soluţie de tip „Eurovision"). Rezultatele obţinute de acest metaclasificator sunt mai bune decât votul majoritar, dar nu semnificativ. Autorul a prezentat o serie de experimente care încearcă diferite valori pentru a pondera valorile fiecărei clase din vectorii generaţi de către clasificatori. Aceste valori ponderează vectorii, în funcţie de ordinea obţinută de fiecare clasă în cadrul vectorului. Cel mai bun rezultat obţinut a fost de **301** documente incorect clasificate, ceea ce reprezintă o acurateţe a clasificării de **87,20%.** Acest rezultat s-a obţinut când s-a utilizat ponderarea liniară cu pasul de „0,5".

O altă contribuţie originală adusă de autor a fost utilizarea de algoritmi genetici pentru rezolvarea problemei „Design Space Exploration": găsirea acelor ponderi care, utilizate în metaclasificatorul neadaptiv prezentat mai sus, să ducă la o îmbunătăţire substanţială a acurateţei de clasificare. Astfel, autorul a creat un metaclasificator original cu ponderi calculate, folosind algoritmi genetici. Rezultatele obţinute de metaclasificatorul cu ponderi calculate s-au îmbunătăţit în medie, cu **1.16%** în cazul utilizării metodei „Roulette Wheel" de selecţie a cromozomilor, acurateţea de clasificare ajungând în medie, la **88,36%.** În cazul utilizării metodei de selecţie Gauss pentru selecţia cromozomilor dintr-o populaţie, îmbunătăţirea a fost de 1**.17%**, ajungând în medie la **88.37%.** Cel mai bun rezultat al acestui metaclasificator a fost obţinut într-un experiment folosind metoda Gauss de selecţie a cromozomilor, acurateţea de clasificare ajungând la **88.55%.** Pentru metaclasificatorii neadaptivi propuşi, acesta reprezintă cel mai bun rezultat.

Contribuţii pentru metaclasificatorii adaptivi

Sunt prezentate experimentele efectuate pentru a analiza dacă este sau nu fezabilă introducerea unui clasificator de alt tip în cadrul metaclasificatorului. Pe baza datelor de test obţinute de clasificatorii de tip SVM prezentaţi în [Mora07], s-a testat dacă un clasificator de tip Bayes ar putea aduce îmbunătăţiri asupra metaclasificatorului, în cazul clasificării de documente text. Problemele care apar la clasificarea documentelor text sunt în primul rând dimensiunea foarte mare a vectorilor de reprezentare a documentelor şi în al doilea rând problema existenţei claselor suprapuse. Dimensiunea foarte mare a vectorilor de reprezentare face ca nu foarte mulţi algoritmi de învăţare automată să se preteze la asemenea probleme, datorită complexităţii calculelor care apar şi a timpilor de învăţare foarte mari. În cazul bazei de date Reuters 2000 pe care s-au făcut experimente, documentele sunt clasificate în mai multe clase, ceea ce face posibilă existenţa de clase suprapuse chiar şi în totalitate, făcând imposibilă învăţarea pentru majoritatea algoritmilor de clasificare automată. Autorul acestei teze a prezentat problemele care au apărut în cazul metaclasificatorului prezentat în [Mora07], precum şi faptul că selectarea clasificatorilor pe baza celui mai bun rezultat întors poate introduce anumite limitări. Aşa cum s-a prezentat încă din acea lucrare, metaclasificatorul avea o limită maximă la care putea ajunge de 94.02%, având un număr de 136 documente care nu puteau fi clasificate corect de niciunul din clasificatoarele selectate. În scopul de a depăşi limitările prezentate în [Mora07] în această teză s-au analizat cele 136 documente şi s-a constatat că există clasificatori SVM prezentaţi în lucrarea amintită care ar fi reuşit să clasifice corect acele documente, dar nu au fost incluşi în metaclasificator. De asemenea, autorul acestei teze prezintă câteva experimente comparative

între clasificatorii de tip SVM şi cel de tip Bayes, realizate pe toate seturile de date prezentate. Ca şi timp de antrenare şi testare, clasificatorul Bayes obţine cel mai bun timp de 1.7s comparativ cu SVM care, în medie, obţine un timp de 1.8s. În cazul testării clasificatorul Bayes pe setul T2 (cel care conţine doar 136 documente), s-au obţinut rezultate încurajatoare. În urma testelor efectuate s-a observat că, deşi acurateţea totală a clasificatorului Bayes este mai slabă decât cea obţinută de SVM, acesta totuşi a reuşit să clasifice corect 104 documente din cele 136, chiar dacă a fost antrenat pe setul A1.

Pe baza rezultatelor prezentate mai sus, s-a decis dezvoltarea unui metaclasificator nou prin **introducerea clasificatorului de tip Bayes** în metaclasificatorul din [Mora07], obţinând astfel o îmbunătăţire semnificativă a limitei superioare la care poate ajunge metaclasificatorul. Astfel, aceasta a crescut de la 94.21% în cazul folosirii a 8 clasificatoare SVM la **98.63%** în cazul folosirii celor 8 clasificatoare SVM plus clasificatorul Bayes. Rezultatele obţinute pentru toate cele trei modele de metaclasificatori, vot majoritar (MV), selecţie pe baza distanţei euclidiene (SBED) şi selecţie pe bază de cosinus (SBCOS) sunt rezumate în continuare. În cazul MV s-a obţinut o acurateţe a clasificării de doar **86.09%,** cu **0.29% mai mică** decât în cazul folosirii doar a 8 clasificatori SVM. În cazul SBED, prin modificările aduse, noul metaclasificator a obţinut rezultate chiar mai slabe, scăzând în medie de la **92.04%** (metaclasificatorul cu 8 clasificatori SVM) la **90.38%.** În cazul SBCOS, acurateţea de clasificare a metaclasificatorului cu 9 clase **a crescut** la **93.10%,** de la **89.74%** cât era la cel cu 8 clasificatori SVM.

O altă contribuţie a autorului a fost modificarea adusă alegerii clasificatorului care va fi utilizat de metaclasificator pentru a clasifica un anumit document. Astfel, în cazul în care există suspiciunea că clasa pe care o va prezice clasificatorul selectat nu va fi cea corectă, metaclasificatorul să prezică următoarea clasă din lista de clase, dacă aceasta este suficient de apropiată, ca şi valoare a clasificării obţinute, de prima clasă prezisă. Aceste modificări au dus la o îmbunătăţire substanţială a rezultatelor metaclasificatorului cu 9 clasificatori. Am făcut experimente doar cu acesta, deoarece doar în acest caz se putea ajunge la o acurateţe maximă de 98.63%. În cazul SBED s-a obţinut o acurateţe a clasificării de **93.32%,** iar în cazul SBCOS s-a obţinut o îmbunătăţire de la 93.10% la **93.87%.**

Contribuţii în realizarea unui metaclasificator hibrid

Ideea de la care a pornit autorul a fost de a construi acest metaclasificator format din două componente. O componentă, considerată ca fiind etapa de preclasificare, realizată din metaclasificatorul (selector) neadaptiv şi o componentă nouă, considerată ca fiind etapa de postclasificare, adaptivă, realizată dintr-o reţea neuronală de tip backpropagation. Autorul a prezentat elementele necesare pentru dezvoltarea unei reţele neuronale de tip backpropagation, adaptată pentru funcţionarea în acest context. Parametrii reţelei, care au fost modificaţi în cadrul experimentelor, sunt numărul de neuroni de pe stratul ascuns şi coeficientul de învăţare al reţelei. Algoritmul prezentat se aplică reţelelor feed-forward care conţin 2 niveluri de unităţi cu neuroni cu funcţia de activare sigmoidală, fiecare unitate de pe un nivel fiind conectată la toate unităţile de pe nivelul anterior. Deoarece reţeaua neuronală prezentată este o reţea cu învăţare supervizată, a avut nevoie de o etapă de antrenare a acesteia. Autorul a prezentat experimente realizate, utilizând seturi diferite pentru antrenare şi testare. În acest caz s-au folosit valori descrescătoare ale coeficientului de învăţare, oprindu-se învăţarea la anumite etape, scăzând coeficientul de învăţare şi apoi continuând învăţarea. Doar în momentul în care s-a redus şi coeficientul de învăţare, s-a reuşit antrenarea reţelei până la o valoare mică a erorii totale de antrenare (medie 0.017 per exemplu de antrenament). Cele mai bune rezultate (**99.40%** acurateţe de clasificare) s-au obţinut folosind o reţea neuronală cu **192** de neuroni pe stratul ascuns. Totuşi, din punct de vedere al numărului de rezultate bune obţinute pe parcursul antrenării, optimul a fost atins pentru cazul utilizării unei reţele cu **176** de neuroni pe stratul ascuns (chiar dacă în final doar s-a apropiat de eroarea de antrenare minimă). În urma experimentelor efectuate s-a putut observa că introducerea unei reţele neuronale în cadrul metaclasificatorului face ca acesta să se adapteze

mult mai bine la documentele care trebuie clasificate, reuşind astfel să clasifice şi documentele cu problemă pe care metaclasificatorii adaptivi şi neadaptivi prezentaţi anterior nu au reuşit să le „învețe”. Acest nou metaclasificator a reuşit **să depăşească** şi limita maximă de **98.63%** la care ar fi putut ajunge „teoretic” (din cauza algoritmului neadaptiv de agregare finală a clasei optime) clasificatorii incluşi în cadrul metaclasificatorului.

7.1.2 Contribuţii originale în problema clusteringului

În cercetările efectuate s-a analizat îmbunătăţirea adusă algoritmilor de clustering în cazul utilizării reprezentării STDM a documentelor text. În abordarea autorului se va folosi această reprezentare pentru a calcula similaritatea/disimilaritatea între oricare două documente pentru a reduce dimensiunea arborelui de sufixe care reprezintă documentele din setul de documente.

În cadrul acestui capitol, o altă contribuţie originală a autorului a constat în elaborarea unei noi metrici (NEWST), utilizată în calculul similarităţii între două documente reprezentate prin modelul STDM. Din experimentele efectuate se poate constata clar că această măsură a obţinut rezultate convingătoare. Astfel, această metrică aplicată modelului STDM de reprezentare în cazul algoritmului de clustering HAC, a obţinut pe seturile de date S01-S07 o îmbunătăţire a acurateţei clusteringului de **34.84%** faţă de metrica Jaccard utilizată cu reprezentarea VSM. Acurateţea medie calculată pe seturile S01-S07 pentru metrica NEWST cu modelul STDM a fost de **87.23%,** faţă de media de **52.39%** obţinută de metrica Jaccard aplicată reprezentării VSM. Seturile cu care s-a lucrat în acest capitol au fost create atât folosind un algoritm de stemming pentru extragerea rădăcinii cuvintelor din document cât şi fără utilizarea unui algoritm de stemming. În cadrul acestor experimente s-a observat că, în cazul algoritmului HAC, aplicarea algoritmului de extragere a rădăcinilor cuvintelor nu a îmbunătăţit rezultatele clusteringului.

Pentru verificarea metricii NEWST propuse, autorul a repetat testele cu un algoritm partiţional K-Medoids. Şi în cazul acestui algoritm de clustering, metrica NEWST a obţinut o îmbunătăţire cu **5.04%** faţă de metrica Jaccard aplicată modelului VSM. Acurateţea medie a clusteringului pentru NEWST cu algoritmul *k*-Medoids pe seturile S01-S07 a fost de **84.20%** iar pentru metrica Jaccard aplicată modelului VSM a fost de **79.16%.** În cazul algoritmului *k*-Medoids, extragerea rădăcinii cuvintelor a dus la o îmbunătăţire a rezultatelor pentru toate metricile utilizate.

7.1 *Concluzii sintetice şi dezvoltări ulterioare*

Prezenta lucrare cu contribuţiile originale care au fost prezentate, poate fi utilă în cazul în dezvoltării unui sistem pentru clasificarea automată a documentelor web sau a fluxurilor de ştiri RSS, în special gruparea automată a paginilor de web sau a snipett-urilor returnate de motoarele de căutare clasice. Rearanjarea şi gruparea poate fi făcută pe baza conţinutului acestor paginilor şi nu doar pe un rezumat al acestor pe care îl oferă creatorul acestora şi care nu în toate cazurile reflectă în totalitate ce se găseşte în acele pagini.

De asemenea, această teză prezintă soluţii pentru utilizarea celor două modele de reprezentare a datelor STDM şi VSM. Prin reprezentarea STDM şi metrica NEWST în cazul algoritmilor ierarhici rezultatele clusteringului pe documente de tip text se îmbunătăţesc simţitor. Totuşi nu există o soluţie generală în cazul clusteringului. Fiecare algoritm trebuie adaptat problemei care trebuie rezolvată. Utilizarea modelului STDM cu algoritmi partiţionali duce la timpi de execuţie foarte mari, după cum s-a arătat în lucrare.

Ar fi utilă combinarea clasificatorilor utilizând metode neadaptive, adaptive şi combinaţii între ele, pentru îmbunătăţirea rezultatelor fără ca timpul de lucru să crească semnificativ.

În cercetări viitoare voi testa să combinarea metodelor de clustering cu metodele de clasificare, folosind documente etichetate şi neetichetate într-un algoritm de clasificare hibrid. Acest algoritm ar utiliza un set mic de date în prealabil etichetate, care să ghideze algoritmul de clustering care se antrenează, folosind un set mare de date neetichetate.

O altă idee este de a schimba reprezentarea STDM pentru algoritmii de clustering, astfel încât să se regăsească şi informaţia semantică conţinută în text, care în momentul de faţă, este prezentă doar prin păstrarea ordinii cuvintelor din propoziţie.

O problemă majoră a tuturor algoritmilor de clustering şi clasificare este utilizarea lor în situaţii reale este greoaie aceştia trebuind să fie adaptaţi specific. De exemplu, algoritmii de clustering tind să formeze clusteri foarte mari în detrimentul altora, care pot să conţină documente foarte puţine. Aceasta problemă apare deoarece documentele din aceeaşi categorie au puţine cuvinte comune şi atunci multe documente sunt grupate împreună, pentru că fiecare conţine câteva cuvinte comune cu alt document din acea categorie. Chiar dacă documentele care vorbesc despre aceeaşi problemă ar trebui să folosească aceleaşi cuvinte, în realitate sunt folosite foarte multe sinonime, şi abordarea clasică nu consideră sinonimele ca fiind cuvinte comune. Abordările pur computaţionale ale acestei probleme nu vor duce la îmbunătăţiri spectaculoase. Ca dezvoltare ulterioară, doresc să testez oportunitatea introducerii sensurilor cuvintelor în reprezentarea documentelor şi care implică folosirea de algoritmi de dezambiguizare (Word Sense Disambiguation), eventual cu ajutorul WordNet unor algoritmi de dezambiguizare a sensurilor cuvintelor, şi introducerea unor măsuri semantice în modelul STDM.

Având în vedere faptul că exploatarea sinergismului mai multor algoritmi simpli de clasificare poate duce la rezultate foarte bune în clasificarea automată a documentelor text, îmi propun ca pe viitor să realizez paralelizarea calculului pentru metaclasificatorul hibrid, astfel încât timpii necesari clasificării să se reducă semnificativ (în cadrul laboratorului de cercetare ACAPS din cadrul ULB Sibiu există facilităţi High Performance Computing care permit implementarea cu succes a acestor idei – v. http://acaps.ulbsibiu.ro/index.php/en/). Pentru aceasta însă este necesară rescrierea aplicaţiilor software implementate în vederea exploatării caracteristicilor unui model de programare concurentă.

"Equations are more important to me, because politics is for the present, but an equation is something for eternity."

A. Einstein

8 Glosar de termeni

Termenii din acest glosar sunt definiţi în ideea de a exprima cât se poate de bine contextul în care apar în prezenta teză. Nu am avut pretenţia unei definiri exhaustive a acestor termeni, aşa cum apar ei în ştiinţa şi ingineria calculatoarelor.

Atribut	reprezintă rădăcina cuvântului extrasă din document
card(X)	cardinalul mulţimii *X* şi reprezintă numărul de elemente al mulţimii
Clasă	reprezintă categoria propusă de algoritm pentru un exemplu dat
Clasificare	gruparea documentelor cu ajutorul unui algoritm care parcurge iniţial etapa de antrenare şi apoi testează „învăţarea” grupând date de testare în categorii predefinite
Clustering	gruparea documentelor fără a avea la dispoziţie un set de antrenare
Cuvinte de stop	un set de cuvinte care apar foarte des în documente şi sunt irelevante pentru clustering sau clasificare a documentelor.
Data mining	procesul de extragere a cunoştinţelor din date organizate în baze de date
Data warehouse	o bază de date care este proiectată pentru interogări, analiză şi vizualizări ale tendinţelor, predicţii ale evoluţiei indicatorilor pe baza unui motor de procesare analitică online (OnLine Analitical Processing OLAP).
Etichetă	reprezintă categoria preluată din setul de date pentru un exemplu dat.
Flux rss	documente scurte în format xml care descriu succint un anumit eveniment
Information Retrieval (IR)	procesul de extragere de informaţii din documente de tip text
Învăţare nesupervizată	algoritmii nu au la dispoziţie în etapa de antrenare etichete pentru documente
Învăţare supervizată	algoritmii au la dispoziţie în etapa de antrenare etichete pentru documente
Metaclasificare	o tehnică de combinare a mai multor clasificatori care lucreaza prin intermediul unui selector, pentru a îmbunătăţi

	acurateţea de calsificare
Normalizare	o tehnică de a reprezenta valorile în acelaşi domeniu
Outliner	document care reprezintă zgomot pentru setul de date
Pattern	şablon, porţiuni de text care se repetă
Precizie	este procentajul din documentele regăsite care sunt într-adevăr relevante
Recall	este procentajul de documente care sunt relevante pentru interogare şi care de fapt au fost regăsite
Seturi de date	setul de documente utilizate pentru antrenarea şi testarea algoritmilor
Stemming.	procesul de extragere a rădăcinii cuvintelor; în prezenta teză am utilizat algoritmul de stemming Porter.
Suffix Tree	arbore de sufixe în care nodurile conţin documente care au comune propoziţia de la rădăcina arborelui până la nodul respectiv
Text mining	procesul de extragere a cunoştinţelor din date organizate în fişiere de tip text
Zgomot	un document care reprezintă o valoare „aberantă” pentru un algoritm de clasificare sau clustering (e greu de definit riguros!)

Books serve to show a man that those original thoughts of his aren't very new at all.
Abraham Lincoln

9 Referinţe bibliografice

9.1 *Bibliografie*

[Abra03] Abraham, A., Ramos, V., *Web Usage Mining Using Artificial Ant Colony Clustering and Genetic Programming*. Proc. of the Congress on Evolutionary Computation (CEC 2003), Canberra, pp. 1384-1391, IEEE Press. 2003

[Agra93] Agrawal, R.; Imielinski, T.; Swami, A., *Database mining: a performance perspective in Knowledge and Data Engineering*, IEEE Transactions Volume: 5 Issue:6, 1993

[Anke99] Ankerst, M., Breuning, M., Kriegel, H.-P., Sander, J., *Oprics: Ordering points to identify the clustering structure.* In. Proc. 1999 ACM-SIGMOD, Int. Conf. Management of Data, Philadelphia, 1999

[Bart98] Bartlett, P., Shawe-Taylor, J. *Generalization Performance of Support Vector MAchines and Other Pattern Classifiers*, in "Advances in Kernel Methods, Support Vector Learning", MIT Press, Cambridge, 1998

[Bate08] Batet, M., Valls, A., Gibert, K. *Improving classical clustering with ontologies*, In Proc. IASC 2008, Yokohama, 2008

[Benn89] Bennett, K. P., Demiriz A., *Semi-supervised support vector machines*.In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, Advances in Neural Information Processing Systems, pages 368-374, Cambridge, MA, 1998. MIT Press.

[Berk06] Berkhin, P: *A Survey of Clustering Data Mining Techniques*, Kogan, Jacob;Nicholas, Charles; Teboulle, Marc (Eds.) Grouping Multidimensional Data, Springer Press, pp. 25-72 (2006)

[Bish95] Bishop, C., *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press, 1995.

[Brad98] Bradley, P., Fayyad, U., *Refining initial points for k-means clustering*, In Proceedings of the Fifteenth International Conference on Machine Learning (ICML) San Francisco, AAAI Press, 1998

[Brea06] Breazu, M., *Tehnici fractale şi neuronale în compresia de imagini*, Editura Universităţii „Lucian Blaga” din Sibiu, ISBN 978-973-739-251-0, 2006

[Bruz04] Bruzzone L., Cossu R., Vernazza G., *Detection of land-cover transitions by combining multidate classifiers*, Pattern Recognition Letters, 25(13) p. 1491-1500, 2004.

[Burg98] Burges, C. J. C., *A tutorial on support vector machines for pattern recognition.* Data Mining and Knowledge Discovery, 2(2):121-167, 1998.

[Cama01] Camazine, S., Deneubourg, J.,-L., Franks, N., R., Sneydd, J., Theraulaz, G. Bonabeau, E., *Self-Organization in Biological Systems*. Princeton University Press, 2001.

[Chee96] Cheeseman, P., Stutz, J., *Bayesian classification (AutoClass): theory and results*. In Advances in knowledge discovery and data mining, Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy (Eds.). American Association for Artificial Intelligence, Menlo Park, CA, USA 153-180., 1996

[Cret11_a] **Creţulescu, R**., Morariu, D., Breazu, M., Vintan L. – *Using Genetic Algorithms for Weights Space Exploration in an Eurovision-like weighted Metaclassifier*, The second international conference in Romania of Information Science and Information Literacy, ISSN 2067-9882, April 2011.

[Cret11_b] **Creţulescu, R**., Morariu, D., Vintan L. – *Clustering Text Documents: An Overview*, (Accepted) "Acta Universitatis Cibiniensis", Technical Series, "Lucian Blaga" University of Sibiu, 2011

[Cret10_b] **Creţulescu, R**., Morariu, D, Vinţan, L, Coman, I. – *An Adaptive Metaclassifier for Text document*, 16th International Conference on Information Systems Analysis, pp. 372-377, ISBN-13: 978-1-934272-86-2(Collection), ISBN-13: 978-1-934272-88-6(Volume II) ,Florida, USA, 2010 **indexată (ISI) Thomson Reuters**

[Cret09_a] **Creţulescu R**., Morariu, D., Vintan, L., *Eurovision-like weighted Non-Adaptive Metaclassifier for Text Documents*, The 8th RoEduNet International Conference, Galaţi, Romania, 2009, **indexată (ISI) Thomson Reuters**

[Cret09_b] **Creţulescu, R**, Referat de doctorat nr. 3, *Metaclasificator bazat pe reţea neuronală*, 1.11.2009, ULB Sibiu (conducător ştiinţific: Prof. univ. dr. ing. Lucian N. Vinţan)

[Cret08] **Creţulescu, R**, Referat de doctorat nr. 2, *Support Vector Machine versus Bayes Naive*, 1.11.2008, ULB Sibiu (conducător ştiinţific: Prof. univ. dr. ing. Lucian N. Vinţan)

[Cret07] **Creţulescu, R**, Referat de doctorat nr. 1, *Clusteringul documentelor text*, 1.11.2007, ULB Sibiu (conducător ştiinţific: Prof. univ. dr. ing. Lucian N. Vinţan)

[Cris00] Cristianini, N. Swawe-Taylor, J. *An introduction to Support Vector Machines*, Cambridge University Press, 2000

[Davi79] Davies, D.,L., Bouldin, D. W., *A cluster separation measure*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1:224–227, 1979.

[Domi96] Domingos, P, *Using Partitioning to Speed Up Scientific-to-General Rule Induction*. in Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Model, pp29-34, AAAAI Press, 1996

[Dori00] Dorigo, M., Bonabeau, E., Theraulaz, G., *Ant Algorithms and stigmergy*, Future Generation Computer Systems Vol.16, 2000.

[Fish87] Fisher, D., H., *Knowledge Acquisition Via Incremental Conceptual Clustering*, Machine Learning Journal, Vol 2, Springer, Netherlands, 1987

[Fre1906] Fréchet, M., *Sur quelques points du calcul fonctionnel,* PhD Thesis, in Rendiconti del Circolo Mathematico di Palermo", volume 22, pag. 1-74, 1906.

[Gabr04] Gabrilovich, E., Markovitch S., *Text Categorization with Many Redundant Features Using Aggressive Feature Selection to Make SVM Competitive with C4.5*, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada,2004.

[Ghos05] Ghosh, A., Jain, L.C. *Evolutionary Computation in Data Mining*, Springer Verlag Berlin, Heidelberg, 2005

[Guha98] Guha, S. Rastogi, R., Shim, K., *CURE: an efficient clustering algorithm for large databases*. In Proceedings of the 1998 ACM SIGMOD international conference on Management of data (SIGMOD '98), Ashutosh Tiwary and Michael Franklin (Eds.). ACM, New York, NY, USA1998

[Han01] Han, J., Kamber, M., *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2001

[Hand07] Handl, J., Meyer, B., *Ant-based and swarm-based clustering,* Swarm Intelligence Journal, Springer, New York, 2007

[Hart75] Hartigan, J. A., *Clustering Algorithms*, New York: John Wiley & Sons, Inc, 1975

[Hart79] Hartigan, J. A., Wong, M., Algorithm as 136: *A k-means clustering algorithm*, Journal of the Royal Statistical Society. Series C (Applied Statistics), London, 1979

[Hayk94] Haykin, S., *Neural Networks: A comprehensive Foundation*, MacMillan College, New York, 1994

[Hebb49] Hebb, D.O., *The Organization of Behavior*, John Wiley & Sons, New York, 1949

[Hols94] Holsheimer, M., Siebes, A., *Data mining: The search for knowledge in databases*. Technical Report CS-R9406, CWI , Netherlands, 1994

[Hoth03] Hotho, A., Staab, S. Stumme,G., *Ontologies Improve Text Document Clustering*, IEEE International Conference on, p. 541, Third IEEE International Conference on Data Mining (ICDM'03), 2003

[Hsu03] Hsu, C., Chang C., Lin, C., *A Practical Guide to Support Vector Classification*, Department of Computer Science and Information Engineering National Taiwan University, 2003 (http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf - accesat iunie 2011)

[Ian00] Ian H., Witten,E. F., *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementation*, Morgan Kaufmann Press, 2000

[Ilan10] Ilango M.,R., Mohan, V., *A Survey of Grid Based Clustering Algorithms*, International Journal of Engineering Science and Technology, Vol. 2(8), 2010

[Jaeg08] Jaeger, S., Huanfeng, M., Dörmann, D., *Combinig Calssifiers with Informational Confidence*, Studies in Computational Intelligence (SCI) 90, pag. 163-191, 2008

[Jain88] Jain, A., K., Dubes, R.,C. *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ. 1988

[Jain90] Jain, A. Murty, M. N., Flynn, P. J., *Data Clustering: A Review*, ACM Computing Surveys, Vol. 31(3), pp. 264-323 (1999)

[Jain96] Jain, A., Mao, J., Mohiuddin, K.M., *Artificial Neural Networks: A Tutorial*, Journal of IEEE Computational Science and Engineering, pp. 31-44, 1996

[Janr11] Janruang, J. Guha, S., *Semantic Suffix Tree Clustering*, In Proceedings of 2011 International Conference on Data Engineering and Internet Technology (DEIT 2011), Bali, Indonesia, 2011.

[Kauf87] Kaufman, L., Rousseeuw, *P. J., Clustering by means of medoids*, in Statistical Data Analysis based on the L, Norm, edited by Y. Dodge, Elsevier/North-, Holland, Amsterdam, 1987

[Kauf90] Kaufman, L. and Rousseeuw, P.J. *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley-Interscience, New York (Series in Applied Probability and Statistics), 1990

[Kung93] Kung S.Y., *Digital Neural Networks*, Prentice Hall, New Jersey, 1993

[Labr03] Labroche, N., Monmarché, N., Venturini G., *AntClust: Ant Clustering and Web Usage Mining*, In Genetic and Evolutionary Computation — GECCO 2003 Lecture Notes in Computer Science, Volume 2723/2003, 201, 2003.

[Lanc67] Lance, G. N., Williams, W. T., *A general theory of classification sorting strategies*, Computer Journal, 9 p 373-386, 1967

[Leig02] Leigh W., Purvis R., Ragusa J. M., *Forecasting the NYSE composite index with technical analysis, pattern recognizer, neural networks, and genetic algorithm: a case study in romantic decision support*, Decision Support Systems 32(4), p. 361-377, 2002.

[MacQ67] MacQueen, J. B., *Some Methods for classification and Analysis of Multivariate Observations*, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297, 1967

[Maim04] Maimon 0. Rokach L., *Ensemble of Decision Trees for Mining Manufacturing Data Sets*, Machine Engineering, vol. 4 Nol-2, 2004.

[Mann08] Manning, C., Raghavan, P., Schütze, H. *Introduction to Information Retrieval*, Cambridge University Press, ISBN 978-0-521-86571, 2008

[Mann09] Manning, C., *An Introduction to Information Retrieval*, Cambridge University Press, 2009

[Mang04] Mangiarneli P., West D., Rampal R., *Model selection for medical diagnosis decision support systems*, Decision Support Systems, 36(3) p. 247-259, 2004.

[Merw03] van der Merwe, D. W., Engelbrecht, A.P., *Data clustering using particle swarm optimization*, In: The 2003 Congress on Evolutionary Computation, CEC '03, Canberra, 2003

[Meye05] Meyer,S., Stein, B., Potthast, M., *The Suffix Tree Document Model Revisited*, Proceedings of the I-KNOW 05, 5th International Conference on Knowlegdge Management, Journal of Universal Computer Science, pp.596-603, Graz, 2005

[Mitc97] Mitchell, T. Machine Learning, McGraw Hill Publishers, 1997

[Mora06_a] Morariu, D., Vintan, L., Tresp, V., Feature Selection Method for an Improved SVM Classifier, Proceedings of the 3rd International Conference of Intelligent Systems (ICIS'06), ISSN 1503-5313, vol. 14, pp. 83-89, Prague, August, 2006.

[Mora06_b] Morariu, D., Vintan, L., Tresp, V., *Metaclassification using SVM classifier for Text Document*, Proceedings of the 3rd International Conference on Machine Learning and Pattern Recognition (MLPR'06), ISSN 1503-5313, vol. 15, pp. 222-227, Barcelona, Spain, October, 2006.

[Mora06_c] Morariu, D.,Vintan, L., Tresp, V., *Feature Selection Methods for an Improved SVM Classifier*, Proceedings of 14th International Conference on of Intelligent Systems (ICIS06), Prague, 2006

[Mora07] Morariu, Daniel - *Contributions to Automatic Knowledge Extraction from Unstructured Data*, PhD Thesis, Sibiu, 2007 (scientific supervisor: Prof. Lucian N. Vinţan, PhD)**.**

[Mora08] Morariu, D., *Text Mining Methods based on Support Vector Machine*, Ed. MatrixRom, Bucureşti, 2008.

[Mora10] Morariu, D., **Cretulescu, R**., Vinţan, L. – *Improving a SVM Metaclassifier for Text Documents by using Naive Bayes*, International Journal of Computers, Communications & Control, Vol. V, No. 3, pp. 351-361, ISSN 1841-9836, E-ISSN 1841-9844, September 2010, **cotată (ISI) Thomson Reuters**

[Mora11] Morariu, D, **Creţulescu, R**., Vinţan, L, Using Suffix Tree Document Representation in Hierarchical Agglomerative Clustering, accepted at International Conference on Intelligent Systems, Paris, November 2011.

[Ng02] Ng, R. T., Han, J., *Clarans: A method for clustering objects for spatial data mining*. IEEE Transactions on Knowledge and Data Engineering (TKDE), 14(5), 2002

[Ng94] Ng, R., Han, J., *Efficient and Effective Clustering Methods for Spatial Data Mining*, Proceedings of International Conference on Very Large Data Bases,Santiago, Chile, Sept. 1994

[Opit99] Opitz, D., Maclin, R., *Popular Ensemble Methods: An Empirical Study*, Journal of Artificial Research, 11, p. 169-198, 1999

[Quin96] Quinlan, J.R., *Bagging, Boosting and C4.5*, In Proceedings of the Thirteenth National Conference on Artificial Intelligence, p. 725-730, 1996

[Rijs80] Rijsbergen C.J. van, Robertson S.E., Porter M. F., *New models in probabilistic information retrieval*. London: British Library. (British Library Research and Development Report, no. 5587). 1980

[Roka05] Rokach, L. *Ensemble Methods for Classifiers, in Data Mining and Knowledge Discovery Handbook*, Maimon, Oded; Rokach, Lior (Eds.), p. 957-980, Springer-Link, 2005

[Salt75] Salton, G., Wong, A., Yang, C. S., *A vector space model for information retrieval*. Communications of the ACM, 18(11), 1975.

[Scho02] Schoslkopf, B., Smola, A., *Learning with Kernels, Support Vector Machines*, MIT Press, London, 2002

[Serr10] Serrano1, S. L., Villena-Román, J., Cristóbal, J. C. G., *DAEDALUS at WebPS-3 2010: k-Medoids Clustering using a Cost Function Minimization*, In Proc. Conference on Multilingual and Multimodal Information Access Evaluation, Padua, 2010

[Tan03] Tan A. C., Gilbert D., Deville Y., *Multi-class Protein Fold Classification using a New Ensemble Machine Learning Approach*, Genome Informatics, 14, p. 206-217, 2003.

[Turn99] Turner, K., Gosh, J., *Linear and Order Statistics Combiners for Pattern Classification*, in Combining Artificial Neural Nets, A. Sharkey (Ed), p. 127-162, Springer-Verlag, 1999

[Vint00] Vinţan N. L., *Arhitecturi de procesoare cu paralelism la nivelul instructiunilor*, Editura Academiei Române, Bucureşti, ISBN 973-27-0734-8, 2000

[Vint07] Vinţan N. L., *Prediction Techniques in Advanced Computing Architectures* (in limba engleza), Editura Matrix Rom, Bucureşti, ISBN 978-973-755-137-5, 2007

[Wang97] Wang, W., Yang, J., Muntz, R., *STING: A statistical information grid approach to spatial data mining.* In Proc. 1997 Int. Conf. Very Large Data Bases, Athens, 1997

[Ward63] Ward, J. H., *Hierachical grouping to optimize an objective function*. J. Am. Statist. Assoc. 58, 236-244, 1963

[Wass89] Wassermann, P.D., *Neural Computing. Theory and Practice*, Van Nostrand Reinhold, 1989

[Wei03] Wei, C. P., Lee, Y. H., Hsu, C. M., *Empirical Comparison of Fast Clustering* In Algorithms for Large Data Sets, In Experts Systems with Applications vol. 24, pp. 351– 363 - 2003

[Wen09] Wen, H., *Web SnippetsClustering Based on an Improved Suffix Tree Algorithm*, Proceedings of FSKD 2009, Sixth International Conference on Fuzzy Systems and Knowledge Discovery, Tianjin, China, 14-16 August 2009

[Zami98] Zamir, O, Etzoni, O., *Web Document Clustering: A Feasibility Demonstration*, Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia, 1998

[Zhan01] Zhang, B., *Generalized k-harmonicmeans – dynamic weighting of data in unsupervised learning*, In Proceedings of the 1st SIAM ICDM, Chicago, 2001

[Zhan96] Zhang, T. Ramakrishnan, R. Livny, M., *BRICH: an efficent data clustering method for very large databases*. In Proceedings 1996 ACM-SIGMOD Int. Conf. Management of Data, Montreal, 1996

[Zhao04] Zhao, Y. Karypis, G., *Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering*, Machine Learning, Vol. 55. No. 3, 2004

9.2 *Webliografie*

[M01] *Metrics for Evaluating clustering algorithms*- http://www.scribd.com/Clustering/d/28924807 - accesat 25.08.2010

[Reut00] Misha Wolf and Charles Wicksteed - *Reuters Corpus*: http://www.reuters.com/researchandstandards/corpus/ lansat în noiembrie 2000, accesat în septembrie 2009

[WEB09] http://www.cs.utexas.edu/~mooney/ir-course/, accesat în ianuarie 2009

[WEKA] http://www.cs.waikato.ac.nz/ml/weka/index.html (ultima accesare în noiembrie 2010)

[IR] http://www.cs.utexas.edu/~mooney/ir-course/doc/index.html (ultima accesare în ianuarie 2007)

[NETK] http://news.netcraft.com/archives/2011/ (ultima accesare în iunie 2011)

[W3C] http://www.w3.org/DOM (ultima accesare în iunie 2011)

[CLU] http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html (ultima accesare în iunie 2011)

10 Sinteza lucrărilor publicate/elaborate de către autor pe problematica tezei de doctorat

A. Articole în jurnale:

Morariu, D., **Cretulescu, R**., Vinţan, L. – *Improving a SVM Metaclassifier for Text Documents by using Naive Bayes*, International Journal of Computers, Communications & Control, Vol. V, No. 3, pp. 351-361, ISSN 1841-9836, E-ISSN 1841-9844, septembrie 2010, **cotată (ISI) Thomson Reuters, factor de impact 0,392**

Morariu, D., **Cretulescu, R**., Vinţan, L. *Vector versus Tree Model Representation in Document Clustering*, trimis la jurnalul **Data & Knowledge Engineering,** Olanda, **cotată (ISI) Thomson Reuters, scor relativ de influenţă: 0,80282**

Creţulescu, R., Morariu, D., Vintan L. – *Clustering Text Documents: An Overview*, "Acta Universitatis Cibiniensis", Technical Series, ISSN 1583-7149, "Lucian Blaga" University of Sibiu, mai 2011

B. Articole în conferinţe

Morariu, D, **Creţulescu, R**., Vinţan, L, *Using Suffix Tree Document Representation in Hierarchical Agglomerative Clustering*, accepted at International Conference on Intelligent Systems, **Paris (Franţa)**, noiembrie 2011.

Creţulescu, R., Morariu, D., Breazu, M., Vintan L. – *Using Genetic Algorithms for Weights Space Exploration in an Eurovision-like weighted Metaclassifier*, The second international conference in Romania of Information Science and Information Literacy, ISSN 2067-9882, aprilie 2011.

Creţulescu, R., Morariu, D, Vinţan, L, Coman, I. – *An Adaptive Metaclassifier for Text document*, 16th International Conference on Information Systems Analysis, pp. 372-377, ISBN-13: 978-1-934272-86-2(Collection), ISBN-13: 978-1-934272-88-6(Volume II) ,**Florida (USA),** 2010 **indexată (ISI) Thomson Reuters**

Creţulescu R., Morariu, D., Vintan, L., *Eurovision-like weighted Non-Adaptive Metaclassifier for Text Documents*, Proceedings of the 8th RoEduNet IEEE International Conference Networking in Education and Research, pp. 145-150, ISBN 978-606-8085-15-9, Galaţi, decembrie 2009 **indexată (ISI) Thomson Reuters**

C. Lucrări elaborate:

Cretulescu, R., Morariu, D., Vinţan, L., *Ongoing Research in Document Classification at the „Lucian Blaga" University of Sibiu*, Conferinţa IDC2011, **Delft (Olanda)**, 5-7 octombrie 2011, am susţinut lucrarea efectiv la Workshop: „News from Projects".

Creţulescu, R, Referat de doctorat nr. 3, *Metaclasificator bazat pe reţea neuronală*, 2009, ULB Sibiu (conducător ştiinţific: Prof. univ. dr. ing. Lucian N. Vinţan)

Creţulescu, R, Referat de doctorat nr. 2, *Support Vector Machine versus Bayes Naive*, 2008, ULB Sibiu (conducător ştiinţific: Prof. univ. dr. ing. Lucian N. Vinţan)

Creţulescu, R, Referat de doctorat nr. 1, *Clusteringul documentelor text*, 2007, ULB Sibiu (conducător ştiinţific: Prof. univ. dr. ing. Lucian N. Vinţan)