Universitatea „Lucian Blaga” din Sibiu
Facultatea de inginerie “Hermann Oberth”
Catedra de calculatoare şi automatizări

Contribuţii la extragerea automată de cunoştinţe din masive de date

Teză de doctorat

(rezumat)

Autor:
Asist. univ. ing. Ionel Daniel MORARIU

Conducător ştiinţific:
Prof. univ. dr. ing. Lucian N. VINŢAN

SIBIU, 2007

Cuprins

1 Introducere şi obiective principale

Tehnicile tradiționale pentru recunoaşterea informațiilor din masivele de date, ca şi metode de indexare a documentelor au devenit inadecvate pentru aplicațiile de căutare în date semistructurate. Datele în format text sunt considerate date semistructurate deoarece conțin o structură organizatorică minimă. De obicei, doar o mică parte din toate documentele disponibile sunt relevante pentru utilizator la un moment dat. Fără a şti ce conține documentul, este dificil să formulăm interogări pentru analiza şi extragerea informațiilor necesare. Pentru a putea extrage doar acele documente relevante, utilizatorii au nevoie de componente care să îi ajute să poată compara documentele, cum ar fi eficiența şi relevanța lor în raport cu o anumită interogare, sau pentru a putea găsi şabloane în vederea unor indexări şi regăsiri mai facile.

Un număr impresionant de documente se găsesc pe WEB, iar utilizatorul are nevoie de componente de clasificare automată a acestora pentru a le putea gestiona. Gestiunea lor a devenit o problemă foarte importantă în ultima perioadă. Devine esențială existența unor programe inteligente de organizare automată a documentelor în clase pentru a facilita analiza şi recunoaşterea acestora. O procedură generală posibilă pentru rezolvarea acestei probleme constă în alegerea unei mulțimi de documente preclasificate (mult mai mică în comparație cu documentele existente) şi considerarea acestora ca fiind documente de antrenament. Mulțimea de antrenament este apoi analizată pentru a putea extrage o schemă de clasificare. Această schemă este finisată utilizând o mulțime de documente de test. După aceasta, schema astfel obținută poate fi utilizată pentru clasificarea celorlalte documente existente. Analiza clasificării decide care mulțime de perechi de tip atribut-valoare are o putere de discriminare mai mare în determinarea claselor. O metodă eficientă pentru clasificarea documentelor este de a exploata clasificările bazate pe asocieri, unde documentele sunt clasificate pe baza unei mulțimi de asocieri şi a frecvenței de şabloane. Metodele de clasificare pe baza asocierilor respectă următoarele etape: (1) cuvintele cheie şi termenii pot fi extraşi prin tehnici de recunoaştere a informațiilor şi tehnici de analiză a asocierilor simple; (2) ierarhiile de concepte de cuvinte cheie sau termeni pot fi obținute utilizând termenii claselor disponibile, încrederea în cunoştințele expertului sau unele sisteme de clasificare pe bază de cuvinte cheie. Documentele din mulțimea de antrenament pot fi de asemenea clasificate în ierarhii de clase. Metodele de minerit pe baza asocierii termenilor pot fi apoi aplicate pentru a descoperi mulțimi de termeni de asociere care pot fi utilizați pentru a maximiza diferențele dintre două clase de documente. Acestea produc o mulțime de reguli de asociere pentru fiecare clasă de documente. Astfel de reguli de asociere pot fi ordonate pe baza frecvenței lor de apariție şi a puterii de discriminare şi utilizate apoi pentru clasificarea de noi documente.

Clasificarea documentelor text este un proces foarte complex incluzând o mulțime de etape care trebuiesc realizate pentru rezolvarea problemei. Fiecare dintre aceste cerințe are o influență foarte puternică asupra rezultatului final al clasificării. Este imposibilă o abordare completă a acestui vast domeniu, chiar şi într-o teză de doctorat. În ultimii ani s-au depus eforturi importante de cercetare care s-au concentrat pe clasificarea automată a documentelor. Această teză aduce contribuții în dezvoltarea şi implementarea unor clasificatoare bazate pe tehnica vectorilor suport precum şi câteva metode de selecție a trăsăturilor caracteristice atât pentru documentele text cât şi pentru documentele web.

Acest proces poate fi privit ca un flux de date (Figura 1.1) în mai multe etape. În fiecare etapă, se primesc informații, se procesează şi apoi se transferă mai departe. Fiecare etapă din acest flux de date poate avea unul sau mai mulți algoritmi ataşați. La un moment dat, pentru fiecare etapă putem alege unul dintre algoritmii ataşați cu diferiți parametri de intrare.

În ceea ce priveşte structura tezei am ales să prezint stadiul actual şi cercetările mele în domeniul temei, grupate pe domeniile acestor cercetări. Astfel în fiecare capitol se prezintă stadiul actual şi bazele matematice ale domeniului, urmate de contribuțiile mele în domeniul respectiv precum şi o

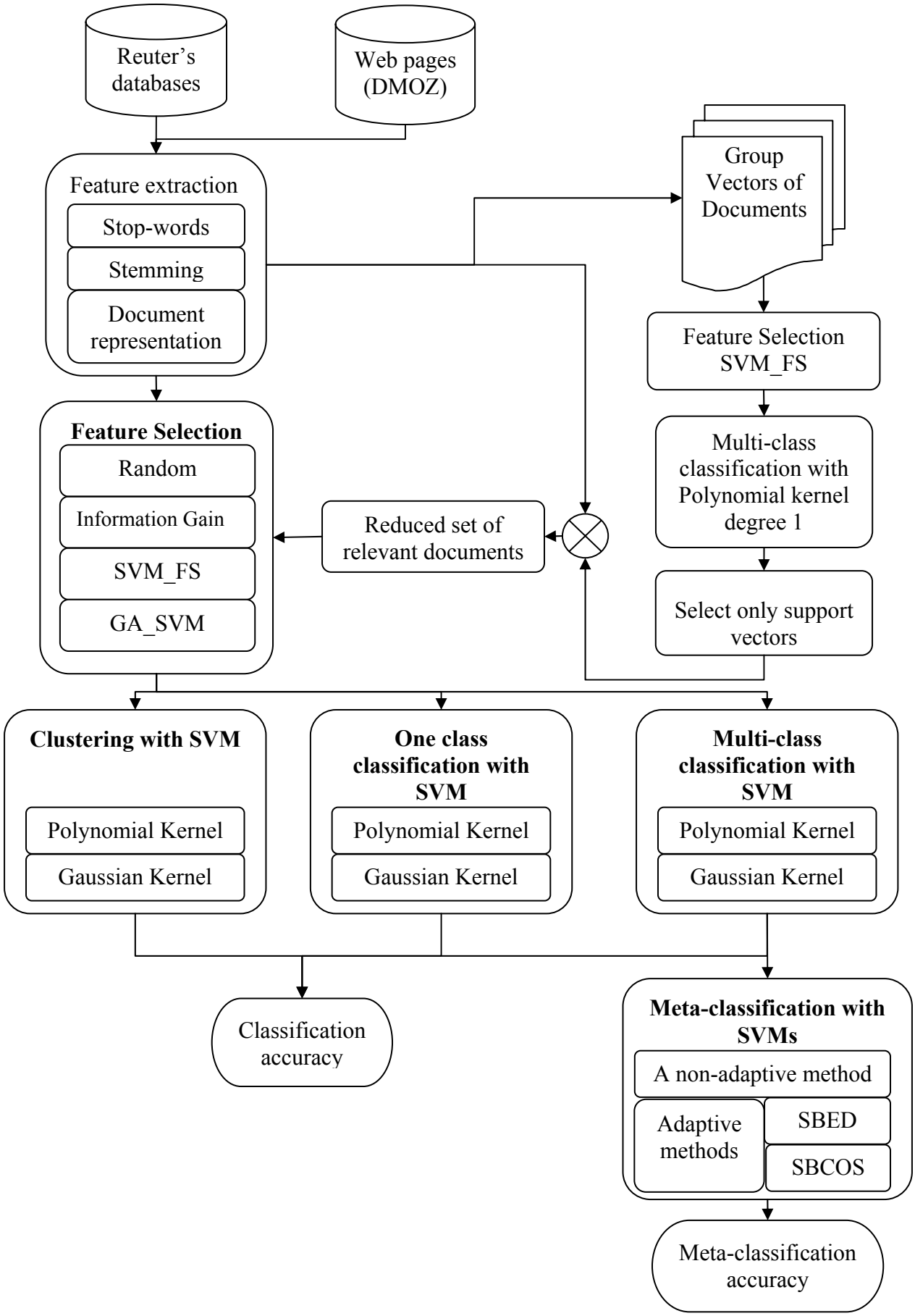

Figura 1.1 – Procesul de clasificare de documnete

serie de rezultate care confirmă îmbunătățirile aduse, o mare parte din aceste rezultate fiind publicate la conferințe internaționale care au avut loc în străinătate şi în țară.

Capitolul doi conține unele tehnici de pre-procesare a documentelor text şi web. În special am prezentat pre-procesarea bazei de date Reuters-2000 care reprezintă o colecție de documente text deosebit de utilizată în acest domeniu. Am continuat cu o scurtă introducere a procesării paginilor web, axându-mă pe diferențele dintre acestea şi documentele text. În această etapă datele de intrare sunt reprezentate prin documente text (fişiere text sau pagini web) şi datele de ieşire sunt reprezentate de vectori de trăsături care caracterizează fiecare document din mulțimea de documente. Trăsăturile reprezentate în aceşti vectori sunt de fapt cuvintele împreună cu frecvențele de apariție ale acestora în documentul respectiv. Această etapă conține un modul de eliminare a cuvintelor ne-relevante din punct de vedere semantic (cunoscute şi sub numele de „stopwords") şi un modul de extragere a rădăcinii cuvintelor pentru limba engleză. Datorită dimensiunii impresionante a vectorilor rezultați această etapă continuă cu o etapă de selectare a trăsăturilor relevante din aceşti vectori. Astfel, în Capitolul 4 am prezentat şi dezvoltat patru metode de selecție a trăsăturilor relevante: selecția Aleatoare, selecția bazată pe Câştigul Informațional, selecția bazată pe Vectori Suport şi selecția utilizând Algoritmi Genetici cu funcția de performanță bazată pe vectori suport.

În capitolul 3 am prezentat în detaliu algoritmul bazat pe tehnica vectorilor suport (SVM) utilizat atât în procesul de clasificare cât şi în procesul de selecție a trăsăturilor caracteristice. M-am axat în general pe tehnica de clasificare a documentelor dar am prezentat aici şi modificările care trebuiesc aduse algoritmului pentru a putea lucra şi cu date ne-etichetate (clustering). Tot aici am prezentat o metodă originală de corelare a parametrilor nucleelor astfel încât să se îmbunătățească acuratețea clasificării şi să se reducă timpii necesari pentru găsirea parametrilor optimi.

În capitolul 5 am prezentat o serie de rezultate obținute cu ajutorul algoritmului implementat precum şi câteva comparații între implementarea mea şi respectiv o implementare utilizată în mod frecvent în literatură, numită „LibSvm". De asemenea, am prezentat într-un spațiu cu două dimensiuni cum acționează noul algoritm în cazul clasificării datelor bidimensionale. Aceasta a fost utilă şi pentru a putea avea o analiză vizuală a modului cum se organizează clasele rezultate.

În Capitolul 6 am prezentat o tehnică de dezvoltare a unor meta-clasificatori pentru a îmbunătăți acuratețea clasificării. Această tehnică se bazează pe folosirea într-un mod adaptiv a mai multor clasificatori de bază. Clasificatorii de bază din acest metaclasificator utilizează tehnica vectorilor suport. Această tehnică se bazează pe alegerea la un moment dat a celui mai bun clasificator pentru clasificarea eşantionului curent.

În capitolul 7 sunt prezentate unele contribuții care îmbunătățesc fluxul clasificării făcându-l mult mai fezabil pentru lucrul cu mulțimi mari de date. Deoarece aplicațiile de clasificare se confruntă în general cu problema numărului mare de date de antrenament şi cu o lipsă a spațiului de memorie, în acest capitol am propus o metodă pentru selectarea celor mai relevante eşantioane din setul de date astfel încât să se reducă timpii de antrenare şi memoria utilizată fără a influența prea mult rezultatele clasificării. Rezultatele prezentate până în acest moment au fost obținute pe o dimensiune mică a setului de date în comparație cu toate datele existente în baza de date. Utilizarea tuturor datelor existente ar face procesul de învățare ineficient din punct de vedere al timpului necesar. Metodologia prezentată face ca aplicația noastră să fie capabilă să lucreze cu o dimensiune impresionantă a datelor de intrare şi cu pierderi cât mai mici din punct de vedere al acurateței clasificării. Această metodologie are două obiective majore antagonice: o dimensiune mare a datelor de intrare şi un timp de învățare cât mai mic pentru o clasificare optimă.

Lucrarea se încheie cu prezentarea contribuțiilor originale şi propuneri de dezvoltare ulterioară, cu un glosar şi cu bibliografia aferentă tezei de doctorat.

Mulţumiri

În încheierea acestei introduceri doresc să exprim sincerele mele mulţumiri către conducătorul meu de doctorat dl. prof. univ. dr. ing. Lucian N. VINŢAN pentru încrederea acordată mie şi pentru coordonarea ştiinţifică pe întreaga perioadă de pregătire a acestui doctorat, pentru discuţiile profesionale extrem de stimulative pe care le-am avut precum şi pentru întreg sprijinul acordat în mod prompt şi variat. De asemenea, doresc să mulţumesc colegilor din cadrul Catedrei de Calculatoare şi Automatizări din cadrul Universităţii „Lucian Blaga" din Sibiu pentru sprijinul acordat şi pentru climatul plăcut existent care a contribuit şi el la motivarea acestei teze.

În acelaşi timp aş dori să mulţumesc firmei SIEMENS AG, CT IC MUNCHEN, Germania, în special d-lui vicepreşedinte Dr. h. c. mat. Hartmut RAFFLER, pentru sugestiile profesionale foarte folositoare şi pentru suportul financiar acordat pe parcursul acestui program de doctorat. Aş dori să mulţumesc tutorelui meu de la firma SIEMENS, Dr. Volker TRESP, cercetător principal în calcul neuronal, pentru sprijinul ştiinţific acordat şi pentru îndrumarea în acest imens şi dificil domeniu al cercetării. De asemenea doresc să mulţumesc d-lui Dr. Kai Yu de la aceeaşi companie pentru informaţiile furnizate în dezvoltarea ideilor mele.

Gândurile mele de recunoştinţă se îndreaptă şi asupra referenţilor acestei teze de doctorat, pentru bunăvoinţa şi efortul depus în recenzarea acestei lucrări.

La sfârşit aş dori să mulţumesc familiei şi prietenilor pentru sprijinul acordat şi pentru faptul că am păstrat această calitate, în ciuda faptului că timpul alocat lor a fost considerabil redus în toată această perioadă.

2 Metode pentru extragerea cunoştinţelor

Deoarece tot mai multe informaţii (date) sunt disponibile în format electronic, regăsirea acestora devine o provocare, fără a şti ce se găseşte în documente şi fără o indexare a conţinutului acestora. Catalogarea documentelor este o soluţie pentru această problemă şi tendinţele actuale încearcă să o combine cu informaţiile despre profilul utilizatorului sau cu analiza semantică a textului. Pentru a putea efectua catalogarea trebuie să putem clasifica automat datele în categorii predefinite. Multe metode de clasificare şi tehnici de învăţare automată s-au utilizat în ultimii ani pentru clasificarea documentelor. Dar din păcate majoritatea informaţiilor disponibile sunt neetichetate (nu au specificată categoria) şi procesul de catalogare devine mult mai dificil.

2.1 Data Mining în baze de date

Tehnica Data Mining se referă la extragerea sau descoperirea „cunoştinţelor" din depozite sau masive mari de date. Data Mining este termenul prescurtat pentru "extragerea cunoştinţelor din date". Mulţi cercetători tratează data mining ca un sinonim pentru un alt termen uzual: "descoperirea de cunoştinţe din baze de date". Alţii văd data mining ca un pas esenţial în procesul de descoperire de cunoştinţe din baze de date. Astfel, la ora actuală data mining reprezintă procesul de descoperire a cunoştinţelor interesante din cantităţi mari de date memorate mai de grabă în depozite de date sau magazii de informaţii decât în baze de date simple. Procesul de descoperire a cunoştinţelor din baze de date are mai mulţi paşi: preprocesarea datelor, data mining, evaluarea modelelor extrase şi reprezentarea cunoştinţelor.

2.1.1 Preprocesarea datelor

Acesta este un pas important în procesul de descoperire a cunoştinţelor. Bazele sau depozitele de date din zilele noastre sunt foarte susceptibile la zgomot, fiind incomplete şi inconsistente datorită îndeosebi dimensiuni mari pe care le au. Scopul preprocesării datelor este de a pregăti datele pentru analiză. Există un număr de paşi de preprocesare a datelor cum ar fi: curăţirea datelor, integrarea datelor, selectarea datelor, transformarea şi reducerea acestora.

2.1.2 Data mining

Data mining este un pas esenţial în procesul de descoperire de cunoştinţe unde metodele de inteligenţă artificială sunt aplicate cu scopul de a extrage reguli din aceste date. În pasul de prelucrare a datelor utilizatorul comunică cu sistemele de extragere a informaţilor utilizând un set de *primitive,* cu scopul de a facilita descoperirea de cunoştinţe importante şi de înţelegere a acestora.

Procesul Data mining poate fi clasificat în două categorii: *mineritul descriptiv al datelor* şi *mineritul predictiv al datelor*. În pasul de analiză a datelor utilizatorii au doar o idee vagă despre atributele care pot fi interesante pentru procesul de extragere de cunoştinţe.

2.1.3 Extragerea regulilor de asociere

Extragerea regulilor de asociere constă în găsirea de mulţimi de atribute frecvente (mulţimi de atribute cum ar fi A sau B care satisfac pragul de încredere minim şi procentajul de grupuri relevante). Aceste reguli de asociere trebuie să satisfacă pragul de încredere minim. Pentru extragerea regulilor de asociere există deja câţiva algoritmi clasici consacraţi cum ar fi „Apriori", „Construirea şabloanelor frecvente", „Reguli de asociere pe mai multe nivele" şi „Extragerea regulilor pe baza constrângerilor".

2.1.4 Clasificarea şi predicţia

Clasificarea şi predicţia sunt două forme de analiză supervizată a datelor care pot fi utilizate pentru extragerea modelelor importante care descriu clasele de date sau prezic tendinţele viitoare ale datelor, fiind de altfel un pas important în procesul de analiză a datelor. În timp ce clasificarea prezice categoriile, predicţia modelează funcţiile cu valori continue.

2.1.5 Clustering

Clustering-ul este un proces de învăţare ne-supervizată care grupează datele în clase (clustere) astfel încât, obiectele dintr-o clasă au o mare similaritate intre ele dar sunt foarte disimilare în comparaţie cu obiectele din celelalte clase (clustere). Disimilaritatea este evaluată pe baza valorii atributelor (trăsăturilor) care descriu obiectul. Analiza clusterului poate fi utilizată fie ca o componentă de extragere a informaţilor de sine stătătoare pentru a evalua câştigul în distribuţia datelor, fie ca un pas de preprocesare pentru alţi algoritmi de analiză a datelor.

2.2 Text mining

Până acum am prezentat analiza datelor ca pe o metodă de căutare a şabloanelor în baze de date care sunt considerate date structurate. În realitate o cantitate impresionantă de date se găseşte în fişiere text constituind colecţii mari de documente, obţinute din diferite surse, cum ar fi articolele de ştiri, articolele de cercetare, cărţile, bibliotecile digitale, mesajele e-mail şi paginile web.

Din această cantitate enormă de date text, doar o mică porţiune va fi relevantă la un moment dat pentru utilizator. Astfel utilizatorul are nevoie de componente pentru a putea compara diferite documente şi a le putea ordona în funcţie de importanţa sau relevanţa acestora. Fără a şti ce se găseşte în documente este dificil să se poată formula interogări eficiente pentru analiza şi extragerea acestora din aceste mulţimi de documente. Astfel, analiza documentelor text a devenit un subiect popular şi esenţial în procesul de data mining. În acest pas documentele care se găsesc de obicei într-un format uşor de înţeles pentru utilizator sunt transformate în formate care sunt mai uşor de înţeles pentru calculator iar după aceea sunt folosite anumite tehnici pentru analiza şi extragerea informaţiilor relevante.

2.2.1 Analiza datelor text şi găsirea informaţiilor relevante

Regăsirea informaţiilor este un domeniu dezvoltat în paralel cu sistemele de baze de date. Regăsirea informaţiei se concentrează pe organizarea şi găsirea informaţiilor dintr-un număr mare de documente text. O problemă tipică de regăsire a informaţiei este de a localiza documentele relevante pe baza intrărilor date de utilizator cum ar fi cuvinte cheie sau exemple de documente. De obicei sistemele de regăsire a informaţiei includ sisteme on-line de cataloage de biblioteci şi sisteme on-line de management al documentelor. De vreme ce atât sistemele de baze de date cât şi sistemele de recunoaştere a informaţiei lucrează fiecare cu tipuri diferite de date, apar câteva probleme de la sistemele de baze de date care de obicei nu sunt prezente în sistemele de regăsire a informaţilor. De asemenea sunt anumite probleme de recunoaştere a informaţilor comune care de obicei nu apar în sistemele tradiţionale de baze de date.

2.2.1.1 Metrici de bază pentru recunoaşterea informaţilor

Există câteva metode pentru măsurarea gradului de relevanţă în sistemele de regăsire a informaţilor. Notăm cu [Relevant] mulţimea de documente relevante pentru o anumită interogare şi [Regăsite] mulţimea de documente găsite după o interogare. Mulţimea de documente care sunt atât relevante pentru interogare cât şi regăsite este notată prin $[\text{Relevantt}] \cap [\text{Regasite}]$. Există două metrici de bază pentru a evalua calitatea unui sistem de regăsire a informaţiei: „Precizia relevanţei” care reprezintă procentajul de documente care sunt relevante din toate documentele găsite şi „Precizia documentelor regăsite” care reprezintă procentajul de documente care sunt relevante şi găsite, din toate documentele relevante existente.

2.2.1.2 Regăsirea bazată pe cuvinte cheie şi similarităţi

Majoritatea sistemelor de regăsire a informaţiilor suportă regăsirea bazată pe cuvinte cheie şi regăsirea bazată pe similarităţi. În sistemele de recunoaştere bazate pe cuvinte cheie un document este reprezentat printr-un şir de valori care poate fi identificat de un set de cuvinte cheie. Utilizatorul furnizează un cuvânt cheie sau o expresie formată dintr-o mulţime de cuvinte cheie cum ar fi „maşină şi atelier de reparaţii" iar documentele relevante sunt acelea care conţin aceste cuvinte cheie.

Sistemele de regăsire a informaţilor bazate pe similarităţi găsesc documente similare pe baza unei mulţimi de cuvinte cheie comune. Ieşirea pentru acest sistem se bazează pe măsurarea gradului de relevanţă prin utilizarea metodelor de calcul a apropierii cuvintelor cheie şi a frecvenţei relative a cuvintelor cheie. În sistemele moderne de regăsire a informaţiei, cuvintele cheie pentru reprezentarea documentelor sunt extrase automat din document. Acest sistem adesea asociază setului de documente o listă de cuvinte de legătură (stopwords). Lista de cuvinte de legătură este o mulţime de cuvinte care sunt considerate nerelevante pentru ceea ce se caută şi care pot varia când setul de documente se schimbă. O problemă abordată în acest context este problema extragerii rădăcinii cuvintelor (stemming). Un grup de cuvinte diferite poate să pornească de la aceeaşi rădăcină. Un sistem de regăsire a informaţiilor trebuie să identifice aceste grupuri de cuvinte când cuvintele din grup au variaţii mici din punct de vedere sintactic şi să colecteze doar cuvântul comun sau rădăcina pentru un grup. Aceasta reprezintă abordarea utilizată în această teză pentru reprezentarea documentelor.

2.2.2 Analiza clasificării documentelor

Există un număr crescând de documente disponibile şi clasificarea automată a acestora a devenit o problemă importantă. Este esenţial ca să putem organiza automat aceste documente în clase astfel încât să putem facilita regăsirea şi analiza acestora. O procedură generală posibilă pentru această clasificare este de a alege un set de documente preclasificate şi a-l considera ca şi mulţime de antrenament. Mulţimea de antrenament este apoi analizată cu scopul de a găsi o schemă de clasificare. Schema de clasificare trebuie rafinată ulterior folosind un proces de testare a acesteia. După aceasta, schema astfel obţinută poate fi folosită pentru clasificarea celorlalte documente disponibile. Analiza clasificării decide care perechi de mulţimi de atribute-valoare au o putere mai mare de discriminare în determinarea claselor. O metodă pentru clasificarea documentelor este de a exploata asocierile pe baza clasificării, care clasifică documentele pe baza unei mulţimi de asocieri şi şabloane de text apărute frecvent.

2.3 WEB mining

Web-ul este un serviciu imens care oferă o cantitate impresionantă de informaţii. Acesta conţine de asemenea o colecţie bogată şi dinamică de informaţii de legătură între paginile web şi informaţii despre utilizare acestora, furnizând surse bogate pentru procesul de descoperire de cunoştinţe. Extragerea cunoştinţelor din paginile web este diferită de extragerea cunoştinţelor din documentele text. Web-ul este enorm şi creşte rapid, informaţiile memorate pe web sunt actualizate continuu şi paginile web conţin de departe cele mai multe stiluri şi variaţii de conţinut decât oricare mulţime de cărţi sau oricare documente bazate pe text tradiţional. O altă problemă pentru analiza web-ului este că web-ul serveşte o diversitate de comunităţii de utilizatori iar utilizatorii pot avea contexte, interese şi propuneri de utilizare foarte diferite. Când utilizatorii vor să găsească ceva pe web doar a mică parte din informaţiile disponibile vor fi relevante pentru ceea ce doreşte utilizatorul curent. Aceste provocări au încurajat cercetarea în domeniul descoperirii şi utilizări eficiente a resurselor de pe web. Există multe motoare de căutare bazate pe indexare care caută pe web, indexează paginile acestuia construind şi memorând cantităţi mari de indexări bazate pe cuvintele cheie conţinute în paginile indexate. Aceste indexări ne ajută să localizăm ulterior mulţimea de pagini

care conţin anumite cuvinte cheie. Astfel un utilizator fără experienţă poate fi capabil să localizeze repede documentele care îl interesează furnizând o mulţime de cuvinte cheie sau fraze cheie.

2.3.1 Clasificarea automată a documentelor WEB

În clasificarea automată a documentelor Web, fiecărui document îi este atribuită o etichetă de clasă (o categorie) dintr-un set de categorii predefinite, pe baza unei mulţimi de exemple de documente preclasificate. De exemplu, taxonomia Yahoo de pe net şi documentele asociate ei poate fi utilizată ca mulţime de antrenare sau testare cu scopul de a construi o schemă de clasificare a celorlalte documente de pe web. Această schemă poate fi utilizată ulterior pentru clasificare de noi documente web prin asignarea de categorii din aceeaşi taxonomie. Această metodă poate fi folositoare pentru clasificare în învăţarea supervizată.

2.3.2 Categorii web mining

Data mining constă în trei paşi: *preprocesarea datelor, descoperirea de reguli (şabloane) şi analiza regulilor*. Similar, web mining e compus din trei părţi, fiecare conţinând cei trei paşi anterior prezentaţi:

- *Analiza conţinutului paginilor web* – reprezintă analiza textului extras din paginile web (datele din paginile web) şi meta-datele furnizate de tag-urile din fişierele HTML.
- *Analiza structurii web* – reprezintă analiza informaţilor care descriu organizarea site-ului şi a conţinutului acestuia.
- *Analiza utilizării web-ului* – reprezintă analiza informaţilor care descriu şabloanele de utilizare a legăturilor dintre paginile Web.

Analiza conţinutului paginilor Web este o formă de data mining. Resursa primară de pe web o reprezintă pagina individuală. Analiza conţinutului web-ului poate avea avantaje datorită naturii structurate sau semistructurate a textului din paginile web. Pentru aceasta există câteva tehnici pentru regăsirea informaţiilor din documentele text, cum ar fi metodele pentru indexarea textului, care au fost dezvoltate pentru a lucra cu documente text nestructurate (semistructurate).

Analiza structurii web-ului de obicei operează asupra structurii legăturilor dintre paginile web. Analiza structurii web-ului explorează informaţii adiţionale care sunt conţinute în structura hipertextului. Un domeniu important de aplicabilitate este identificarea relevanţei relative pentru diferite pagini care apar ca fiind echivalente când le analizăm din punct de vedere al conţinutului.

Analiza utilizării web-ului caută după informaţii care derivă din interacţiunea utilizatorului cu web-ul. Informaţiile despre utilizarea web-ului includ datele memorate în: fişierele log ale serverelor de acces, ale proxy serverelor, ale browser-elor, datelor de înregistrare ale utilizatorului, sesiunii sau tranzacţiei utilizator, cookies, interogărilor utilizator, acţiunilor mouse-ului, profilului utilizator şi oricare altă dată ca rezultat al interacţiunii utilizatorului cu web-ul. Analiza utilizării web-ului se bazează pe tehnicile care pot prezice comportamentul utilizatorului în timpul interacţiunii acestuia cu web-ul.

2.3.3 Sisteme pentru descoperirea de resurse

Chiar dacă proiectanţii interfeţelor şi a conţinutului web-ului includ comportamentul utilizator şi câteva deprinderi în unele motoare de căutare, utilizatorii pot avea dificultăţii în efectuarea de căutări pe web deoarece aceştia nu sunt capabili să formuleze interogările corecte care să reducă numărul de rezultate obţinute şi să crească calitatea acestora. De obicei interfaţa utilizator este dificil de utilizat, funcţiile de organizare ierarhică sunt aplicate static şi informaţiile de pe web sunt foarte rar structurate şi organizate pentru o recunoaştere rapidă.

Pe lângă calitatea rezultatelor motoarelor de căutare, un alt aspect important îl reprezintă utilitatea componentelor de căutare. Acest aspect este adeseori neglijat. Este important să-l facem foarte accesibil şi utilizabil de către oricine, indiferent de condiţiile psihice şi de mediul din care provine

utilizatorul. Accesibilitatea garantează utilizarea, înţelegerea şi navigarea pentru toţi utilizatorii. Uşurinţa în utilizare creează o mai mare eficienţă şi satisfacţie în procesul de utilizare a web-ului

Pentru a rezolva problemele prezentate mai sus acest domeniu de cercetare a crescut continuu în direcţii cum ar fi algoritmii, strategiile şi arhitecturile. Sute de idei au fost testate, unele fiind implementate iar altele existând doar ca şi prototipuri. Unele dintre aceste idei sunt: organizarea documentelor în categorii predefinite, îmbunătăţirea modului de prezentare a rezultatelor, monitorizarea unei pagini specifice, ajutarea utilizatorului in formularea interogării corecte şi dezvoltarea de programe pentru filtrarea rezultatelor căutării.

Organizarea documentelor în categorii predefinite, de obicei numite *directoare web,* se realizează prin crearea unei structuri statice care cu timpul devine foarte greoaie şi dificil de actualizat. Această structură este construită prin eforturi umane şi are ca rezultat o taxonomie gigant de directoare de categorii. Fiecare director conţine o colecţie de legături către documente relevante pentru acea categorie, care sunt adesea cele mai populare sau către site-uri „cu autoritate" relativ la categoria specificată.

O problemă importantă pentru majoritatea motoarelor de căutare este reprezentarea rezultatului căutării. Când motoarele de căutare întorc rezultatul unei căutări de obicei ele întorc mai multe pagini care conţin legături către acele rezultate. Unele dintre aceste legături sunt relevante pentru utilizator, altele nu. Este foarte dificil pentru utilizator să distingă rapid între paginile relevante şi cele nerelevante. Un alt dezavantaj este că motoarele de căutare întorc de obicei ca răspuns la cererea utilizatorului o listă ordonată de pagini web după gradul de accesare al acestora. După recepţionarea rezultatului de la motorul de căutare un alt program (agent) încearcă să analizeze toate aceste rezultate şi încearcă să le clasifice în categorii generate dinamic. Organizarea rezultatelor motoarelor de căutare în ierarhii de categorii sau subcategorii facilitează parcurgerea acestora şi localizarea mai uşoară a rezultatelor interesante.

În ultimi ani o idee comună utilizată pentru creşterea calităţii rezultatelor căutării constă în crearea unui profil utilizator pe baza a ceea ce îl interesează pe acesta. Apoi, acest profil este utilizat pentru a îmbunătăţii rezultatul căutării ori pentru a dezvolta noi interogări pentru a obţine rezultate mai bune.

Utilizarea numai a cuvintelor cheie este de obicei ambiguă sau foarte generală pentru motoarele de căutare. Mai mult decât atât, acestea pot apărea în foarte multe documente, făcând astfel ca procesul de căutarea să întoarcă sute de rezultate, majoritate fiind nerelevante în raport cu ceea ce doreşte utilizatorul. Specificând cuvinte cheie adiţionale putem rafina căutarea şi putem obţine o îmbunătăţire considerabilă a rezultatelor. Cuvintele de rafinare au ca sens eliminarea ambiguităţilor şi specificarea clară a sensului cuvântului original de căutare. Furnizând cuvinte suplimentare de rafinament putem ajuta motoarele de căutare să elimine documentele în care cuvântul apare în oricare dintre sensurile nedorite.

2.3.4 Web-ul semantic şi ontologiile

Definiţia web-ului semantic după Tim Berners-Lee, inventatorul „World Wide Web", este: "*The extension of current web in which information is given well-defined meaning, better enabling computers and humans to work in cooperation.*"[Ber99]. Aceasta ne arată tendinţa actuală în acest vast domeniu şi provocările de a face calculatoarele să ne poată înţelege şi să ne poată oferi informaţii multe, relevante şi mult mai rapid decât acum.

Web-ul semantic şi tehnologiile lui ne oferă o nouă abordare pentru informaţiile şi procesul de management al conţinutului acestuia. Principalul obiectiv de dezvoltare este utilizarea unor metainformaţii semantice. Metainformaţia descrie documentul, de exemplu paginile web, sau o parte din document, de exemplu o persoană sau o companie. În fiecare caz idea principală este că metadata este semantică, ne oferă informaţii despre conţinutul documentului (de exemplu subiectul sau relaţia acestuia cu alte documente) sau despre entităţile din document. Această nouă

metainformaţie este în contrast cu metadatele existente din web-ul curent, (codificate în structura html) şi care descriu sumar formatul în care informaţiile vor fi prezentate utilizatorului.

Ontologiile reprezintă „inima" tuturor aplicaţiile de web semantic. Definiţia ontologiei este după mai mulţi cercetători în domeniu: "*ontology is an explicit and formal specification of a conceptualization of a domain of interest*". Ontologiile sunt utilizate pentru organizarea cunoştinţelor într-o formă structurată în foarte multe domenii – de la filozofie la managementul cunoştinţelor şi la web-ul semantic.

2.4 Seturile de date utilizate în experimente

O mare parte din experimentele prezentate în această teză sunt efectuate folosind colecţia de date Reuters-2000, care conţine 984Mb de articole de ştiri într-un format comprimat. Această colecţie este de obicei utilizată în cercetare pentru clasificarea automată a documentelor. Colecţia include un total de 806791 documente, care reprezintă articolele de ştiri publicate de agenţia de presă Reuters în perioada 20 August 1996 – 19 August 1997. Analizate, articolele conţin 9822391 paragrafe, 11522847 propoziţii şi 310033 rădăcini de cuvinte distincte rămase după eliminarea cuvintelor de legătură (stopword). Documentele sunt preclasificate de Reuters din punct de vedere a trei categorii distincte. După ramura industrială la care se referă articolul există 870 categorii. După regiunea geografică la care se referă articolul există 366 categorii şi după anumite categorii propuse de Reuters, în funcţie de conţinut, există 126 categorii distincte. Dintre acestea, 23 nu conţin nici un articol. În experimente am luat în considerare categoriile propuse de Reuters ca referinţă pentru algoritmul meu.

În partea de extragere a cuvintelor din documente am utilizat o listă generală de cuvinte de legătură pentru limba engleză pusă la dispoziţie de Universitatea din Texas. Această listă cuprinde un număr de 509 cuvinte generale, nu neapărat legate de contextul Reuters.

Pentru fiecare cuvânt rămas după procesul de eliminare a cuvintelor de legătură am extras rădăcina cuvântului şi am numărat apariţiile acestuia în document. Astfel am creat un vector de cuvinte pentru fiecare document din Reuters (aceste cuvinte numindu-se în continuare trăsături). Acest vector va reprezenta semnătura fiecărui document. În continuare, folosind aceşti vectori am creat un vector mai mare care cuprinde toate trăsăturile unice care apar în toate documentele. După acest pas toţi vectorii devin de aceeaşi dimensiune şi reprezintă semnătura documentul în întreg setul de documente. Pentru memorarea tuturor acestor vectori am ales varianta de a memora doar valorile pentru care numărul de apariţii al cuvintelor este mai mare decât zero. Astfel s-au creat perechi de atribut – valoare care reprezintă trăsătura şi numărul de apariţii ale acesteia în documentul curent. La sfârşitul vectorului se păstrează şi categoriile (clasele) propuse de Reuters pentru documentul curent.

2.4.1 Selectarea documentelor pentru antrenare - testare

Datorită dimensiunii mari a bazei de date voi prezenta rezultatele obţinute utilizând o submulţime a acesteia. Din toate 806791 documente, am selectat acele documente care sunt grupate de Reuters în categoria „*System Software*" după codul industrial. După această selecţie am obţinut un număr de 7083 documente care sunt reprezentate utilizând un număr de 19038 trăsături. În setul rezultat se găsesc 68 clase diferite. Dintre aceste clase am eliminat acelea care sunt conţinute în mai puţin de 1% din toate documentele (slab reprezentate). De asemenea am eliminat clasele care sunt conţinute în mai mult de 99% dintre documente (excesiv reprezentate). După aceste eliminări au rămas doar 24 clase distincte şi un număr de 7053 documente. Aceste documente sunt împărţite aleator într-o mulţime de antrenare de 4702 documente şi o mulţime de testare de 2351 documente (eşantioane).

Majoritatea cercetătorilor iau în considerare doar titlul şi rezumatul articolului sau doar primele 200 cuvinte din fiecare articol. În experimentele mele am luat în considerare întreg articolul şi titlul acestuia pentru crearea vectorului documentului.

2.4.2 Alegerea unui set de date mai mare

În anumite experimente din această teză am avut nevoie de un set de date de dimensiune mai mare. Astfel pornind de la baza de date Reuters am selectat acele documente care sunt grupate de Reuters în categoria „*Computer Systems and Software*" după codul industrial. În această selecţie am obţinut un număr de 16177 documente. Din aceste documente am extras un număr de 28624 trăsături după eliminarea cuvintelor de legătură şi extragerea rădăcinii cuvintelor. Aceste documente sunt preclasificate de Reuters în 69 clase diferite. Dintre aceste clase le-am eliminat pe acelea care sunt slab sau excesiv reprezentate în întreaga mulţime rămânând un număr de 25 clase şi un număr de 16139 eşantioane. Aceste eşantioane le-am împărţit aleator în setul de antrenare de 10757 eşantioane şi setul de test de 5282 eşantioane.

2.4.3 Crearea setului de documente Web

O parte din experimentele prezentate în această teză sunt efectuate folosind o bază de date care conţine documente web preluate automat de pe site-ul DMOZ. În prelucrarea acestor documente sunt interesat doar de partea de analiză a conţinutului paginilor web nefiind interesat de analiza structurii şi a utilizării web-ului. Selectând o serie de categorii de pe site-ul DMOZ am creat o mulţime de lucru cu documentele din acele categorii pe care am folosit-o pentru testarea algoritmului de clasificare. Am ales site-ul DMOZ deoarece documentele de pe acest site sunt preclasificate. Un document este considerat ca fiind clasificat în categoria în care este găsit pe site şi de asemene şi în trei categorii de pe nivelele anterioare, dacă acestea există. De asemenea poate exista situaţia ca acelaşi document să se găsească în două sau mai multe categorii diferite. În acest caz am lăsat documentul în ambele categorii. Am selectat un număr de 745 documente din care am extras un număr de 26998 rădăcini de cuvinte. Documentele selectate au fost grupate aleator în mulţimea de antrenare de 445 documente şi mulţimea de testare de 377 documente.

2.4.4 Tipuri de reprezentare a datelor

Există mai multe posibilităţi de definire a ponderilor (trăsăturilor) din vectori. În funcţie de reprezentarea aleasă anumiţi algoritmi de clasificare funcţionează mai bine sau mai prost. Eu am reprezentat datele de intrare în trei formate diferite şi am încercat să analizez influenţa fiecărui format în parte.

- Reprezentarea Binară – în vectorul de intrare sunt memorate doar valorile „0" dacă cuvântul respectiv nu apare în document sau „1" daca cuvântul respectiv apare în document fără a mă interesa numărul de apariţii ale acelui cuvânt în document.
- Reprezentare Nominală – valorile vectorului de intrare sunt normalizate astfel încât toate valorile să fie între 0 şi 1 utilizând următoarea formulă:

$$TF(d,t) = \frac{n(d,t)}{\max_{\tau} n(d,\tau)} \tag{2.1}$$

unde *n(d, t)* reprezintă numărul de apariţii al termenului *t* în documentul *d* şi numărătorul reprezintă valoarea termenului care apare de cele mai multe ori în documentul *d*, iar cu $TF(d,t)$ este notată frecvenţa termenului *t*.

- Reprezentarea Cornell SMART – în vectorul de intrare valoarea ponderilor este calculată utilizând formula:

$$TF(d,t) = \begin{cases} 0 & if\ n(d,t) = 0 \\ 1 + \log(1 + \log(n(d,t)) & otherwise \end{cases} \tag{2.2}$$

unde $n(d,t)$ reprezintă numărul de apariţii ale termenului *t* în documentul *d*. În acest caz ponderea poate lua valoarea 0 dacă cuvântul nu apare în document sau o valoare între 1 şi 2 dacă apare în funcţie de numărul de apariţii. Valoarea este mărginită superior pentru un număr de apariţii mai mic ca 10^9 deoarece logaritmul este în baza 10.

3 Support Vector Machine. Considerații Matematice

În acest capitol prezint câteva aspecte teoretice referitoare la tehnica de învățare bazată pe vectori support şi nuclee numită Support Vector Machine (SVM). Am ales acest algoritm datorită posibilității lui de a lucra cu vectori de dimensiune foarte mare şi datorită fiabilității şi performanțelor obținute de acesta în cazul învățării datelor neliniar separabile. De asemenea voi prezenta câteva aspecte referitoare la tehnicile de implementare ale acestui algoritm pentru cazul clasificării şi al grupării automate de documente text. Clasificarea este o tehnică de învățare supervizată care utilizează o mulțime de exemple etichetate pentru antrenare şi încearcă să găsească regulile după care sunt împărțite cel mai bine mulțimile de antrenament. Clustering-ul este o metodă nesupervizată de grupare a exemplelor în clase (clustere) astfel ca obiectele din aceeaşi clasă să fie similare iar cele din clase diferite să fie disimilare.

3.1 Tehnica SVM pentru clasificarea binară

Support Vector Machine (SVM) este o tehnică de clasificare bazată pe teoria învățării statistice care a fost aplicată cu mult succes în multe probleme neliniare de clasificare şi pentru mulțimi foarte mari de date.

Idea algoritmului SVM este de a găsi un hiperplan care împarte optim setul de date de antrenament. Hiperplanul optim se poate distinge prin marginea de separare maximă dintre toate punctele de antrenare şi hiperplan. Pentru o problemă într-un spațiu bidimensional algoritmul caută după o dreaptă care separă „cel mai bine" punctele din clasa pozitivă de punctele din clasa negativă. Hiperplanul este caracterizat printr-o funcție de decizie de forma:

$$f(x) = \operatorname{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) \tag{3.1}$$

unde **w** este vectorul pondere, perpendicular pe hiperplan, "*b*" este un scalar care reprezintă marginea hiperplanului, "*x*" este eşantionul curent testat şi $\langle \cdot, \cdot \rangle$ reprezintă produsul scalar. *Sgn* este funcția semn care întoarce 1 dacă valoarea obținută este mai mare sau egală cu 0 şi -1 altfel. Dacă **w** este de lungime 1, atunci <**w,** *x*> este de lungimea lui *x* de-a lungul direcției lui **w**. În general **w** va fi scalat prin ||**w**||. În partea de antrenare algoritmul trebuie să găsească vectorul normal "**w**" care conduce la cea mai mare margine "*b*" a hiperplanului.

Problema pare foarte uşor de rezolvat dar trebuie să reamintim că linia optimă de clasificare trebuie să clasifice corect toate elementele generate cu aceeaşi distribuție. Există o multitudine de hiperplane care îndeplinesc cerințele clasificării, dar algoritmul încearcă să determine hiperplanul optim. Acest algoritm de învățare are loc într-un spațiu în care există produs scalar iar pentru datele liniar separabile construieşte funcția *f* din date empirice. Se bazează pe două idei principale: dintre toate hiperplanele de separare există un hiperplan optim unic care se distinge prin marginea maximă de separare dintre orice punct de antrenare şi hiperplan. A doua idee este: capacitatea hiperplanului de a separa clasele descreşte o dată cu creşterea marginii.

Pentru datele de antrenament care nu sunt liniar separabile printr-un hiperplan de separare în spațiul de intrare idea algoritmului SVM este de a proiecta datele de intrare într-un spațiu de dimensiune mai mare prin intermediul unei funcții Φ, şi a încerca să găsească acolo un hiperplan de separare cu marginea maximă. Aceasta conduce la o graniță de separare neliniară în spațiul inițial de intrare. Datorită faptului că toți vectorii apar în produse scalare şi prin utilizarea funcției nucleu $\langle \phi(\mathbf{w}), \phi(x) \rangle$,este posibil să calculăm hiperplanul de separare fără a proiecta explicit datele în noul spațiu de trăsături.

Pentru a găsi hiperplanul de separare optim care se distinge prin marginea maximă, trebuie să rezolvăm următoarea funcție obiectiv:

$$\begin{aligned} &\underset{\mathbf{w}\in H, b\in\Re}{\text{minimize}}\ \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \\ &\textit{subject to } y_i\left(\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right) \geq 1 \textit{ for all } i = 1,...,m \end{aligned} \tag{3.2}$$

Constrângerile asigură faptul că $f(x_i)$ va fi +1 pentru y_i=+1 şi -1 pentru y_i=-1. Această problemă este atractivă din punct de vedere al calculelor deoarece poate fi abordată prin rezolvarea unei probleme de programare pătratică pentru care există algoritmi eficienți. Funcția τ se numeşte funcția obiectiv cu inegalități de constrângeri. Acestea formează aşa numita problemă de optimizare primară. Pentru rezolvarea acestei probleme este mult mai convenabil să lucrăm cu problema duală prin introducerea multiplicatorilor Lagrange $\alpha_i \geq 0$ şi a Lagrangianului care conduce la problema duală de optimizare:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i (\langle \mathbf{x_i}, w\rangle + b) - 1) \tag{3.3}$$

cu multiplicatorii Lagrange $\alpha_i \geq 0$. Observăm că restricțiile sunt incluse în partea a doua a Lagrangianului şi nu mai trebuie să fie aplicate separat. Lagrangianul *L* trebuie să fie maximizat în raport cu variabilele duale α_i, şi minimizat în raport cu variabilele primare **w** şi *b*. Aceasta conduce la:

$$\begin{aligned} &\mathbf{w} = \sum_{i=0}^{m} \alpha_i y_i x_i \\ &\sum_{i=0}^{m} \alpha_i y_i = 0 \end{aligned} \tag{3.4}$$

Vectorul soluție este o extensie în termeni de exemple de antrenament. Observăm de altfel că soluția **w** este unică (datorită convexității stricte pentru problema primară de optimizare), coeficienți α_i, nu trebuie să fie unici. În acord cu teorema *Karush-Kuhn-Tucker (KKT)* doar multiplicatorii Lagrange α_i care sunt diferiți de zero sunt puncte de sprijin, corespunzătoare constrângerilor care se întâlnesc. Formal, pentru toți *i*=1,…,m acesta poate fi scris:

$$\alpha_i \left[y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) - 1 \right] = 0 \textit{ for all } i = 1,..., m \tag{3.5}$$

Exemplele x_i pentru care $\alpha_i > 0$ se numesc vectori suport. Această terminologie se referă la termeni corespondenți din teoria convexității. În acord cu condițiile KKT, aceştia vor sta exact pe margine. Toate celelalte exemple de antrenament rămase sunt nerelevante. Prin eliminarea variabilelor primare **w** şi *b* din Lagrangian obținem aşa numita problemă de optimizare duală, care este problema care se rezolvă de obicei în practică.

$$\underset{\alpha\in\Re^m}{\text{maximize}}\, W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{3.6}$$

care se mai numeşte şi funcția țintă, cu următoarele restricții:

$$\text{subject to } \alpha_i \geq 0 \textit{ for all } i = 1,..., m \textit{ and } \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{3.7}$$

Astfel hiperplanul poate fi scris, în problema de optimizare duală ca:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \langle x_i, x \rangle + b \right) \tag{3.8}$$

unde *b* este calculat utilizând condiţiile KKT.

Structura problemei de optimizare este foarte similară cu cea apărută în formularea Lagrange mecanică. În rezolvarea problemei duală apare frecvent cazul în care doar o submulţime de restricţii devine activă. De exemplu, dacă vrem să păstrăm o bilă într-o cutie atunci aceasta în mod uzual se va rostogoli într-un colţ. Restricţiile corespund pereţilor care nu sunt atinşi de bilă şi care sunt nerelevanţi în acel context deci pot fi eliminaţi.

Totul a fost formulat în produse scalare. La nivel practic aceasta oferă posibilitatea ca algoritmul să lucreze într-un spaţiu de dimensiune mare. Astfel noile şabloane $\Phi(x_i)$ pot fi rezultatul mapării datelor de intrare originale x_i într-un spaţiu de dimensiune mai mare utilizând funcţia Φ. Maximizarea funcţiei ţintă şi evaluarea funcţiei de decizie implică calculul produsului scalar $\langle \phi(x), \phi(x) \rangle$ într-un spaţiu de dimensiune mai mare. Aceste calcule costisitoare sunt reduse semnificativ prin utilizarea unui nucleu pozitiv definit *k*, astfel ca $k(x, x') \coloneqq \langle x, x' \rangle$. Această substituire, care este referită uneori ca trucul nucleu, este utilizată pentru a extinde clasificarea cu hiperplane la SVM-ul neliniar. Trucul nucleu poate fi aplicat de vreme ce toţi vectorii de trăsături apar doar în produse scalare. Vectorii de trăsături devin o expresie în spaţiul de trăsături şi aşadar Φ va fi funcţia prin care reprezentăm vectorii de intrare în noul spaţiu. Astfel funcţia de decizie va avea următoarea formă:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b \right) \tag{3.9}$$

Principalul avantaj al acestui algoritm este că nu necesită transpunerea tuturor datelor de intrare într-un spaţiu de dimensiune mai mare. De aceea nu vor exista nici calcule costisitoare ca in cazul reţelelor neuronale. De asemenea obţinem o micşorare a setului de date în faza de antrenare când vom considera doar vectorii suport care de obicei sunt în număr redus. Un alt avantaj al acestui algoritm este că permite utilizarea datelor de intrare cu un număr oricât de mare de trăsături fără a creşte exponenţial timpii de antrenare. Această caracteristică nu este adevărată pentru reţelele neurale, de exemplu algoritmul backprogapagion are probleme când trebuie să lucreze cu un numnăr mare de trăsături. Singura problemă care apare la SVM este numărul de vectori suport rezultaţi. Când numărul acestoar creşte timpul de răspuns în faza de testare creşte şi el liniar.

3.2 Clasificare în mai multe clase

Majoritatea problemelor care trebuiesc rezolvate necesită clasificare în mai mult de două clase. Algoritmul prezentat este conceput doar pentru clasificare în două clase, unde etichetele claselor sunt ±1.Există câteva metode clasice care permit ca algoritmii care sunt dezvoltaţi pentru două clase să poate lucra cu mai multe clase. Unele dintre aceste metode sunt: o clasă versus celelalte clase, clasificare pe perechi de clase, codificarea ieşirii pe baza corecţiei erorii şi funcţiile obiectiv pe mai mute clase.

Cea mai utilizată metodă constă în clasificare „o clasă versus celelalte clase” în care elementele care aparţin unei clase sunt diferenţiate de celelalte elemente. În acest caz putem calcula un hiperplan optim care separă fiecare clasă de celelalte. La ieşirea algoritmului vom alege valoarea maximă obţinută de toate funcţiile de decizie existente. Pentru a obţine o clasificare în M clase, construim o mulţime de clasificatori binari în care fiecare clasificator este antrenat separat pentru fiecare clasă comparativ cu celelalte clase. După aceea îi combinăm cu scopul de a obţine o clasificare în mai multe clase care va corespunde ieşirii maxime înainte de aplicarea funcţiei semn:

$$\arg\max_{j=1,..,M} g^j(x), \quad where\ g^j(x) = \sum_{i=1}^{m} y_i \alpha_i^j k(x, x_i) + b^j \tag{3.10}$$

şi

$$f^j(x) = \text{sgn}(g^j(x)) \tag{3.11}$$

unde *m* reprezintă numărul de vectori de intrare. Această problemă are o complexitate liniară pentru M clase trebuind să calculăm M hiperplane.

3.3 Clustering utilizând tehnica vectorilor suport

Până în acest moment am analizat clasificarea ca un proces de învățare supervizată care lucrează cu date etichetate. În continuare voi analiza clustering-ul care este un proces de învățare nesupervizată care lucrează cu date neetichetate. Vapnik prezintă în [Vap01] o alternativă la algoritmul clasic care poate utiliza date neetichetate. În acest algoritm găsirea hiperplanului se transformă în găsirea unei sfere de dimensiune maximă cu cost minim care grupează cele mai multe date asemănătoare. În continuare voi prezenta această abordare.

Pentru acest algoritm, datele de intrare vor fi transpuse într-un spațiu de dimensiune mai mare utilizând nucleul Gaussian. În acest spațiu vom încerca să găsim o sferă de dimensiune minimă care include noua imagine a datelor. Acest lucru este posibil deoarece datele sunt generate cu aceeaşi distribuție şi când ele sunt transpuse într-un spațiu de dimensiune mai mare vor fi grupate într-un cluster. După calcularea dimensiunii sferei datele vor fi remapate în spațiul original. Granița sferei va fi transpusă în una sau mai multe granițe care vor conține datele clasificate. Granițele rezultate pot fi considerate ca şi margini ale clusterului în spațiul de intrare. Punctele care aparțin aceluiaşi cluster vor avea aceeaşi margine. Cu cât parametrul de lărgime din nucleul Gaussian scade cu atât creşte numărul de contururi deconectate din spațiul de intrare. Când parametrul de lărgime creşte se vor obține suprapuneri ale clusterelor. Vom utiliza următoarea formă pentru nucleul Gaussian:

$$\Phi(x, x') = e^{-\frac{\|x-x'\|^2}{2\cdot\sigma^2}} \tag{3.12}$$

unde σ reprezintă dispersia. Observăm că dacă σ^2 descreşte, exponentul creşte în valoare absolută şi astfel valoarea nucleului va tinde la 0. Acesta va transfera datele într-un spațiu de dimensiune mai mică în loc de un spațiu de dimensiune mai mare. Matematic vorbind algoritmul încearcă să găsească „văile” distribuției cu care sunt generate datele de antrenare.

3.4 SMO – Optimizarea minimă secvențial

SMO (Sequential Minimal Optimization) reprezintă una din metodele de implementare a problemei de programare pătratică introdusă de Platt şi folosită de mine în această teză. Strategia pentru SOM este de a împărți constrângerile în grupuri cât mai mici posibile. Observăm că nu este posibil să modificăm individual variabila α_i fără a modifica suma constrângerilor (condițiile KKT). Astfel am generat o problemă de optimizare convexă cu constrângeri pentru perechi de variabile. Vom considera optimizarea pentru două variabile α_i şi α_j în timp ce celelalte variabile sunt considerate fixe, optimizând funcția de decizie în raport cu acestea. Algoritmul este următorul: prima dată rezolvăm problema de optimizare pătratică pentru două variabile şi mai târziu determinăm valoarea funcției de decizie pentru problema generică. Ajustăm pe *b* în acord cu modificările aduse şi alegem exemplele astfel ca să se asigure o convergență rapidă.

Astfel problema prezentată mai sus implică rezolvarea problemei de optimizare analitică pentru două variabile. Singura problemă rămasă acum este alegerea ordinii în care exemplele de antrenament sunt analizate pentru a avea o convergență rapidă. Prin utilizarea doar a două exemple la un moment dat avem avantajul de a lucra cu funcții de decizie liniare care sunt mai simplu de rezolvat decât problema de optimizare pătratică în ansamblu. Un alt avantaj pentru SOM este că nu toate datele de antrenament trebuie să se găsească simultan în memorie ci doar eşantioanele pentru

care se optimizează funcția de decizie la acel moment. Aceasta ne permite ca să lucrăm cu eşantioane de dimensiune foarte mare.

3.5 Tipurile de nuclee folosite

În această teză de doctorat am testat o idee nouă de corelare a parametrilor nucleelor. Astfel pentru nucleul polinomial am corelat gradul nucleului cu deplasamentul acestuia şi pentru nucleul Gaussian am corelat parametrul acestuia cu dimensiunea vectorilor de intrare.

Pentru nucleul polinomial singurul parametru care trebuie modificat este gradul iar pentru nucleul Gaussian rămâne de modificat parametru *C* după următoarele formule:

- Polinomial:

$$k(x,x') = \left(2 \cdot d + \langle x \cdot x' \rangle\right)^d \tag{3.13}$$

  - *d* fiind sigurul parametrul care trebuie modificat.

- Gaussian (radial basis function RBF):

$$k(x,x') = \exp\left(-\frac{\|x-x'\|^2}{n \cdot C}\right) \tag{3.14}$$

  - *C* fiind singurul parametru care trebuie modificat şi *n* reprezentând numărul de elemente care au ponderea mai mare ca 0 din vectorii de intrare.

În ambele formule *x* şi *x'* reprezintă vectorii de intrare

Un alt nucleu interesant utilizat în mod constant în literatură este nucleul liniar. Pentru acest tip de nucleu eu am utiliza formula nucleului polinomial cu gradul (parametrul *d*) 1. Acest nucleu va fi utilizat atât în teste cât mai ales în pasul de selecție a trăsăturilor caracteristice unde a fost utilizat pentru calculul vectorului pondere **w**.

3.6 Corelația parametrilor nucleelor

Nucleul este folosit pentru a calcula norma diferenței între doi vectori într-un spațiu de dimensiune mai mare fără a reprezenta vectorii în acel spațiu. Adăugând un scalar constant la acest nucleu se observă o îmbunătățire a rezultatelor clasificării. Aici am dorit să prezint o nouă idee de a corela acest scalar cu dimensiunea spațiului în care datele de intrare vor fi reprezentate. Astfel am considerat că aceşti doi parametrii (gradul şi scalarul) trebuie să fie corelați.

3.6.1 Corelația parametrilor nucleului polinomial

De obicei când se utilizează nucleul polinomial cercetătorii utilizează un nucleu de forma:

$$k(\mathbf{x},\mathbf{x}') = \left(\langle \mathbf{x} \cdot \mathbf{x}' \rangle + b\right)^d \tag{3.15}$$

unde *d* şi *b* sunt parametri independenți. "*d*" reprezintă gradul nucleului şi este parametrul care este utilizat pentru a ne ajuta să transpunem datele din spațiul de intrare într-un spațiu de dimensiune mai mare. Acest parametru este intuitiv de ales. Al doilea parametru "*b*", nu este la fel de uşor de ales. În toate articolele de cercetare acest parametru este utilizat dar nicăieri nu se prezintă o metodă pentru selectarea lui fiind de obicei ales ca fiind egal cu numărul de trăsături al vectorilor de intrare. Am observat că dacă acest parametru este eliminat se obțin rezultate mai slabe din punct de vedere al acurateții clasificării. De aceea am încercat să corelez aceşti doi parametrii astfel ca deplasamentul *b* să fie şi el modificat o dată cu modificarea dimensiunii

spațiului (parametrul *d)*. Astfel, pe baza mai multor teste prezentate în capitolul 6, am sugerat utilizarea "*b=2*d"* în aplicație.

3.6.2 Corelația parametrilor nucleului Gaussian

Am modificat de asemenea nucleul Gaussian față de cel standard utilizat de comunitatea de cercetare. În mod uzual nucleul arată de forma:

$$k(x,x') = \exp\left(-\frac{\|x-x'\|}{C}\right) \tag{3.16}$$

unde parametrul *C* reprezintă numărul de trăsături din setul de antrenament (şi este de obicei un număr foarte mare în cazul problemelor de clasificare a documentelor text). Folosirea unei valori atât de mari în cadrul documentelor text poate duce la rezultate foarte slabe calitativ. De asemenea dimensiunea vectorilor poate fi foarte diferită pe acelaşi set de date şi de aceea am introdus un parametru nou (*n*) care reprezintă numărul de trăsături distincte din cei doi vectori de intrare (*x* şi *x'*). a căror valoare este mai mare de 0 Acest nou parametru (*n)* corelează nucleul cu dimensiunea celor doi vectori de intrare folosiți şi este multiplicat printr-un nou parametru notat de asemenea cu *C*. Am păstrat acest parametru pentru a avea un parametru de reglaj al nucleului Gaussian iar acesta va avea de obicei o valoare foarte mică.

După câte ştiu sunt primul autor care propune o corelație între parametri atât pentru nucleul polinomial cât şi pentru cel Gaussian.

4 Câteva metode dezvoltate pentru selecția trăsăturilor caracteristice

4.1 Reducerea datelor

Selecția unei submulțimi de trăsături caracteristice este definită ca procesul de selecție a acelei submulțimi de trăsături *d* dintr-un set de dimensiune mult mai mare *D* care maximizează performanțele de clasificare peste toate submulțimile posibile. Căutarea după o astfel de submulțime este o problemă foarte dificilă. Spațiul de căutare poate fi foarte mare, de exemplu în cazul nostru folosind baza de date Reuters dimensiunea spațiului de căutare are 2^{19034} combinații posibile!

4.2 Selecția aleatoare (RAN)

Am ales această metodă simplă doar pentru a avea o bază de comparare (limită inferioară) în evaluarea câştigului de performanță introdus de celelalte trei metode prezentate. În această metodă de selecție a trăsăturilor, valori (ponderi) aleatoare între 0 şi 1 sunt atribuite fiecărei trăsături. Mulțimile de antrenare şi testare de dimensiune variabilă sunt alese prin selectarea trăsăturilor în funcție de valoarea descendentă a ponderii. Aceste mulțimi (de dimensiune variabilă) sunt obținute în aşa fel încât mulțimile de dimensiuni mai mari vor conține mulțimile de dimensiuni mai mici. După fiecare pas de selecție am făcut un pas de clasificare şi am calculat acuratețea clasificării. Am repetat acest proces(selecție plus clasificare) de trei ori. Acuratețea finală a clasificării este calculată ca medie peste cele trei valori ale acurateții clasificării obținute individual pentru fiecare pas de selecție. Această valoare este considerată acuratețea clasificări pentru selecția aleatoare.

4.3 Entropia şi Câştigul Informațional (IG)

Câştigul informațional şi entropia sunt funcții ale distribuției probabilistice care susțin procesul de comunicare. Entropia este o măsură a incertitudini unei variabile aleatoare. Dându-se o colecție S de *n* eşantioane grupate în *c* concepte țintă (clase), entropia lui S relativă la clasificare este:

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2(p_i) \tag{4.1}$$

unde p_i este procentajul din *S* care aparține clasei *i*.

Pe baza entropiei este definită o măsură a gradului de eficiență în selecția trăsăturilor. Această măsură este numită *Câştigul informațional* şi reprezintă de fapt reducerea în entropie, cauzată de gruparea eşantioanelor în acord cu un atribut. Mai precis, câştigul informațional pentru un atribut relativ la o mulțime de eşantioane S este definit ca:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{4.2}$$

unde *Values(A)* este mulțimea de valori posibile pentru atributul A şi S_v este submulțimea din S pentru care atributul A are valoarea v.

Utilizând ecuația 4.2, pentru fiecare trăsătură se calculează câştigul informațional obținut dacă mulțimea este împărțită utilizând acea trăsătură. Se obțin valori între 0 şi 1 fiind apropiate de 1 dacă trăsătura împarte mulțimea originală în două submulțimi cu dimensiuni apropiate. Pentru selectarea trăsăturilor relevante am utilizat diferite praguri. Dacă câştigul informațional obținut pentru o trăsătură depăşeşte pragul acea trăsătură va fi selectată, altfel nu va fi selectată.

Forman în [For04] a arătat că câştigul informaţional eşuează în cazul problemelor reale de clasificare când se lucrează cu multe trăsături. Autorul atribuie acest eşec proprietății multor metode de selecție a trăsăturilor care ignoră sau elimină trăsături necesare pentru discriminarea claselor dificile.

4.4 Selecția trăsăturilor utilizând SVM (SVM_FS)

Pentru această metodă am utilizat în pasul de selecție a trăsăturilor algoritmul de clasificare bazat pe SVM folosind nucleul liniar. În această metodă am fost interesat în special în găsirea vectorului pondere care caracterizează hiperplanul de separare a exemplelor pozitive de cele negative din mulțimea de antrenament. Pentru aceasta am utilizat următoarea formulă a nucleului liniar:

$$k(w,x) = \left(2 + \langle w \cdot x \rangle\right) \tag{4.3}$$

unde **w** reprezintă vectorul pondere perpendicular pe hiperplanul de separare iar *x* reprezintă vectorul de intrare. Am adăugat la nucleu constanta 2 deoarece doresc ca acest nucleu să utilizeze corelația parametrilor prezentată în secțiunea 3.6 şi aşa cum voi arăta această corelație va duce la obținerea celor mai bune rezultate. Deoarece în mulțimea de antrenament am utilizat mai mult de două categorii, pentru fiecare categorie se va obține câte o funcție de decizie utilizând nucleul liniar. Algoritmul SVM asigură că, după pasul de antrenare, trăsăturile care au o influență mai mare în trasarea hiperplanului vor avea o valoare absolută a ponderii mai mare şi trăsăturile care nu au nici o influență în trasarea hiperplanului vor avea valoare ponderii apropiată de 0. Astfel pasul de selecție a trăsăturilor devine un pas de învățare în care se utilizează toate trăsăturile (în cazul prezentat 19038) şi se încearcă găsirea acelui hiperplan care desparte cel mai bine eşantioanele din clasa pozitivă de cele din clasa negativă. Câte un hiperplan este găsit pentru fiecare categorie în parte. Vectorul pondere rezultat din funcția de decizie are aceeaşi dimensiune cu a spațiului de trăsături deoarece se lucrează doar în spațiul de intrare. Vectorul pondere final este obținut ca o sumă a tuturor vectorilor pondere obținuți pentru fiecare categorie în parte. Utilizând acest vector pondere normalizat selectez doar acele trăsături a căror valoare absolută a ponderii depăşeşte un prag specificat. Mulțimea rezultată are o dimensiune mai mică din punct de vedere al numărului de trăsăturilor şi este utilizată în paşi de antrenare şi testare ai algoritmului pentru clasificarea datelor şi calcularea acurateței clasificării.

4.5 Selecția trăsăturilor utilizând algoritmi genetici (GA_FS)

Algoritmii genetici codifică soluția potențială la o problemă specifică într-un cromozom ca o structură de date şi aplică operatorii genetici la aceste structuri în aşa fel încât să păstreze informațiile critice. În problema mea de selecție a trăsăturilor caracteristice, cromozomul este considerat ca având următoarea structură:

$$c = \left(w_1, w_2, ..., w_n, b\right) \tag{4.4}$$

unde $w_i, i = \overline{1,n}$ reprezintă ponderea pentru fiecare trăsătură şi *b* reprezintă deplasamentul pentru hiperplanul din algoritmul SVM. Considerăm că mulțimea de antrenament este de forma $\left\{\langle \vec{x}_i, y_i \rangle, i = 1,...,m\right\}$, unde y_i reprezintă ieşirea pentru fiecare exemplu de intrare $\vec{x}_i$, şi poate lua doar valorile -1 sau +1. Am ales această formă a cromozomului pentru a facilita utilizarea SVM-ului în calculul funcției de performanță (fitness). În acest mod am păstrat în cromozom doar parametrii care se modifică în funcția de decizie din SVM. Soluția potențială codifică parametrii hiperplanului de separare *w* şi *b*. La sfârşitul algoritmului, cel mai bun candidat din toate generațiile ne oferă valorile optime pentru orientarea hiperplanului prin vectorul pondere *w* şi deplasamentul hiperplanului *b*. Urmând ideea propusă pentru clasificarea în mai mult de două

clase ("una versus celelalte clase"), am încercat să găsesc cel mai bun cromozom pentru fiecare dintre cele 24 clase distincte Reuters. Pentru fiecare clasă am pornit cu o generație de 100 cromozomi, fiecare dintre ei având valori generate aleator între -1 şi +1.

Utilizând în algoritmul SVM nucleul liniar de forma $\langle w,x\rangle + b$ putem calcula funcția de performanță pentru fiecare cromozom. Evaluarea folosind funcția de performanță este definită ca:

$$f(c) = f((w_1, w_2, ..., w_n, b)) = \langle \mathbf{w}, \mathbf{x} \rangle + b \qquad (4.5)$$

unde **x** reprezintă eşantionul curent şi *n* reprezintă numărul de trăsături. În pasul următor generăm următoarea populație utilizând operatorii genetici de selecție, mutație sau crossover. Procesul se opreşte după ce s-au parcurs un număr predefinit de paşi sau după ce în ultimii 20 de paşi nu s-a mai adus nici o îmbunătățire.

La sfârşitul algoritmului, obținem pentru fiecare clasă cel mai bun cromozom care caracterizează cea mai bună funcție de decizie obținută. După aceea normalizăm fiecare vector pondere în scopul de a obține valori ale ponderii între 0 şi 1. Pentru selecția celor mai bune trăsături facem o medie peste toți cei 24 vectori pondere obținuți şi selectăm acele trăsăturile în acord cu valoarea descendentă a ponderilor. Metoda dezvoltată este detailată în [Mor06_5].

4.6 Selecția submulțimii de trăsături. Abordări comparative

Pentru o comparație corectă între metodele prezentate de selecție a trăsăturilor, trebuie să se utilizeze acelaşi număr de trăsături pentru fiecare metodă. De aceea pentru metoda bazată pe „Câştigul Informațional" pragul pentru selecția trăsăturilor relevante reprezintă o valoare între 0 şi 1, urmând să se selecteze acele trăsături pentru care ponderea calculată depăşeşte acest prag. Pentru celelalte trei metode prezentate pragul reprezintă numărul de trăsături care se doreşte a fi extras în acord cu valoarea descrescătoare a ponderilor. Acest număr este egal cu numărul de trăsături obținute utilizând metoda bazată pe câştigul informațional.

4.6.1 Influența dimensiunii numărului de trăsături

În continuare voi prezenta influența numărului de trăsături asupra acurateței clasificării pentru fiecare reprezentare a datelor de intrare şi pentru fiecare dintre metodele de selecție a trăsăturilor propuse. În aceste rezultate voi lua în considerare toate cele 24 de clase distincte. Voi prezenta rezultatele obținute pentru gradului nucleului polinomial egal cu 2 şi pentru parametrul *C* din nucleul Gaussian egal cu 1.3 deoarece sunt interesant să prezint doar influența metodelor de selecție a trăsăturilor asupra acurateței clasificării. În următorul capitol voi prezenta mult mai multe rezultate pentru diferite valori ale parametrilor nucleelor. Aici prezint rezultate doar pentru parametrii pentru care se obțin cele mai bune valori.

În Figura 4.1 sunt prezentate rezultatele obținute pentru nucleul polinomial cu gradul 2 şi reprezentare binară a datelor. Pentru un număr mic de trăsături cele mai bune valori se obțin utilizând metoda SVM_FS iar când numărul de trăsături creşte metoda IG obține cele mai bune rezultate.

Performanțele clasificării aşa cum pot fi observate din Figura 4.1 şi din Figura 4.2 nu cresc când numărul de trăsături creşte (în special pentru SVM_FS). Am observat că este o mică creştere în performanțe când am crescut procentuajul de trăsături extrase din setul inițial de la 2.5% (însemnând 475 trăsături) la 7% (însemnând 1309 trăsături) pentru nucleul polinomial. Dacă sunt alese mai mult de 42% de valori din setul inițial, acuratețea finală are o uşoară scădere pentru majoritatea metodelor de selecție a trăsăturilor. Aceasta poate apărea deoarece trăsăturile suplimentare adăugate pot fi considerate zgomot pentru procesul de clasificare curent din punct de vedere a ceea ce se învață. Aşa cum mă aşteptam, pentru selecția aleatoare a trăsăturilor valoarea acurateței este foarte mică în comparație cu celelalte metode. Celelalte metode IG, SVM_FS şi

GA_FS obțin rezultate comparabile. SVM_FS obține rezultate mai bune decât IG pentru nucleul polinomial şi Gaussian şi respectiv obține cele mai bune rezultate pentru un număr mic de trăsături. Utilizarea unui număr mic de trăsături are avantajul unui timp de antrenare şi testare scăzut.

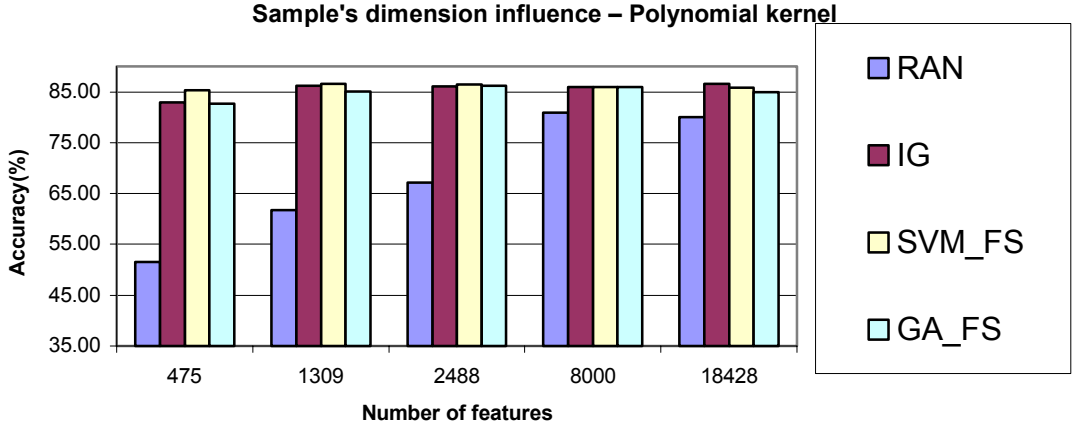

Figura 4.1 – Influența numărului de trăsături asupra acurateței clasificării utilizând nucleul polinomial cu gradul 2 şi reprezentarea binară a datelor

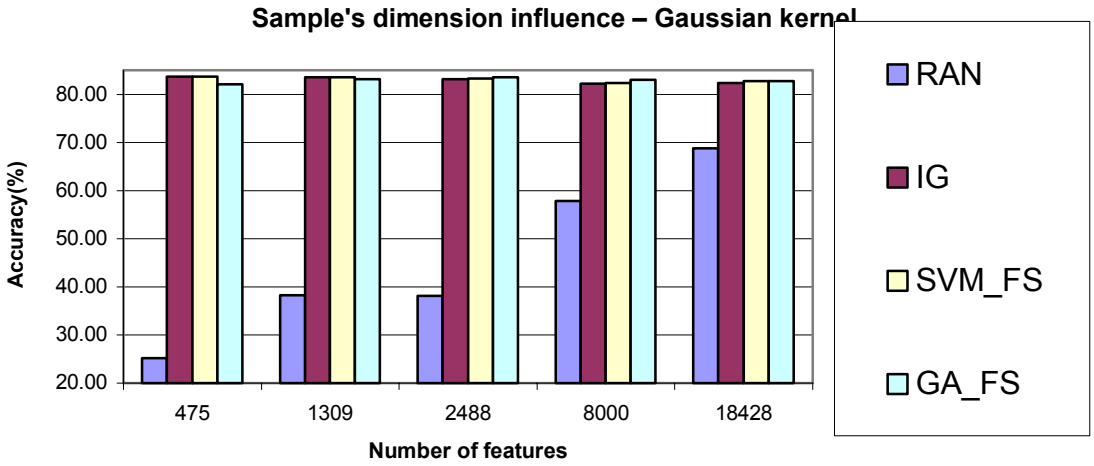

Figura 4.2 – Influența numărului de trăsături asupra acurateței clasificării utilizând nucleul Gaussian cu parametrul *C* egal cu 1.3 şi reprezentarea binară a datelor

Algoritmul SVM depinde, în pasul de antrenare, de ordinea selectării vectorilor de intrare, găsind diferite hiperplane optime când datele de intrare sunt selectate într-o ordine diferită. Algoritmul genetic cu funcția SVM pentru calculul performanței stipulează acest lucru în pasul de selecție al trăsăturilor. SVM_FS şi GA_FS obțin rezultate comparabile care sunt în mod constant mai bune comparativ cu cele obținute folosind metoda IG. Aşa cum se poate observa GA_FS obține rezultate mai bune cu nucleul Gaussian comparativ cu celelalte trei metode de selecție. GA_FS este în medie mai bun cu 1% comparativ cu SVM_FS (84.27% pentru GA_FS în comparație cu doar 83.19% pentru SVM_FS) şi cu 1.7% comparativ cu IG (84.27% pentru GA_FS şi 82.58% pentru IG). Pentru nucleul polinomial SVM_FS obține în medie cu 0.9% rezultate mai bune comparativ cu IG (de la 86.24% pentru SVM_FS la 85.31% pentru IG) şi cu 0.8% comparativ cu GA_SVM (de la 86.24% la 85.40%). În aproape toate cazurile cele mai bune rezultate se obțin pentru un număr mic de trăsături selectate (în medie pentru 1309 trăsături).

În Figura 4.3 sunt prezentați timpii de antrenare ai clasificării în funcție de numărul de trăsături selectate pentru fiecare dintre metodele prezentate utilizând nucleul polinomial cu gradul 2. După

cum se poate observa, timpul de antrenare creşte cu aproape 3 minute când numărul de trăsături creşte de la 475 la 1309 şi cu aproape 32 minute când numărul de trăsături creşte de la 1309 la 2488, pentru metoda SVM_FS. De asemenea timpul necesar antrenării cu trăsăturile selectate folosind IG este de obicei mai mare decât timpul necesar pentru antrenare cu trăsăturile selectate folosind SVM_FS. Când numărul de trăsături creşte cel mai bun timp de antrenare a fost obținut pentru trăsăturile selectate folosind metoda GA_FS.

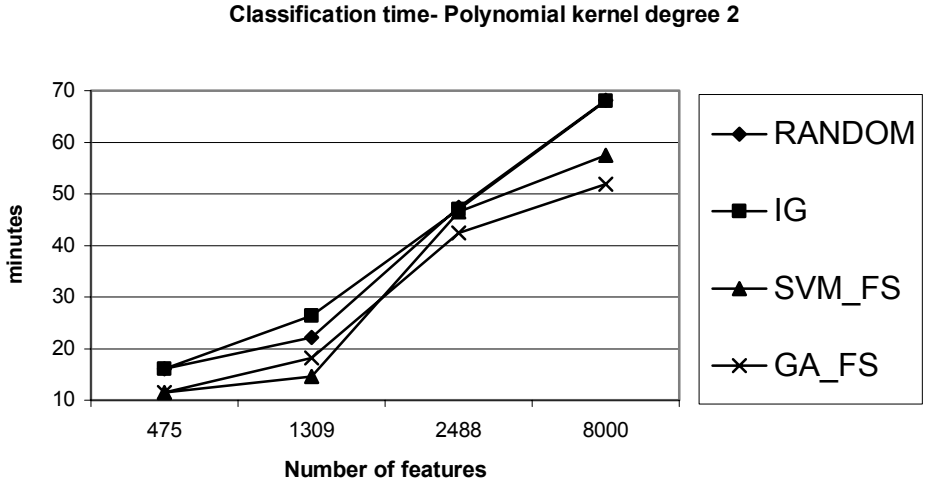

Figura 4.3 – Timpii de antrenare funcție de numărul de trăsături selectate

Pentru nucleul Gaussian timpul de antrenare este în medie (pentru toate testele făcute) cu 20 de minute mai mare decât timpul necesar pentru nucleul polinomial pentru toate metodele de selecție a trăsăturilor. Valorile au fost obținute pe un calculator Pentium IV la 3.4 GHz, cu 1GB DRAM, 512KB cache şi WinXP.

4.6.2 Influența gradului nucleului polinomial

În continuare voi prezenta unele rezultate obținute pentru nucleul polinomial şi reprezentarea nominală a datelor pentru diferite grade ale nucleului (de la 1 la 5). Sunt de asemenea prezentate câteva rezultate obținute pentru nucleul Gaussian pentru diferite valori ale parametrului *C* (1.0, 1.3, 1.8, 2.1, şi 2.8) şi reprezentarea binară a datelor. Pentru nucleul polinomial am ales să prezint rezultatele doar pentru reprezentarea nominală a datelor deoarece, după cum vom vedea în secțiunea 5.6, cu această reprezentare se obțin cele mai bune rezultate. De asemenea, nucleul Gaussian obține în medie cele mai bune rezultate folosind reprezentarea binară a datelor. În Figura 4.4 sunt prezentate rezultatele obținute pentru o mulțime având 475 trăsături selectate cu fiecare dintre cele 4 metode prezentate şi nucleul polinomial. Pentru această dimensiune a setului de trăsături cele mai bune valori se obțin utilizând metoda SVM_FS indiferent de gradul nucleului.

Ca o concluzie pentru nucleul polinomial SVM_FS se obțin cele mai bune rezultate comparativ cu GA_FS pentru dimensiuni mici ale vectorilor de intrare. În momentul în care dimensiunea vectorilor de intrare creşte (selectăm mai multe trăsături) se obțin rezultate mai bune folosind GA_FS. Aceasta poate apare deoarece GA_FS are un punct de plecare aleator. Când numărul de trăsături creşte probabilitatea de a alege trăsături mai bune creşte de asemenea.

O altă concluzie interesanta este că pentru SVM_FS, pentru dimensiuni mici ale setului de intrare, se obțin cele mai bune rezultate şi de asemenea cei mai buni timpi de antrenare. Când dimensiunea vectorilor de intrare creşte cele mai bune rezultate şi cei mai mici timpi de antrenare se obțin pentru GA_FS. Deci de aici se poate trage concluzia că o selecție bună a trăsăturii poate îmbunătății acuratețea clasificării şi reduce timpii de antrenare.

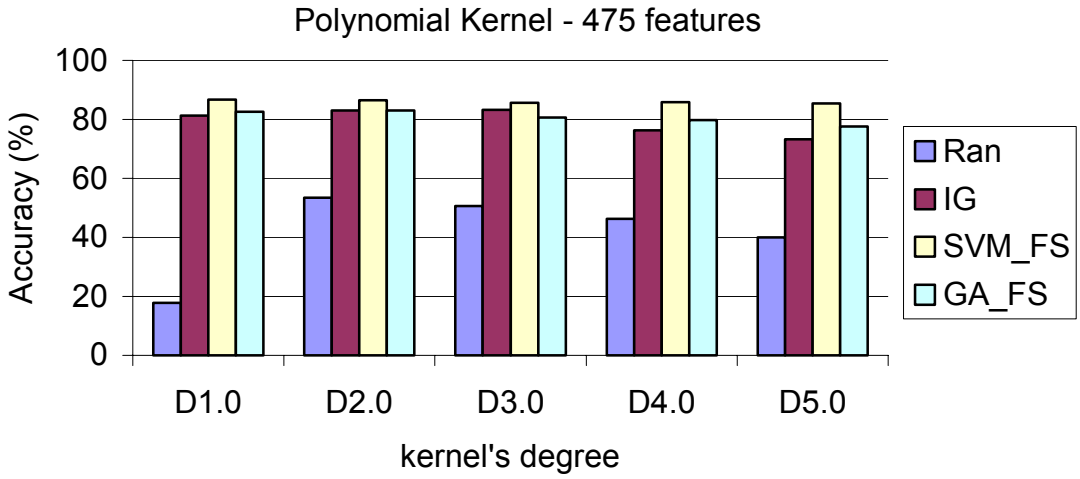

Figura 4.4 – Influența metodei de selecție a trăsăturilor şi a gradului nucleului asupra acurateței clasificării

4.6.3 Influența parametrului *C* pentru nucleul Gaussian

În următorul grafic am prezentat comparativ rezultatele obținute pentru nucleul Gaussian şi diferite valori ale parametrului *C* folosind reprezentarea binară a datelor de intrare.

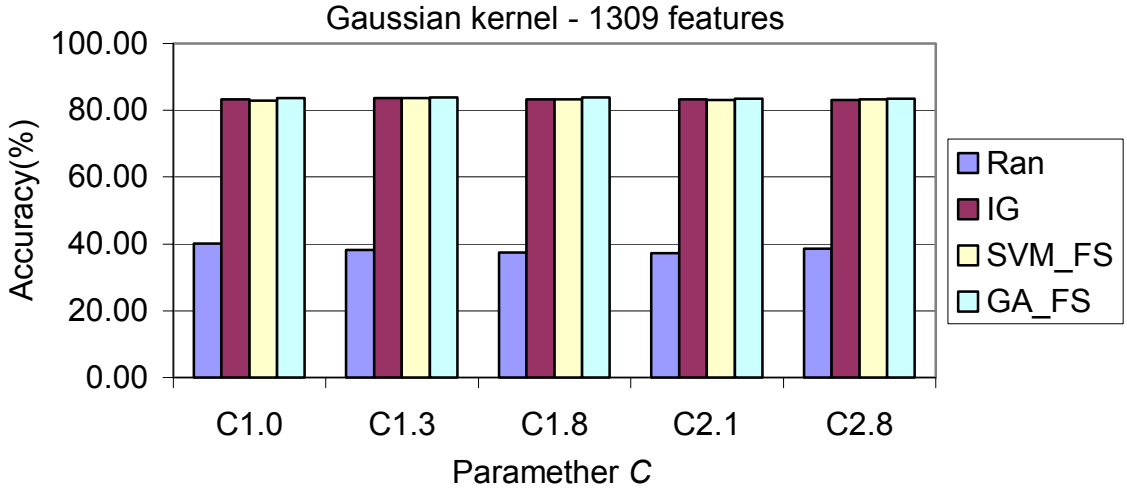

Figura 4.5 – Comparație între metodele de selecție a trăsăturilor pentru 1309 trăsături şi nucleul Gaussian

Când numărul de trăsături creşte la 1309 (Figura 4.5) metoda GA_FS obține rezultate cu 0.22% mai bune decât SVM_FS. Astfel GA_FS obține o acuratețe de 83.96% pentru *C*=1.3 iar SVM_FS obține doar 83.74% tot pentru *C*=1.3 şi reprezentarea binară a datelor. Dar, după cum se poate observa din grafic, pentru toate testele realizate GA_FS obține cele mai bune rezultate. Pentru această dimensiune a setului IG obține doar 83.62% acuratețe a clasificării.

Când numărul de trăsături creşte şi mai mult acuratețea maximă obținută nu mai creşte. Ea rămâne la valoarea de 84.85% pentru GA_FS cu aceleaşi caracteristici ca cele din graficul anterior. Pentru metoda SVM_FS când numărul de trăsături creşte (la 2488) acuratețea clasificări descreşte la valoarea de 82.47%. De asemenea pentru metoda IG acuratețea clasificări descreşte la 82.51% pentru *C*=1.8 şi pentru un număr de 2488 trăsături.

Pentru nucleul Gaussian rezultatele obținute cu GA_FS sunt mai bune pentru toate dimensiunile testate în comparație cu SVM_FS. De asemenea interesant, această metodă obține rezultate mai bune cu o formă simplificată de reprezentare a datelor – forma binară.

Comparând cele două tipuri de nuclee folosite, cele mai bune rezultate sunt obținute de către nucleul polinomial cu un grad al nucleului mic (86.68% pentru 1309 trăsături şi gradul egal cu 2 folosind metoda SVM_FS). Pentru nucleul Gaussian am obținut un maxim de 84.84% pentru 2488 trăsături şi *C*=1.8 utilizând metoda GA_FS. În Tabelul 4.1 se prezintă mediile acurateței clasificării pentru toate rezultatele obținute pentru fiecare dimensiune a setului şi pentru fiecare tip de nucleu.

Tip nucleu	Nr. De trăsături	RAN	IG	SVM_FS	GA_FS
Polinomial	475	39.30%	76.81%	83.38%	71.20%
	1309	44.72%	80.17%	82.92%	78.46%
	2488	52.07%	81.10%	81.88%	74.49%
	8000	66.65%	77.23%	77.33%	75.85%
Gaussian	475	26.51%	82.77%	83.00%	81.61%
	1309	38.49%	83.25%	83.27%	83.23%
	2488	40.31%	83.07%	82.85%	83.75%
	8000	51.52%	83.20%	82.45%	83.75%

Tabelul 4.1 – Mediile acurateților obținute pentru toate testele realizate pentru fiecare dimensiune a mulțimi de trăsături şi fiecare tip de nucleu

După cum se poate observa metoda GA_FS obține rezultatele cele mai bune pentru nucleul Gaussian şi pentru un număr relativ mare de trăsături. Putem de asemenea observa că pentru SVM_FS dimensiunea optimă a spațiului de trăsături este una mică (în jur de 4% din numărul total de trăsături). Pentru nucleul Gaussian diferențele dintre rezultatele obținute de către GA_FS şi SVM_FS nu sunt mai mari de 1.5%. IG obține rezultate comparabile cu SVM_FS pentru nucleul Gaussian dar rezultate mult mai proaste pentru nucleul polinomial. Metoda de selecție aleatoare obține întotdeauna cele mai proaste rezultate, fapt absolut previzibil.

4.6.4 Influența numărului de trăsături pentru documentele Web

Pentru acest set de date am ales doar metoda SVM_FS pentru selecția trăsăturilor caracteristice relevante deoarece în pasul anterior am obținut în medie cele mai bune rezultate. Deoarece şi în această mulțime de test am mai mult de două clase, am utilizat metoda de clasificare în mai multe clase care a fost prezentată în secțiunea 3.2. Voi prezenta rezultatele pentru toate cele trei tipuri de reprezentare a datelor: binară, nominală şi Cornell Smart, prezentate în secțiunea 2.4.4 şi pentru două tipuri de nuclee.

După pasul de selecție a trăsăturilor am utilizat diferite dimensiuni ale vectorilor de intrare. Astfel utilizând valori diferite ale pragului am selectat 8 dimensiuni diferite ale vectorilor: 500, 1000, 1500, 2000, 2500, 5000, 10000 şi 25646. Am observat de asemenea că şi pentru acest tip de fişiere când numărul de trăsături creşte acuratețea clasificări descreşte după cum se poate observa şi din Figura 4.6 şi Figura 4.7.

În Figura 4.6 cele mai bune rezultate se obțin pentru o dimensiune mică a vectorilor de trăsături (intre 500 şi 1500) pentru toate valorile gradului nucleului. De asemenea o observație interesantă apare pentru gradul 1 şi reprezentarea binară a datelor deoarece chiar dacă dimensiunea vectorului este foarte mică se obțin cele mai bune rezultate. Aceasta arată că fişierele HTML sunt de obicei cvasi-liniar separabile şi conțin multe cuvinte redundante, concluzie care a fost de asemenea obținută şi pentru baza de date Reuters. De asemenea este interesant faptul că o dată cu creşterea gradului nucleului polinomial descreşte şi acuratețea clasificării.

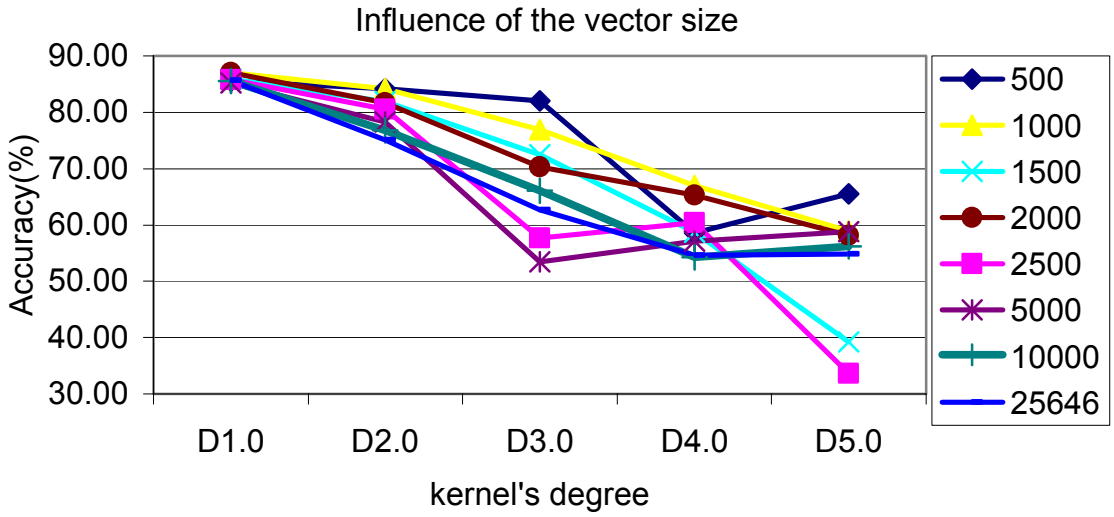

Figura 4.6 – Influența dimensiuni vectorului pentru nucleul polinomial şi reprezentarea binară a datelor

În Figura 4.7 sunt prezentate rezultatele obținute pentru nucleul Gaussian şi reprezentarea Cornel Smart a datelor de intrare dintr-un alt punct de vedere. Pentru fiecare valoare a parametrului *C* sunt prezentate rezultatele obținute pentru fiecare dimensiune a vectorului. Deci este mai uşor să observăm tendința de descreştere a acurateței clasificări când numărul de trăsături creşte.

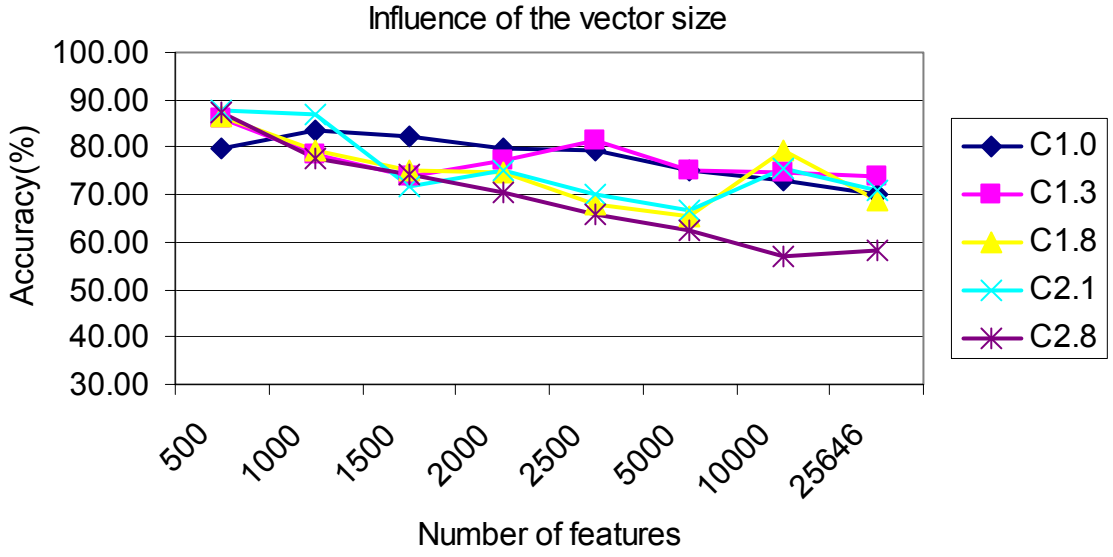

Figura 4.7 – Influența dimensiuni vectorului pentru nucleul Gaussian şi reprezentarea Cornell Smart

Analizând toate graficele prezentate putem observa că toate valorile maxime ale acurateței clasificări sunt obținute pentru dimensiuni mici ale vectorilor de intrare de obicei între 500 şi 2000. De exemplu pentru nucleul polinomial şi reprezentarea Cornell Smart pentru gradele 1, 3, 4 şi 5 cea mai bună acuratețe a fost obținută pentru un număr de 500 trăsături. Doar pentru gradul 2 acuratețea maximă (84.41%) este obținută pentru un număr de 1000 trăsături. Cel mai bun rezultat obținut este de 87.70% care s-a obținut pentru nucleul Gaussian cu parametru *C*=2.1 şi reprezentarea Cornell Smart a datelor de intrare. Valoarea maximă pentru nucleul polinomial a fost de doar 87.01% obținută pentru un grad al nucleului egal cu 2 şi reprezentarea Cornell Smart a datelor de intrare. Ca medie peste toate teste efectuate cea mai bună valoare o obține tot nucleul polinomial, ca şi la baza de date Reuters.

5 Rezultate referitoare la metodele de clasificare – clustering utilizând tehnici SVM

5.1 Rezultatele clasificării obținute pentru nucleul polinomial

În scopul îmbunătățirii acurateței clasificării utilizând nucleul polinomial idea mea a fost de a corela deplasamentul nucleului cu gradul acestuia şi a fost prezentată în secțiunea 3.6.1. În acest scop am efectuat teste pentru mai multe grade ale nucleului, considerând pentru fiecare dintre acestea 16 valori distincte ale deplasamentului, respectiv pentru fiecare reprezentare a datelor de intrare. Astfel pentru fiecare grad al nucleului am variat deplasamentul de la 0 la numărul total de trăsături (prezentând aici doar cele mai reprezentative rezultate obținute pentru 16 valori distincte).

În continuare prezint rezultatele obținute pentru o mulțime cu dimensiunea de 1309 trăsături obținute folosind metoda SVM_FS deoarece, după cum am arătat înainte, se obțin cele mai bune rezultate. Pentru această mulțime am variat deplasamentul între 0 şi 1309. De obicei în literatură, când deplasamentul nu este corelat cu gradul nucleului, este selectat cu o valoare între 0 şi numărul total de trăsături fără a se specifica exact de ce s-a ales acea valoare.

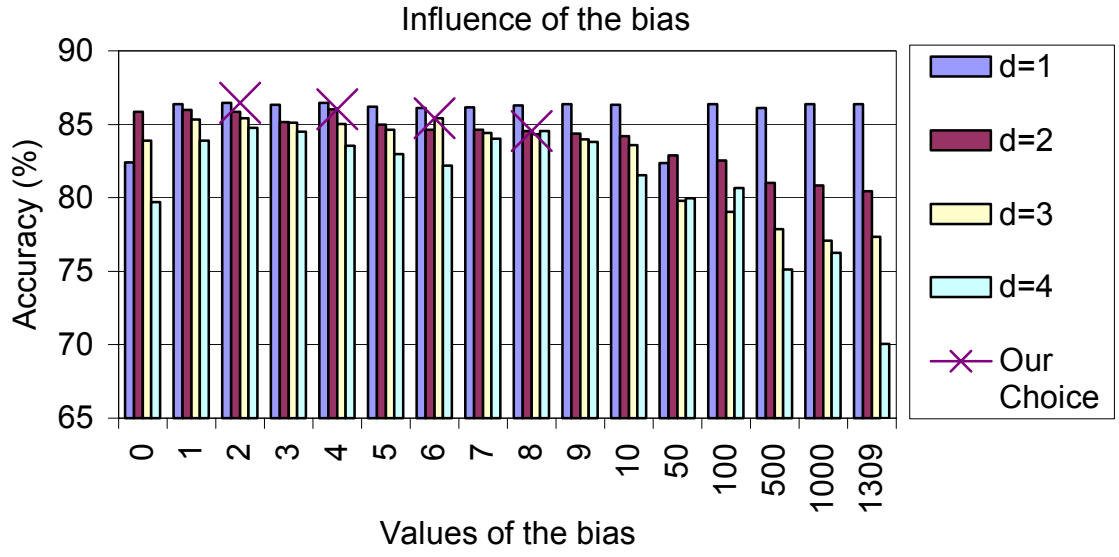

Figura 5.1 –Influența deplasamentului pentru reprezentarea nominală a datelor de intrare

În Figura 5.1 sunt prezentate rezultatele obținute pentru nucleul polinomial cu reprezentarea nominală a datelor de intrare şi diferite valori ale deplasamentului. În intrarea „Our choice" am marcat doar acele valori care sunt selectate automat folosind formula de corelare propusă (ecuația 3.13) pentru parametrii nucleului polinomial. După cum se poate observa folosind corelația, în cele mai multe cazuri obținem cele mai bune rezultate. În acest caz doar pentru gradul 4 cea mai bună valoare a fost obținută pentru deplasamentul egal cu 2 iar formula de corelare propusă de mine obține o valoare cu 0.21% mai mică decât cel mai bun rezultat obținut – 84.77%. În restul cazurilor, fără a fi nevoie să facem foarte multe teste în încercarea de a găsi un parametru optim pentru deplasament, utilizând formula propusă se obțin cele mai bune rezultate.

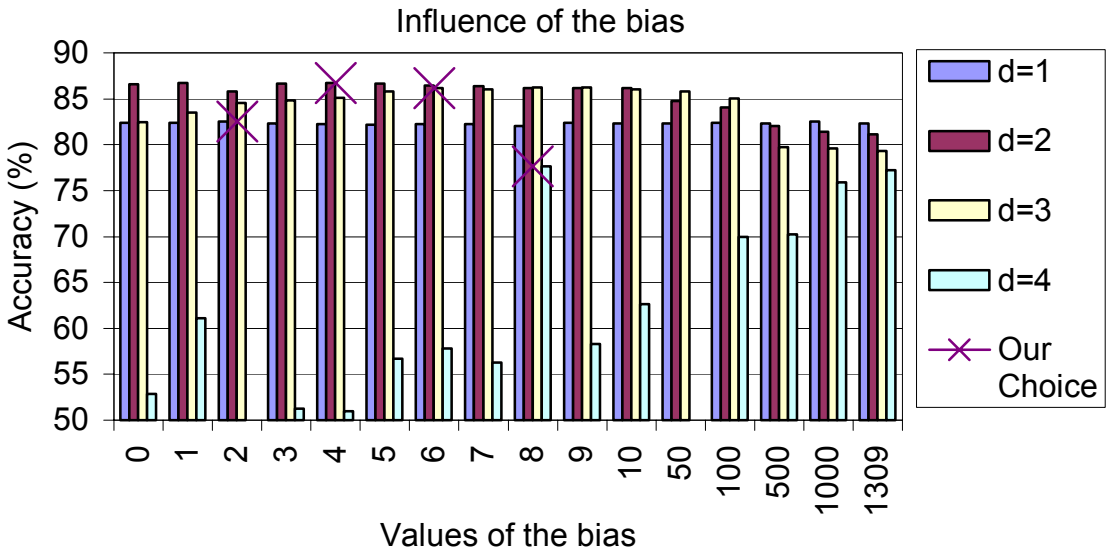

Figura 5.2 – Influența deplasamentului pentru reprezentarea binară a datelor de intrare

Bias	D=1	D=2	D=3	Our Choice
0	80.84	85.69	82.35	
1	81.84	86.64	83.37	
2	**82.22**	86.81	84.01	82.22
3	**82.22**	86.56	84.77	
4	**82.22**	**87.11**	65.12	87.11
5	82.01	86.47	85.54	
6	81.71	86.60	**86.51**	86.51
7	81.71	86.43	86.18	
8	82.09	86.43	86.47	
9	81.84	86.18	86.47	
10	81.80	85.96	86.26	
50	81.92	84.73	84.90	
100	**82.22**	83.71	82.82	
500	82.05	81.88	8.34	
1000	82.09	80.94	53.51	
1309	82.09	80.77	50.40	

Tabelul 5.1 Influența deplasamentului pentru reprezentarea CORNELL SMART

Valorile obținute pentru reprezentarea Cornell Smart pentru fiecare grad al nucleului şi pentru fiecare deplasament sunt prezentate în Tabelul 5.1, cu bold fiind reprezentate cele mai bune valori obținute pentru diferite valori ale gradului şi ale deplasamentului. Pentru fiecare grad există mai multe deplasamente pentru care se obțin cele mai bune rezultate dar formula propusă asigură că aceste valori se găsesc întotdeauna. De asemenea, o observație importantă este acea că pentru gradul egal cu 1 am obținut în aproape toate cazurile valori foarte apropiate ale acurateței clasificării, cea mai mare diferență fiind cu 0.51% față de valoarea maximă obținută. După cum se

poate observa din Tabelul 5.1, fără deplasament sau când deplasamentul este foarte mare se obțin rezultate nesatisfăcătoare pentru fiecare grad al nucleului testat. Cu reprezentarea Cornell Smart şi gradul nucleului egal cu 2 am obținut cea mai bună valoare de clasificare 87.11% iar folosind formula propusă de mine este de asemenea obținută această valoare. Aceleaşi teste au fost dezvoltate şi pentru reprezentarea binară şi în acest caz există mai multe valori pentru care se obțin cele mai bune rezultate iar utilizând formula de corelare aceste valori sunt obținute (Figura 5.2). Doar pentru gradul 3 cea mai mare valoare s-a obținut pentru deplasament egal cu 8, fiind cu 0.085% mai mare decât alegerea făcută de mine.

5.2 Rezultatele clasificării obținute pentru nucleul Gaussian

Pentru nucleul Gaussian am modificat parametru utilizat în mod constant *C* care reprezintă de obicei numărul de trăsături din vectorul de intrare, cu un produs între un număr mai mic (notat tot cu *C* în formula 3.14) şi respectiv un număr *n* care este calculat automat pentru fiecare pereche de vectori de intrare în parte. Voi prezenta aici teste cu patru valori distincte pentru parametrului *C*. Pentru fiecare dintre aceste valori am variat pe *n* de la 1 la 1309 (numărul total de trăsături din setul folosit în aceste teste). Deoarece valoarea propusă pentru *n* este calculată automat pentru fiecare pereche de vectori în parte, acest număr nu poate fi specificat din linia de comandă. De acea pentru fiecare valoare a parametrului *C* am prezentat în Figura 5.3 o intrare notată cu „auto" care înseamnă valoarea calculată automat utilizând formula propusă.

Am efectuat teste doar pentru reprezentarea binară şi Cornell Smart a datelor de intrare. Datorită parametrului *n* care apare în nucleul Gaussian şi care reprezintă numărul de elemente diferite de zero care apar în vectorii de intrare (parametrul *n* din ecuația 3.14) şi datorită reprezentării nominale a datelor de intrare între 0 şi 1 aceste valori devin foarte mici implicând rezultate foarte proaste ale acurateței clasificări. De acea nu voi prezenta aici rezultatele obținute cu reprezentarea nominală şi nucleul Gaussian.

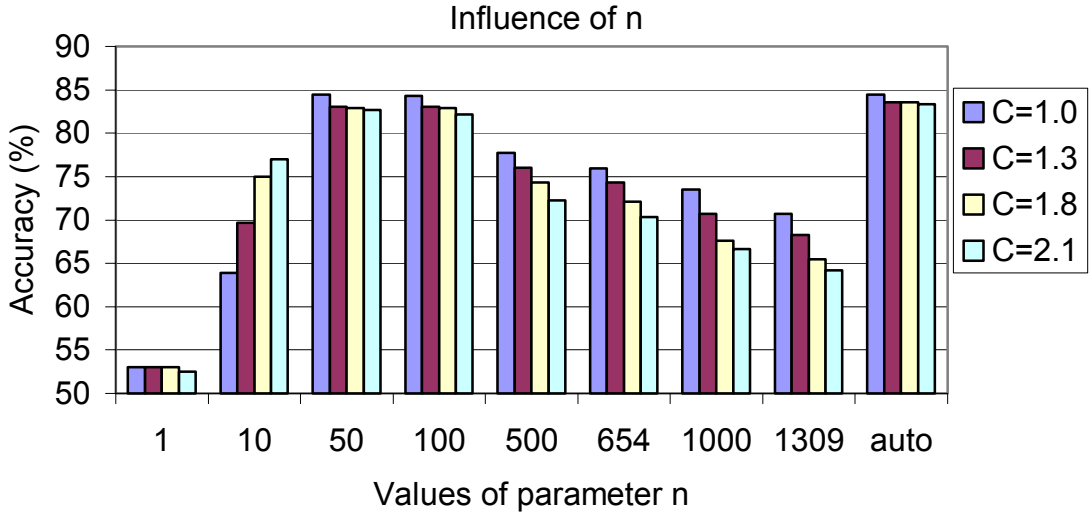

Figura 5.3 – Influența parametrului "*n*" din nucleul Gaussian şi reprezentarea binară a datelor

În Figura 5.3 sunt prezentate rezultatele obținute pentru reprezentarea binară a datelor de intrare. Când se utilizează corelația propusă se obțin cele mai bune rezultate. De asemenea rezultate bune se obțin când valoarea lui *n* este între 50 şi 100. Aceasta se întâmplă deoarece media numărului de trăsături care apar în fiecare document este între aceste valori. Când valoarea lui *n* este egală cu numărul total de trăsături (o valoare des utilizată în literatură) acuratețea descreşte, în medie pentru toate testele cu mai mult de 10% în comparație cu valorile obținute cu formula propusă pentru calcularea automată a parametrului *n*. De asemenea se poate observa că atunci când valoarea parametrului *n* creşte acuratețea clasificări are o descreştere substanțială. Aceleaşi teste au fost

făcute utilizând reprezentarea Cornell Smart şi s-a obţinut aceeaşi tendinţă iar acurateţea obţinută este cu 1% mai bună decât în cazul reprezentări binare (Figura 5.4).

În contrast cu nucleul polinomial, în cazul nucleului Gaussian am obţinut cele mai bune valori doar cu ajutorul formulei propuse pentru calculul parametrului "*n*".

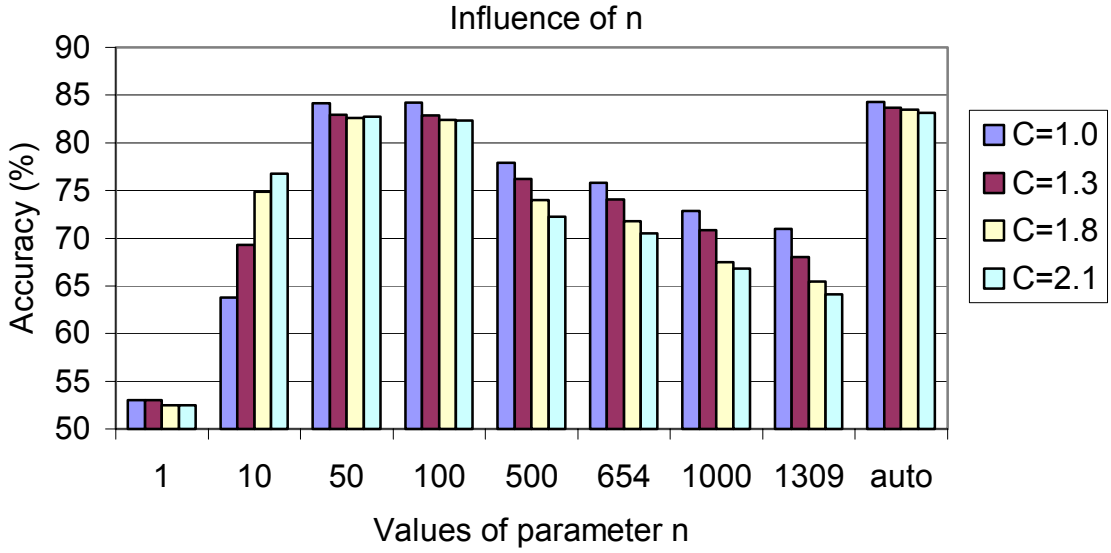

Figura 5.4 – Influenţa parametrului *n* pentru nucleul Gaussian şi reprezentarea Cornell Smart a datelor de intrare

5.3 Un standard „de facto": LibSVM

Majoritatea cercetătorilor prezintă rezultatele obţinute utilizând o implementare a tehnicii SVM numită LibSvm realizată de doi cercetători de la universitatea naţională din Taiwan şi disponibilă la [LibSvm]. LibSvm este o aplicaţie simplă, uşor de utilizat fiind un software eficient pentru clasificarea şi regresia SVM. În cea ce urmează voi prezenta rezultatele clasificării utilizând aplicaţia mea comparativ cu rezultatele clasificării utilizând LibSvm. LibSvm este o aplicaţie de linie de comandă şi permite utilizatorului să selecteze diferiţi algoritmi pentru SVM, diferite tipuri de nuclee şi diferiţi parametri pentru fiecare nucleu. LibSvm implementează patru tipuri de SVM (C-SVM, nu-SVM, one-class SVM, epsilon-SVR şi nu-SVR) şi are 4 tipuri de nuclee (liniar, polinomial, Gaussian şi sigmoid). Forma nucleelor din LibSvm pe care le voi utiliza este:

- Polinomial $k(x,x') = \left(gamma \cdot \langle x \cdot x' \rangle + coef0\right)^d$, unde *gamma*, *d* şi *coef0* sunt variabile;
- Radial basis function (RBF) $k(x,x') = \exp\left(-gamma * \|x - x'\|^2\right)$, unde *gamma* este singurul parametru variabil.

În momentul în care aceşti parametrii nu sunt specificaţi valoarea implicită pentru *gamma* acesta este 1/k, k fiind numărul total de trăsături, valoarea implicită pentru *d* este 3, şi valoarea implicită pentru *coef0* este 0.

Am utilizat pentru comparare dintre cele cinci implementări ale algoritmului SVM doar două "C-SVM" şi "one-class SVM" cu diferiţi parametri de intrare din LibSvm. Am realizat comparaţii cu C-SVM pentru clasificarea supervizată şi cu "one-class SVM" pentru cea nesupervizată.

5.4 LibSvm versus UseSvm

În această secţiune prezint influenţa corelării parametrilor nucleelor asupra acurateţei clasificării utilizând LibSvm. Am realizat o scurtă comparaţie între rezultatele obţinute utilizând LibSvm şi

implementarea mea numită UseSvm. LibSvm utilizează ca şi metodă pentru clasificarea în mai multe clase metoda „o clasă comparativ cu altă clasă”. Aplicaţia dezvoltată de mine utilizează metoda „o clasă comparativ cu restul claselor”. Baza de date Reuters, utilizată în aceste teste, conţine clase suprapuse şi în acest caz prima metodă obţine de obicei rezultate slabe. În baza de date Reuters aproape toate documentele aparţin mai multor categorii. Există categorii mai generale care conţin mai multe documente şi categorii mai specifice care conţin mai puţine documente. În cazul „o clasă comparativ cu restul claselor” toate documentele care sunt în acea categorie sunt comparate cu restul documentelor din mulţime. Astfel în acest caz întotdeauna în ambele clase vor exista documente diferite. În cazul „o clasă comparativ cu altă clasă” dacă una din clase este reprezentată de o categorie generală iar cealaltă de una specifică, documentele din clasa specifică vor fi şi în clasa generală făcând procesul de învăţare foarte dificil deoarece aceste clase pot fi în totalitate suprapuse.

Am utilizat doar o singură mulţime pentru aceste teste, aceea de dimensiune 1309 trăsături obţinute utilizând SVM_FS ca şi metodă pentru selecţia trăsăturilor. Pentru a avea o comparaţie corectă între cele două aplicaţii am încercat să elimin, pe cât posibil, suprapunerile din datele Reuters pentru a lucra doar cu clase fără suprapuneri, formal obţinând:

$$\forall i, j = \overline{1,13}, \mathbf{C}_i \cap \mathbf{C}_j = \varnothing \; for \; each \; i \neq j \qquad (5.1)$$

Am ales pentru fiecare eşantion doar prima clasă care a fost propusă de Reuters. Am eliminat de asemenea clasele care sunt slab sau excesiv reprezentate obţinând doar 13 clase distincte. Mulţimea de exemple astfel obţinute a fost împărţită aleator în mulţimile de antrenament şi respectiv de test care vor fi folosite atât pentru LibSvm cât şi pentru UseSvm. Rezultatele obţinute cu LibSvm sunt mai slabe în comparaţie cu rezultatele obţinute cu UseSvm, deoarece, în ciuda eforturilor depuse, datele au rămas totuşi suprapuse. În următoarele figuri prezint rezultatele obţinute pentru nucleul polinomial şi cel Gaussian utilizând parametri echivalenţi pentru ambele aplicaţii. Cum LibSvm are mai mulţi parametri de intrare care trebuiesc specificaţi decât UseSvm, pentru parametrii care apar doar la LibSvm am lăsat valorile implicite.

Aşa cum am specificat deja, pentru nucleul polinomial sugestia mea a fost “b=2*d” (vezi secţiunea 3.6.1). Prezint rezultatele obţinute utilizând LibSvm cu b=0 (valoarea implicită propusă de LibSvm) respectiv cu b=2*d (specificat explicit prin linia de comandă şi notat în grafice cu LibSvm+”b”) comparativ cu rezultatele obţinute cu UseSvm.

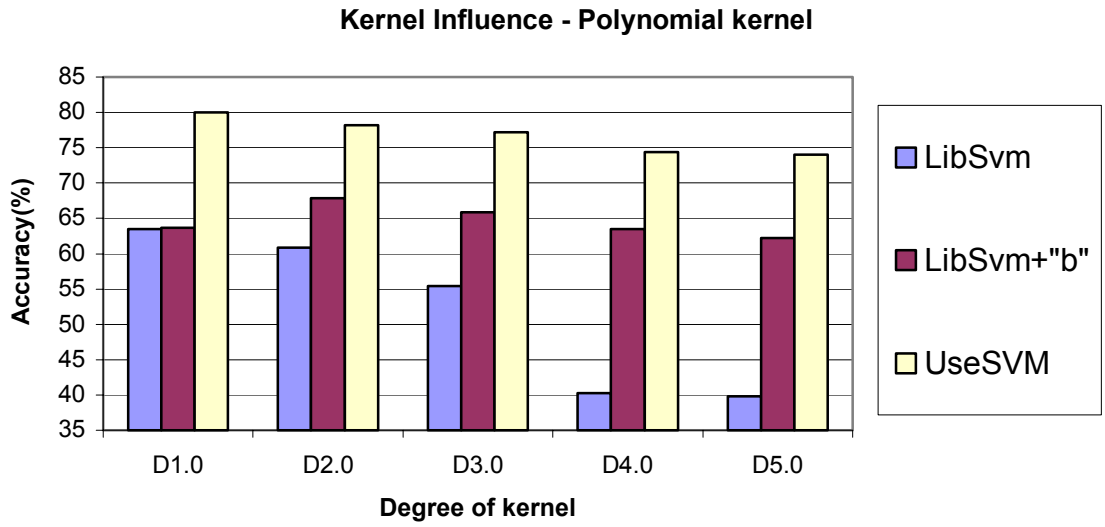

Figura 5.5 – Influenţa corelaţiei dintre parametri pentru nucleul polinomial

După cum se poate observa din Figura 5.5, programul UseSvm obține de departe cele mai bune rezultate comparativ cu bine cunoscutul LibSvm (cu o medie a câştigului cu 18.82% mai mare). Prin compararea LibSvm, având parametru implicit pentru deplasament, cu LibSvm în care specificăm clar parametrul se observă o îmbunătățire în medie cu 24.26% pentru toate gradele nucleului testate. Media câştigului este calculată ca şi media obținută de LibSvm cu parametrul implicit pentru deplasament împărțită la media obținută de LibSvm cu deplasamentul modificat utilizând formula propusă pentru corelare. Pentru gradul 1 s-au obținut rezultate comparabile deoarece valoarea implicită a deplasamentului şi valoarea calculată utilizând formula mea sunt apropiate.

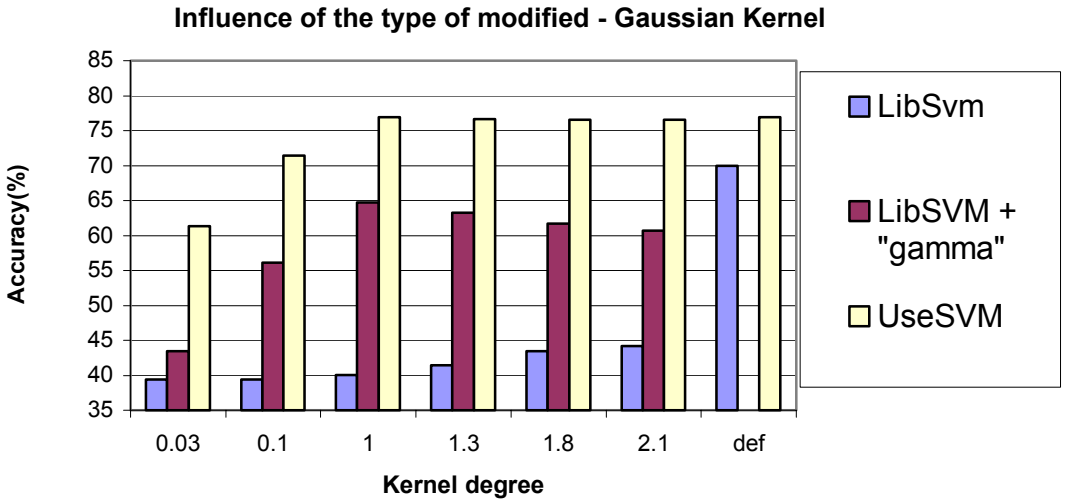

Figura 5.6 – Influența modificării nucleului Gaussian asupra clasificării

Simulările efectuate pentru nucleul Gaussian sunt prezentate în Figura 5.6. Sugestia mea a fost de a multiplica parametrul *C* cu un parametru *n* (aşa cum deja am explicat în secțiunea 3.6.2). Este dificil de specificat pentru LibSvma acest parametru din linia de comandă deoarece *n* este calculat automat pentru fiecare pereche de vectori în parte, iar LibSvm are doar un singur parametru pentru acest tip de nucleu care poate fi modificat, numit *gamma*. Mai precis, LibSvm utilizează $gamma = 1/n$ doar atunci când *gamma* este lăsat implicit (*n* însemnând media numărului de atribute ale datelor de intrare). Pentru LibSvm am utilizat gamma egal cu $1/C$. Pentru LibSvm+"gamma" am considerat "*gamma*" ca fiind egal cu $2/nC$, unde *n* este numărul total de trăsături împărțit la 2. Singurul caz al echivalențelor între aceste două programe (LibSvm şi UseSvm) este obținut pentru valoarea implicită pentru *gamma* în LibSvm şi respectiv pentru *C*=1 în UseSvm. Acest caz este prezentat separat ca „def" în Figura 5.6.

După cum se poate observa, utilizând idea mea de corelare pentru nucleul Gaussian rezultatele obținute utilizând LibSvm sunt mai bune în comparație cu rezultatele obținute utilizând LibSvm cu nucleul standard (cu un câştig mediu de 28.44%). Programul UseSvm obține de departe rezultate mai bune decât programul LibSvm, des utilizat în literatură (cu un câştig mediu de 25.57%). Pentru parametrul implicit la LibSvm aplicația mea a obținut de asemenea rezultate mai bune, obținând o valoare de 76.88% în comparație cu 69.97% obținut de LibSvm.

5.5 Clasificarea în mai multe clase – aspecte cantitative

Pentru un anumit număr de trăsături am testat clasificarea în mai multe clase utilizând toate cele 68 trăsături care au fost obținute inițial şi în toate cazurile am obținut o acuratețe a clasificării de

99.98% (având şi cazuri în care doar un singur eşantion este clasificat incorect). Aceasta anomalie poate apărea dacă există foarte multe elemente într-o clasă (de exemplu în cazul nostru clasa „ccat" care apare în 7074 exemple din toate cele 7083 exemple luate în considerare). În acest caz funcţia de decizie pentru această clasă va întoarce întotdeauna cea mai mare valoare şi celelalte funcţii de decizie nu au nici un efect pentru clasificarea în mai multe clase. În acest caz este uşor de exemplu să utilizăm un clasificator trivial, care prezice tot timpul aceeaşi clasă şi care va avea o acurateţe de 99.88%.

În faza de antrenare pentru fiecare clasă este învăţată câte o funcţie de decizie. În faza de evaluare fiecare eşantion este testat cu fiecare funcţie de decizie în parte şi este clasificat în clasa pentru care se obţine valoarea absolută cea mai mare. Dacă se obţin mai multe valori identice eşantionul este clasificat în toate clasele respective. Rezultatele obţinute sunt comparate cu clasificările propuse de Reuters. Reuters de obicei clasifică documentele în mai multe clase. Dacă rezultatele obţinute de noi aparţin setului de clase desemnate de Reuters considerăm că avem o clasificare corectă. Voi prezenta de asemenea rezultatele pentru nucleul polinomial şi cel Gaussian şi pentru toate cele trei tipuri de reprezentare a datelor de intrare.

In idea de a găsi ce mai bună combinaţie între tipul nucleului, gradul acestuia şi reprezentarea datelor de intrare am realizat 5 teste pentru nucleul polinomial cu gradul nucleului variind între 1 şi 5 şi respectiv 5 teste pentru nucleul Gaussian cu valorile pentru parametrul *C* (1.0, 1.3, 1.8, 2.1 şi 2.8). În [Mor05_2] sunt prezentate rezultate suplimentare. Pentru aceste experimente prezint rezultate obţinute pe o mulţime de 1309 trăsături selectate utilizând metoda SVM_FS. Voi prezenta rezultatele pentru nucleul polinomial doar pentru reprezentarea nominală iar pentru nucleul Gaussian doar pentru reprezentarea binară.

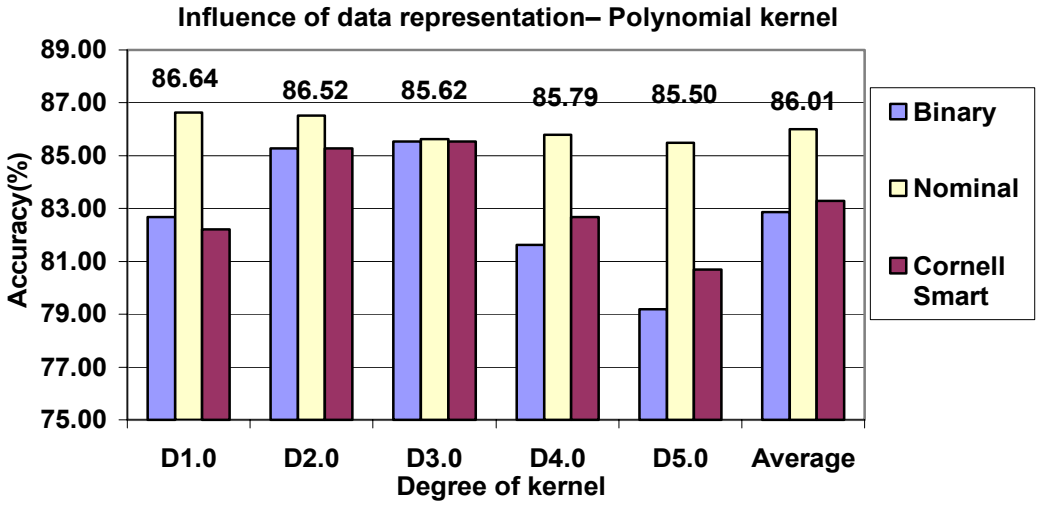

Figura 5.7 – Influenţa gradului pentru nucleu şi a tipului de reprezentare a datelor

În Figura 5.7 se observă că în general fişierele text sunt liniar separabile în spaţiul de intrare (dacă spaţiul de intrare are o dimensiune corect aleasă) şi rezultatele cele mai bune au fost obţinute pentru valori mici ale gradului nucleului (1 sau 2). Luând în considerare tipul de reprezentare al datelor de intrare pentru toate cele 5 teste, cele mai bune rezultate au fost obţinute cu reprezentarea nominală care obţine în medie 86.01% acurateţe în comparaţie cu reprezentarea binară care obţine 82.86% sau cea Cornell Smart care obţine 83.28%.

În general pentru nucleul polinomial timpul de antrenare este mai mic de o oră indiferent de metodele utilizate pentru selecţia trăsăturilor. Pentru metoda SVM_FS, gradul nucleului egal cu 2 şi reprezentarea nominală a datelor de intrare timpul de antrenare este de 14.56 minute pentru 1309 trăsături. Timpul prezentat este obţinut pe un calculator Pentium IV cu un procesor la 3.2GHz, 1Gb memorie cu WinXP. Mai multe rezultate referitoare la timp sunt prezentate în Figura 4.3.

În Figura 5.8 prezint rezultatele obținute pentru nucleul Gaussian pentru două tipuri de reprezentare a datelor de intrare şi pentru 5 valori diferite ale parametrului *C,* utilizând o mulțime cu 1309 trăsături, obținută utilizând metoda GA_FS. Am ales această metodă deoarece obține cele mai bune rezultate pentru nucleul Gaussian (după cum am prezentat în secțiunea 4.6.3).

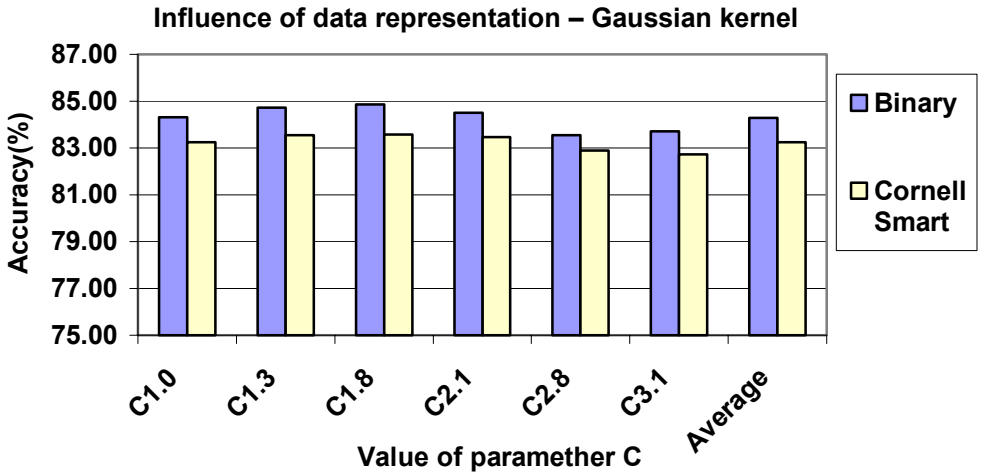

Figura 5.8 – Influența reprezentări datelor de intrare şi a parametrului *C* din nucleul Gaussian

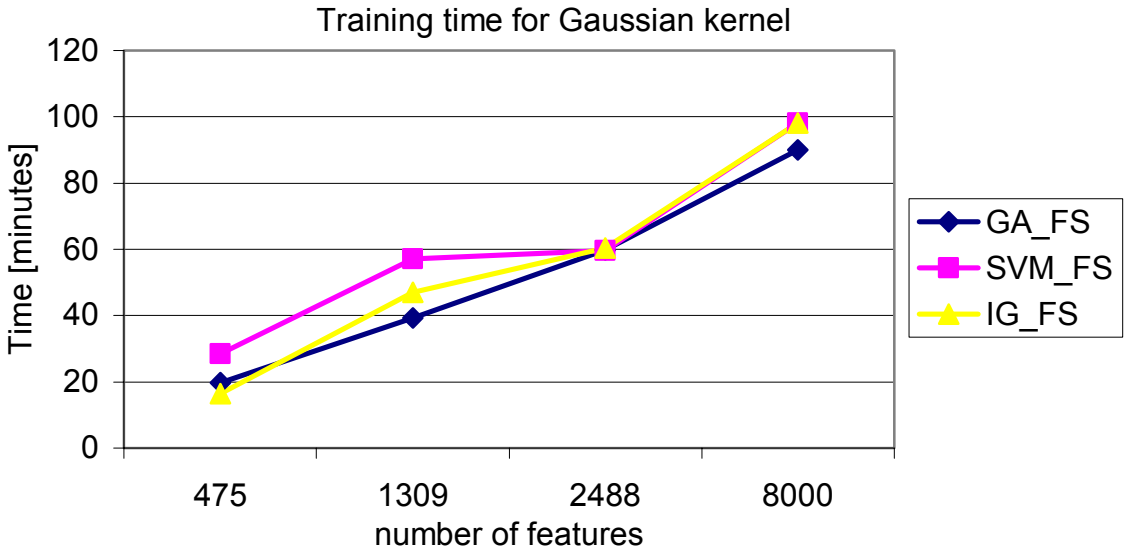

Figura 5.9 – Timpul de antrenare pentru nucleul Gaussian cu *C*=1.3 şi reprezentarea binară

Timpul de antrenare pentru nucleul Gaussian este prezentat pentru parametrul C=1.3 şi reprezentarea binară a datelor de intrare (Figura 5.9). Am mai prezentat aici şi timpul obținut cu seturile de date generate utilizând metodele IG şi SVM_FS pentru selecția trăsăturilor prezentate în secțiunea 4. Pentru acest tip de nucleu metoda SVM_FS necesită în toate cazurile de antrenare testate mai mult timp decât metoda GA_FS chiar dacă nu obține cele mai bune rezultate. De asemenea metoda IG obține pentru o dimensiune mică cel mai bun timp de antrenare, când dimensiunea creşte, creşte de asemenea şi timpul de antrenare mai mult decât timpul necesar folosind metoda GA_FS.

6 Proiectarea unor metaclasificatoare folosind SVM

Metaclasificarea sau clasificarea hibridă se bazează pe predicția clasificatorului (algoritmului) corect pentru o problemă particulară pe baza caracteristicilor vectorului de intrare şi a istoriei clasificărilor. Una dintre problemele principale care apare când sunt utilizați în practică algoritmi de clasificare este de a determina dacă clasificatorul obținut este fezabil şi pentru noi instanțe. Abordarea metaclasificării este una dintre cele mai simple abordări ale acestei probleme. Având mai mulți clasificatori de bază, ideea este de a învăța un metaclasificator care prezice gradul de corectitudine pentru fiecare dintre clasificatorii de bază. Metaetichetarea unei instanțe indică încrederea în clasificarea făcută de acesta, dacă instanța este clasificată corect de către acel clasificator dintre toți ceilalți clasificatori utilizați. Regula de clasificare a metaclasificatorului este ca fiecare clasificator de bază să atribuie o clasă la instanța curentă şi apoi metaclasificatorul să decidă dacă clasificarea este demnă de încredere sau nu. Un avantaj al metaclsificării este posibilitatea de exploatare a paraleilismelor funcționale (multiprocesor).

6.1 Selectarea clasificatorilor de bază

Metaclasificatorul meu conține mai mulți clasificatori SVM. În cercetările anterioare am arătat că anumite documente sunt clasificate corect doar de anumite tipuri de clasificatori SVM şi în anumite contexte. Astfel, am pus împreună mai mulți clasificatori bazați pe SVM cu diferiți parametri de intrare cu scopul de a îmbunătății acuratețea finală a clasificării. Strategia abordată de mine pentru dezvoltarea metaclasificatorului se bazează pe ideea selectării clasificatorului adecvat pentru documentele dificile. Clasificatorii selectați diferă prin tipul de nucleu folosit, prin valoarea parametrilor nucleului şi prin tipul de reprezentare a datelor de intrare. După analiza rezultatelor de test pentru fiecare clasificator studiat utilizând aceleaşi date de antrenament şi de test am selectat 8 clasificatori diferiți (Tabelul 6.1) ca şi componente pentru dezvoltarea metaclasificatorului.

NR. CRT.	TIP NUCLEU	PAREMETRUL NUCLEULUI	REPREZENTAREA DATELOR DE INTRARE	ACURATEȚEA OBȚINUTĂ(%)
1	Polinomial	1	Nominal	86.69
2	Polinomial	2	Binary	86.64
3	Polinomial	2	Cornell Smart	87.11
4	Polinomial	3	Cornell Smart	86.51
5	Gaussian	1.8	Cornell Smart	84.30
6	Gaussian	2.1	Cornell Smart	83.83
7	Gaussian	2.8	Cornell Smart	83.66
8	Gaussian	3.0	Cornell Smart	83.41

Tabelul 6.1 – Clasificatorii selectați

6.2 Limita de performanță a metaclasificatoarelor dezvoltate

O întrebare importantă care a apărut după selectarea clasificatorilor este: Care este limita superioară a metaclasificatorului? Cu alte cuvinte, vreau să ştiu dacă sunt anumite documente de intrare pentru care toți clasificatorii selectați atribuie o clasă incorectă. Oricum, clasificatorii au fost selectați astfel încât aceştia să aibă un număr mic de documente incorect clasificate.

Reamintesc faptul că şi aici voi lua în considerare ca referință în evaluarea rezultatelor obținute clasificarea propusă de Reuters.

Pentru a calcula limita metaclasificatorului am considerat toți clasificatorii selectați şi am numărat documentele care sunt incorect clasificate de către toți clasificatorii. Documentele verificate sunt cele din mulțimea de test deoarece sunt interesat dacă în acea mulțime se găsesc documente cu probleme. Am găsit un număr de 136 documente dintre cele 2351 din mulțimea de test care sunt clasificate incorect de către toți clasificatorii de bază. Astfel limita maximă a metaclasificatorului este de 94.21%.

6.3 Modelarea metaclasificatoarelor

Principalul obiectiv pe care l-am avut când am proiectat metaclasificatoarele a fost acela că acestea trebuie să poată furniza rapid răspunsul. De asemenea , în implementarea metaclasificatoarelor am fost interesat de idea de a selecta clasificatorul adecvat pentru vectorul de intrare dat. Pentru proiectarea metaclasificatoarelor am utilizat trei modele. Primul este un model simplu bazat pe votul majoritar, de altfel fără nici o adaptare la datele de intrare. Celelalte două modele se bazează pe o metodă adaptivă.

6.3.1 O metodă neadaptivă: Votul majoritar

Primul model al metaclasificatorului a fost proiectat datorită simplității lui. Este o metodă ne-adaptivă care obține același rezultat la fiecare rulare. Idea este de a utiliza toți clasificatorii selectați pentru a clasifica documentul curent. Fiecare clasificator votează o anumită clasă pentru documentul curent. Metaclasificatorul va păstra pentru fiecare clasă votată un contor, incrementând contorul clasei când un clasificator votează pentru ea. Metaclasificatorul va selecta clasa cu cel mai mare contor. Dacă se obțin două sau mai multe clase cu aceeaşi valoare a contorului voi clasifica documentul curent în toate clasele propuse de către metaclasificator. Marele dezavantaj al acestui metaclasificator este că nu îşi modifică evoluția o dată cu datele de intrare în scopul îmbunătății acurateței clasificării. Procentul de documente corect clasificate cu acest model este de 86.38%. Acest rezultat este cu 0.73% mai mic decât cea mai mare valoare obținută de către unul dintre clasificatorii selectați, dar este mai mare decât media clasificărilor făcute de fiecare clasificator selectat (85.26%). De asemenea un alt dezavantaj este faptul că acest metaclasificator necesită un timp destul de mare pentru furnizarea rezultatului.

6.3.2 Metodele adaptive dezvoltate

6.3.2.1 Selecția pe baza distanței euclidiene (SBED)

Deoarece metoda prezentată anterior nu obține rezultate satisfăcătoare am dezvoltat un metaclasificator care îşi modifică comportamentul în funcție de datele de intrare. Pentru a face aceasta, am realizat un metaclasificator care selectează clasificatorul care va fi folosit în funcție de eşantionul curent de intrare. Astfel, am făcut ca metaclasificatorul să învețe datele de intrare. Metaclasificatorul va învăța doar eşantioanele incorect clasificate deoarece numărul de eşantioane corect clasificate este mai mare decât numărul de eşantioane incorect clasificate. Fiecare clasificator va învăța eşantioanele care sunt incorect clasificate de către el. Metaclasificatorul va conține pentru fiecare clasificator o coadă proprie în care se vor memora toate documentele incorect clasificate de acel clasificator. Astfel metaclasificatorul va conține 8 cozi ataşate celor 8 clasificatori componenți.

6.3.2.1.1 Selectarea primului clasificator pe baza distanței euclidiene (FC-SBED)

Considerăm un document de intrare (eşantion curent) care trebuie să fie clasificat, alegând aleator un clasificator din cei 8 disponibili. Calculăm distanța euclidiană (ecuația 6.1) dintre eşantionul

curent şi toate eşantioanele care se găsesc în coada clasificatorului selectat. Dacă obținem cel puțin o distanță mai mică decât un prag prestabilit atunci vom renunța la clasificatorul selectat şi vom selecta un alt clasificator dintre cei rămaşi. Dacă nu, îl vom folosi pentru clasificarea eşantionului curent. În cazul în care toți clasificatorii sunt eliminați îl vom alege pe acela pentru care am obținut distanța euclidiană cea mai mare.

$$Eucl(\mathbf{x},\mathbf{x}') = \sqrt{\sum_{i=1}^{n}([x]_i - [x']_i)^2} \qquad (6.1)$$

unde $[x]_i$ reprezintă valoarea pentru intrarea *i* a vectorului **x**, iar **x** şi **x'** reprezintă vectorii de intrare.

După selectarea clasificatorului acesta va fi utilizat pentru a clasifica eşantionul curent. Dacă clasificatorul selectat reuşeşte să clasifice corect eşantionul curent nu se acționează asupra metaclasificatorului. În caz contrar eşantionul curent este pus în coada de documente a clasificatorului selectat. Se face aceasta deoarece doresc să evit ca acest clasificator să fie folosit ulterior pentru a clasifica documente asemănătoare cu acest document.

Acest proces are doi paşi. Toate acțiunile prezentate până acum se realizează în primul pas numit şi pasul de învățare. În acest pas metaclasificatorul analizează setul de antrenament şi de fiecare dată când un document este clasificat prost este pus în coada clasificatorului selectat. În cel de-al doilea pas, numit pasul de testare, se verifică procesul de clasificare. În acest pas caracteristicile metaclasificatorului rămân neschimbate. Deoarece după fiecare pas de antrenare caracteristicile metaclasificatorului vor fi schimbate, am repetat aceşti doi paşi de mai multe ori.

6.3.2.1.2 Selecția celui mai bun clasificator pe baza distanței euclidiene (BC-SBED)

Această metodă este identică cu metoda prezentată mai sus FC-SBEC cu o singură modificare. În faza de selecție a clasificatorului de bază acesta nu este ales aleator ci se aleg pe rând toți clasificatorii de bază şi se calculează distanța euclidiană între eşantionul curent şi toate eşantioanele aflate în cozi. Se va alege clasificatorul care obține distanța cea mai mare. În comparație cu metoda anterioară această metodă este mai lentă.

6.3.2.2 Selecția pe baza cosinusului (SBCOS)

Cosinusul este o altă posibilitate de a calcula similaritatea între documente. Această metrică este des utilizată în literatură când se lucrează cu vectori care caracterizează documentele şi se bazează pe calcularea produsului scalar dintre doi vectori. Formula utilizată pentru calculul cosinusului unghiului θ dintre doi vectori de intrare **x** şi **x'** este:

$$\cos\theta = \frac{\langle \mathbf{x},\mathbf{x}' \rangle}{\|\mathbf{x}\| \cdot \|\mathbf{x}'\|} = \frac{\sum_{i=1}^{n}[x]_i[x']_i}{\sqrt{\sum_{i=1}^{n}[x]_i^2} \cdot \sqrt{\sum_{i=1}^{n}[x']_i^2}} \qquad (6.2)$$

unde **x** şi **x'** sunt vectori de intrare (documentele) şi $[x]_i$ reprezintă componenta *i* a vectorului **x**.

Această metodă se aseamănă cu metoda SBED singura modificare apărând în calculul similarității între documente. De asemenea pentru această metodă de calcul a similarității între documente am implementat două metode diferite pentru selectarea clasificatorului care va fi utilizat la fel ca şi pentru SBED. Prima este selecția primului clasificator găsit pe baza cosinusului(FS-SBCOS) iar a

doua este selecția celui mai bun clasificator pe baza cosinusului (BC-SBCOS). În această metodă consider că clasificatorul curent selectat este acceptat dacă toate cosinusurile calculate între eşantionul curent şi toate eşantioanele care se găsesc în coadă sunt mai mici decât un prag prestabilit. Clasificatorul va fi respins dacă cel puțin un cosinus este mai mare decât pragul.

6.3.2.3 Selecția pe baza cosinusului şi a mediei

În toate metodele prezentate până acum în coada clasificatorului se păstrează pentru fiecare clasificator vectorul documentului care a fost incorect clasificat. Ca o alternativă a acestei metode am încercat să reduc dimensiunea cozii prin păstrarea doar a unui singur vector care reprezintă media pentru toți vectorii care ar trebui păstrați în coadă. Aceasta face algoritmul mult mai rapid dar din păcate duce şi la obținerea de rezultate mult mai slabe.

6.4 Evaluarea metaclasificatorilor propuşi

După cum am menționat deja cele două metode propuse SBED şi SBCOS necesită anumiți paşi de antrenare. Am să prezint aici rezultatele obținute pentru primii 14 paşi cu diferite valori ale pragului. M-am oprit după 14 paşi deoarece am observat că după acest prag acuratețea clasificării nu mai creşte, uneori chiar descreşte puțin.

Când utilizăm selecția pe baza distanței euclidiene pragul a fost ales pentru primii 7 paşi egal cu 2.5 şi pentru ultimii 7 paşi egal cu 1.5. Prima dată am selectat un prag mai mare cu scopul de a elimina mult mai mulți clasificatori. Când în coadă există deja eşantioane voi renunța mult mai uşor la acel clasificator. Astfel în primii paşi sunt interesat în popularea tuturor cozilor din metaclasificator. În ultimi 7 paşi am scăzut pragul pentru a face eliminarea clasificatorului mai dificilă. Aceste două valori ale pragului au fost alese după mai multe experimente cu diferite praguri care nu sunt prezentate aici.

În cazul selecției pe baza cosinusului pragul a fost ales pe durata primilor 7 paşi ca fiind egal cu 0.8 şi pe durate ultimilor 7 paşi ca fiind egal cu 0.9. De asemenea am fost interesat să elimin uşor un clasificator în primii 7 paşi. În ultimi 7 paşi am utilizat o valoare mai apropiată de 1, care înseamnă documente similare. Aceasta asigură că în primii paşi populez mult mai uşor cozile iar în ultimii paşi calibrez metaclasificatorul pentru documentele care sunt mult mai dificil de clasificat. Am făcut de asemenea experimente cu valori ale pragului mai mici de 0.8 dar aici voi prezenta doar cele mai bune rezultate obținute.

În comparație cu SBED, SBCOS are un punct de start mai bun (85.33%). După 14 paşi acuratețea a crescut doar la valoarea de 89.75%. Considerând raportul între valoarea maximă obținută şi limita superioară a metaclasificatorului putem observa că acest metaclasificator poate atinge valoarea de 95.26%. Folosind media eşantioanelor memorate în cozi acuratețea clasificării nu creşte aşa de mult. Acuratețea creşte de la 86.38% doar până la 86.77% în ultimul pas. Acest maxim este mai mare decât valoarea obținută cu metoda bazată pe votul majoritar cu doar 0.39%. De asemenea diferența dintre FC-SBCOS şi BC-SBCOS nu este aşa de mare, de obicei se obțin aproximativ aceleaşi rezultate. În ultimul pas metoda BC-SBCOS obține un rezultat cu 1.06% mai mare decât cealaltă metodă.

În Figura 6.1 sunt prezentate rezultatele comparative între toate metodele utilizate pentru a construi metaclasificatorul. Pentru metoda SBED am ales să prezint rezultatele obținute cu FC-SBED iar pentru SBCOS am ales să prezint rezultatele pentru BC-SBCOS. Mai multe rezultate sunt prezentate în [Mor06_5].

Cu Votul Majoritar acuratețea clasificării este de doar 86.38%. Acest rezultat este cu 0.73% mai mic decât valoarea maximă obținută de cel mai bun clasificator individual. Comparând metoda SBED cu metoda SBCOS ce-a de-a doua metodă are un punct de pornire mai bun (85.33%). După 14 paşi acuratețea clasificării a crescut doar la valoarea de 89.75% în comparație cu SBED care

obține o acuratețe finală maximă de 92.04%, această valoare fiind cu 2.17% mai mică decât limita maximă care poate fi atinsă de acest metaclasificator.

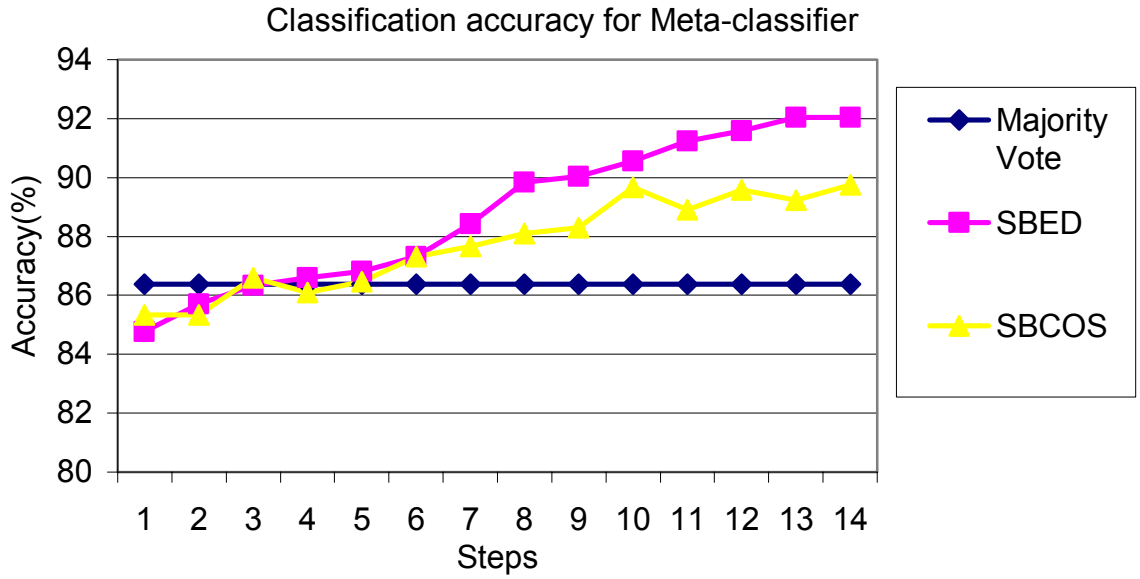

Figura 6.1 – Comparație între metodele de proiectare a metaclasificatorului

In Figura 6.2 sunt prezentați timpii de răspuns obținuți de metodele prezentate mai sus. Pentru Votul Majoritar am considerat de mai multe ori aceeaşi timp de răspuns pentru a putea avea o vizualizare comparativă mai bună. Pentru celelalte două metode am prezentat o medie între timpul obținut de metoda bazată pe selecția celui mai bun clasificator şi metoda bazată pe selecția primului clasificator bun găsit.

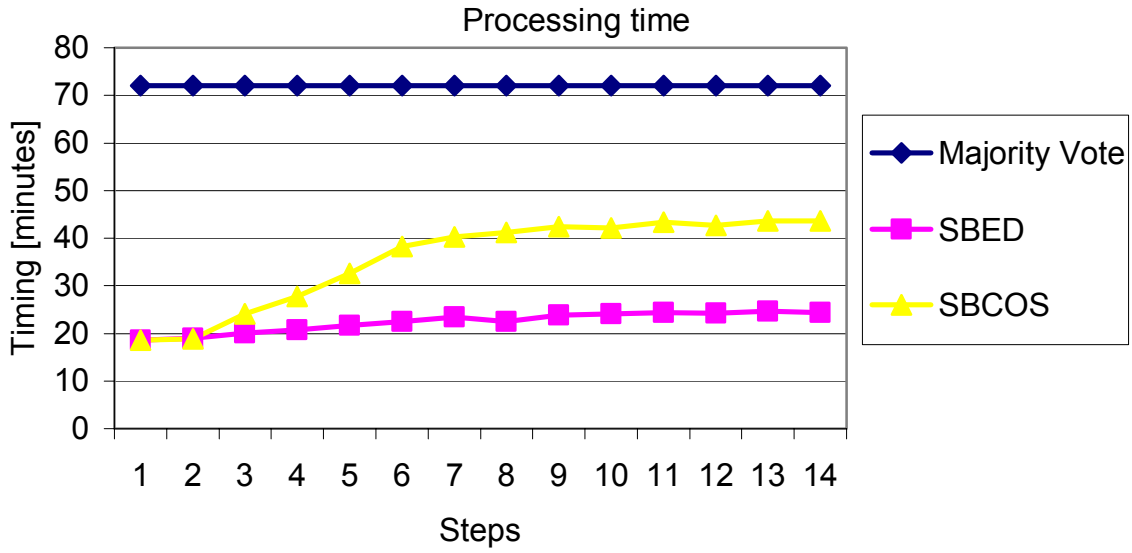

Figura 6.2 – Timpul de procesare comparație între SBED şi SBCOS

Ca o comparație putem vedea că cea mai rapidă metodă este acea care utilizează distanța euclidiana pentru calculul similarității. Uneori obținem o diferență de până la 20 de minute pe durata ultimului pas. Aceasta apare datorită dimensiunii cozilor care în cazul metodei bazate pe cosinus este mai mare. Diferența dintre aceste două metode poate apărea deoarece în cazul SBDE valoarea obținută nu este normalizată şi se poate specifica mai uşor un prag care să facă diferența între acceptarea sau neacceptarea clasificatorului. În cazul SBCOS valoarea obținută este între 0 şi 1 şi este mai dificil de a găsi un astfel de prag optim.

7 Cercetări asupra scalabilității metodelor de clasificare

În ultimi ani bazele de date text disponibile devin din ce în ce mai mari şi o multitudine de algoritmi au fost propuşi pentru a lucra cu ele. De asemenea, se încearcă transformarea algoritmilor existenți astfel încât să permită lucrul cu aceste baze de date mari (de exemplu scalabilitatea algoritmilor) într-o perioadă de timp care să nu crească exponențial. Scalabilitatea a fost întotdeauna o preocupare majoră pentru algoritmii de recunoaştere a informaților.

7.1 Metode pentru determinarea scalabilității algoritmilor de clasificare

În acest capitol doresc să analizez dacă există modificări majore ale acurateții finale a clasificării când algoritmul meu trebuie să lucreze cu baze de date mari din care se selectează doar anumiți vectori de intrare. Pentru a putea realiza aceasta am dezvoltat o strategie în trei etape care să ne permită să lucrăm cu o dimensiune mult mai mare a mulțimii de antrenament, reducând timpul de antrenare de care am fi avut nevoie daca am fi lucrat cu mulțimea completă la un moment dat. Ne concentrăm pe partea de antrenare unde cantitatea de date prezentate pentru antrenare are o mare influență asupra capacității sistemului de învățare. Pentru proiectarea acestei strategii m-am inspirat din strategia prezentată în [Yu03] care utilizează o structură arborescentă în care pe fiecare nivel sunt grupate datele similare din baza de date pe nivelul anterior. Am modificat această strategie pe care autorul nu a recomandat-o pentru a fi utilizată în cazul documentelor text în aşa fel încât să grupeze toate datele pe un singur nivel.

În funcție de metoda utilizată pentru calculul vectorului reprezentativ (vectorul care caracterizează un grup de documente considerate similare) am utilizat două tipuri de reprezentare ale acestuia. Primul dintre acestea conține suma peste toate elementele care sunt incluse în acel grup precum şi o valoare care reprezintă numărul total de elemente din acel grup pentru a putea calcula media aritmetică. În cea de-a doua reprezentare fiecare vector reprezentativ este calculat utilizând ecuația 7.1.

$$\vec{\mathbf{w}}_i(t+1) = \vec{\mathbf{w}}_i(t) + \alpha(\vec{\mathbf{x}} - \vec{\mathbf{w}}_i(t)) \tag{7.1}$$

unde **w** reprezintă vectorul reprezentativ pentru clasa *i*, x reprezintă vectorul de intrare şi α reprezintă rata de învățare.

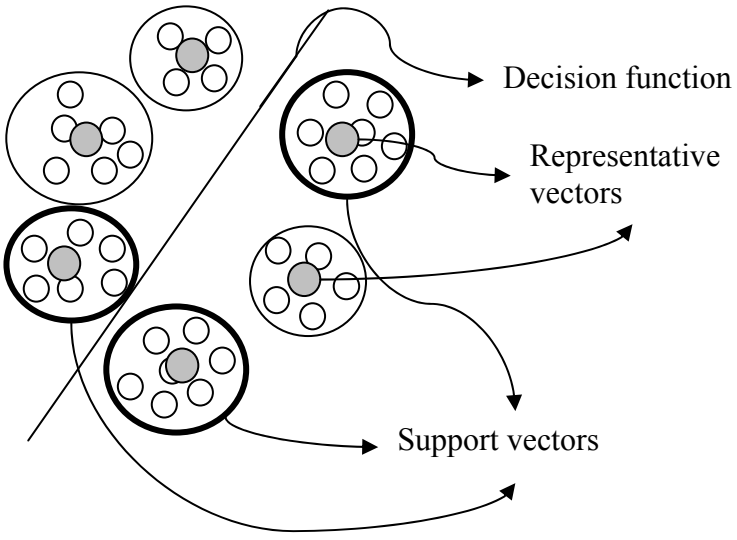

Figura 7.1 – Selectarea vectorilor suport

Întreg procesul este descris în continuare în 7 paşi:

1. Se normalizează vectorii de intrare astfel încât aceştia să devină de lungime 1 utilizând formula:

$$TF(d,t) = \frac{n(d,t)}{\sum_{\tau=0}^{19038} n(d,\tau)} \tag{7.2}$$

   unde $TF(d,t)$ este frecvența termenului, $n(d,t)$ reprezintă numărul de apariții ale termenului t în documentul d şi împărțitorul reprezintă suma tuturor termenilor care apar în documentul d.

2. După normalizare se calculează distanța euclidiană dintre vectorul de intrare şi fiecare vector reprezentativ (Figura 7.1 cercurile mici gri) care a fost creat până în acel moment. Aceasta face ca strategia mea să lucreze mai lent când avem multe grupuri create. Formula folosită pentru calculul distanței euclidiene este:

$$E(t,c_i) = \sqrt{\sum_{k=0}^{19038} (x_k - w_{ik})^2} \tag{7.3}$$

   unde x_k reprezintă termenii din vectorului de intrare şi w_{ik} reprezintă termenii vectorului reprezentativ al clasei i. Astfel, se calculează toate distanțele şi se păstrează cea mai mică. Dacă distanța obținută este mai mică decât un prag prestabilit se introduce eşantionul curent în grupul câştigător şi se recalculează vectorul reprezentativ pentru acel grup, dacă nu, se creează un nou grup pentru care vectorul reprezentativ este chiar vectorul de intrare.

3. După această grupare (toate cercurile mari din Figura 7.1), se creează o nouă mulțime de antrenament care conține toți vectorii reprezentativi. Această mulțime va fi utilizată în pasul de clasificare în care nu ne interesează acuratețea clasificării, deoarece vectorii din mulțime nu sunt vectori de intrare, ci doar vectorii suport. În pasul de clasificare utilizăm o metodă care necesită date etichetate, astfel trebuie să specificăm o etichetă pentru fiecare vector reprezentativ. Această etichetă este specificată automat ca fiind cea mai frecventă etichetă propusă de Reuters care apare în exemplele din acel grup.

4. Pe acest set redus în care fiecare vector are 19038 trăsături se realizează un pas de selecție a trăsăturilor relevante. Pentru acest pas am preferat să utilizez metoda SVM_FS. După calcularea tuturor ponderilor se selectează un număr de 1309 trăsături deoarece pentru această dimensiune a vectorului s-au obținut cele mai bune rezultate.

5. Vectorii rezultați, de dimensiune mai mică, sunt utilizați în pasul de învățare folosind algoritmul SVM. Pentru acest pas am utilizat nucleul polinomial cu gradul 1 şi reprezentarea nominală a datelor. Am utilizat nucleul polinomial deoarece de obicei obține un număr redus de vectori suport în comparație cu nucleul Gaussian. Am utilizat gradul 1 cu reprezentarea nominală deoarece nu se dorește transpunerea datelor într-un spațiu de dimensiune mai mare şi se obțin de obicei rezultate bune.

6. După pasul de învățare se aleg doar acei vectori care sunt vectori suport. În teoria SVM vectorii suport sunt acei vectori care au parametrul α (multiplicatorul Lagrange) mai mare decât 0. Aceştia sunt reprezentați în Figura 7.1 prin cercurile mari cu linie îngroşată. În continuare se aleg doar acele grupuri pentru care vectorii reprezentativi au fost selectați ca şi vectori suport. Cu elementele din aceste grupe se creează o nouă mulțime care conține vectori de intrare originali dar care are o dimensiune mult mai mică decât cea inițială.

7. Această mulțime va fi utilizată în pasul de clasificare în care suntem interesați de acuratețea clasificării obținute pe o dimensiune mult mai mică, dar relevantă, a datelor de intrare.

7.1.1 Gruparea datelor utilizând media aritmetică

În datele prezentate am pornit cu o mulțime de 7083 vectori de intrare. După pasul de grupare (pasul 2) am redus această dimensiune la 4474 vectori reprezentativi ceea ce înseamnă o reducere la 63% a setului inițial. Pentru calcularea vectorilor reprezentativi am folosit calculul mediei aritmetice a tuturor vectorilor din acea clasă. Pentru a obține această reducere am folosit un prag cu valoarea 0.2. În următorul pas am făcut un pas de selecție a trăsăturilor utilizând SVM_FS şi după acea un pas de clasificare pentru selectarea vectorilor relevanți. După clasificare am obținut un număr de 874 vectori suport. Luând aceşti vectori suport am creat o mulțime de date care conține doar 4256 elemente (vectori relevanți) care reprezintă aproximativ 60% din mulțimea inițială. Această mulțime a fost împărțit aleator în mulțimea de antrenament de 2555 eşantioane şi cea de test de 1701 eşantioane.

7.1.2 Gruparea datelor utilizând algoritmul LVQ

Pentru a putea face o comparație între aceste două metode am încercat să obțin aproximativ acelaşi număr de vectori, ca cei obținuți mai sus, în ambii paşi de selecție (pasul de grupare şi pasul de selectare a vectorilor suport) dar de data aceasta folosind algoritmul LVQ. Pentru primul pas am folosit o valoare a pragului egală cu 0.15 şi o rată de învățare egală cu 0.9 obținând un număr de 3487 grupuri. Am utilizat o valoare mare pentru rata de învățare deoarece de obicei grupurile create au un număr mic de elemente şi sunt interesat în crearea acestor grupuri doar într-un singur pas. Mă aştept ca această metodă să lucreze mult mai bine cu mulțimi de date foarte mari. După crearea grupurilor am continuat cu pasul de selecție a trăsăturilor reducând numărul acestora de la 19038 la 1309. În următorul pas, utilizând aceşti vectori mai mici am antrenat clasificatorul cu scopul de a selecta vectorii suport. După acest pas am selectat doar acei vectori care au multiplicatorul Lagranfge mai mare ca 0 (vectorii suport). Am luat aceşti vectori ca fiind cei mai relevanți vectori din cei reprezentativi. Am creat o mulțime care conține doar 4333 eşantioane. Acest număr reprezintă aproximativ 61% din mulțimea inițială. Pentru această nouă mulțime am selectat de asemenea 1309 trăsături. Mulțimea obținută a fost împărțită automat în mulțimea de antrenament de 2347 eşantioane şi mulțimea de test de 1959 eşantioane.

7.2 Scalabilitatea metodelor de clasificare – aspecte cantitative

Voi prezenta rezultate doar pentru o singură dimensiune a numărului de trăsături caracteristice – 1309- deoarece pentru această mulțime am obținut cele mai bune rezultate de până acum. De asemenea am fost interesat să cercetez dacă acuratețea clasificării scade excesiv când se aplică metoda de selectare a unei părți relevante din setul inițial comparativ cu acuratețea obținută pe întreg setul.

În Figura 7.2 am prezentat comparativ rezultatele obținute pentru nucleul polinomial şi reprezentarea nominală a datelor pentru toate cele trei mulțimi (mulțimea originală notată prin SVM-7053, mulțimea obținută utilizând media aritmetică pentru calculul vectorilor reprezentativi notată AM-4256 şi mulțimea obținută utilizând metoda LVQ pentru calculul vectorilor reprezentativi numită LVQ-4333).

După cum se poate observa există o mică diferență între rezultatele obținute cu AM-4256 comparativ cu cele obținute cu LVQ-4333. Diferența în acuratețea obținută între setul original şi AM-4256 este în medie cu 1.30% mai mică pentru toate gradele nucleului. Aceeaşi diferență este de asemenea obținută între setul original şi LVQ-4333. Când lucrăm cu un grad al nucleului mic, de exemplu 1, diferența între setul original şi AM-4256 este mai mică de 1% iar diferența dintre setul original şi LVQ-4333 este un pic mai mare. Când gradul nucleului creşte rezultatele sunt mai bune pentru setul LVQ-4333 comparativ cu setul AM-4256 dar de obicei diferența poate fi considerată nesemnificativă. De exemplu în medie pentru toate gradele nucleului şi toate tipurile

de reprezentare a datelor diferența dintre setul original şi setul AM-4256 este de 1.65% şi diferența între setul original şi setul LVQ-4333 este de 1.64%.

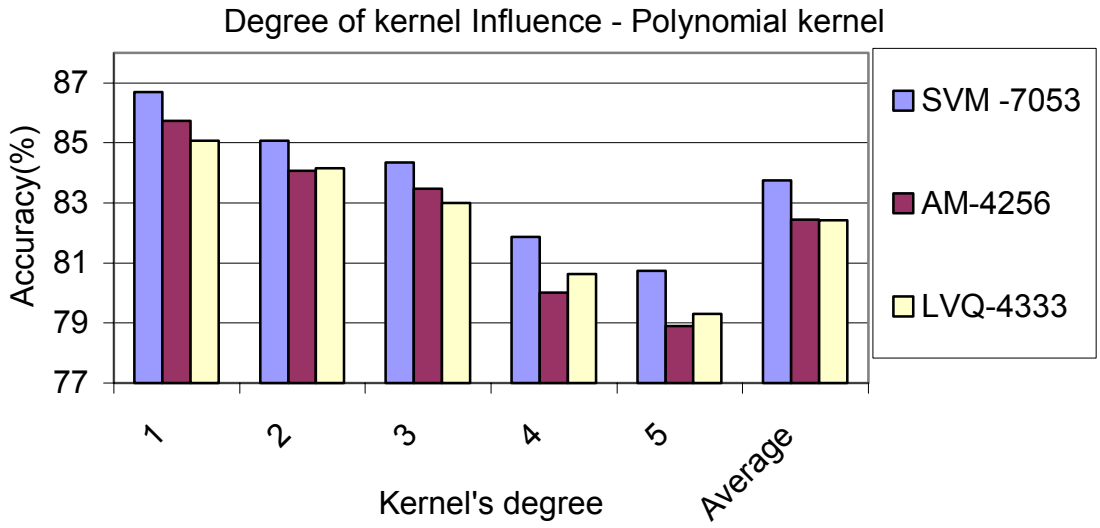

Figura 7.2 – Rezultate comparative pentru diferite dimensiuni ale mulțimii şi nucleul polinomial

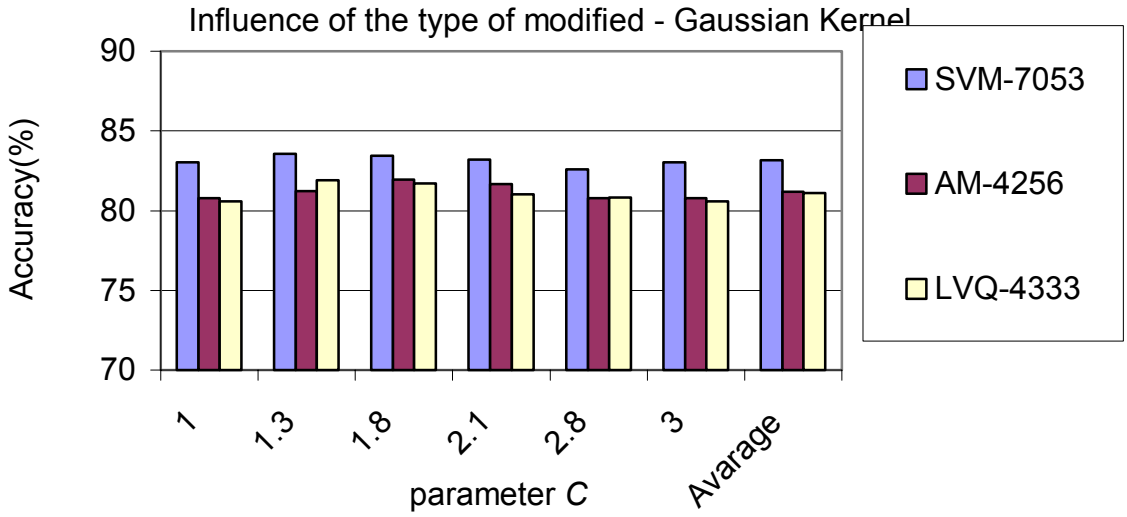

Figura 7.3 – Rezultate comparative pentru diferite dimensiuni şi nucleul Gaussian

În Figura 7.3 sunt prezentate rezultatele obținute pentru nucleul Gaussian şi reprezentarea Cornell Smart a datelor. Pentru acest nucleu diferența dintre aceste două mulțimi de date şi mulțimea originală este mai mare decât în cazul nucleului polinomial, fiind în medie de 1.89% pentru AM-4256 şi de 1.95% pentru LVQ-4333. Observăm de asemenea că pentru valorile pentru care se obțin cele mai bune rezultate ale clasificării obținem şi cele mai mici diferențe între setul original şi seturile reduse. Cea mai mică diferență a fost obținută pentru parametrul C =1.8, valoare pentru care în toate testele anterioare am obținut cele mai bune rezultate pentru nucleul Gaussian. Pentru acest tip de nucleu metoda bazată pe LVQ obține întotdeauna rezultate puțin mai slabe decât metoda bazată pe medie aritmetică.

Am redus datele în primul pas la 63% din datele inițiale iar în al doilea pas la 60% din datele inițiale, pentru prima metodă respectiv la 50% în primul pas şi la 61% în al doilea pas, pentru a doua metodă. Cu această reducere pierderea în acurateţe a fost în jur de 1% pentru nucleul polinomial şi în jur de 1.8% pentru nucleul Gaussian. Este interesată observația că pentru valori ale parametrilor pentru care obținem cea mai bună clasificare pe mulțimea completă se obține de asemenea şi cea mai mică diferență între setul normal şi cel redus. Oricum am făcut o reducere a

setului inițial cu aproximativ 40%, reducând substanțial şi timpii de antrenare, iar scăderea în acurateţe a clasificării a fost doar de 1-2%.

7.3 Rezultate aşteptate când se utilizează un set de date mai mare

Metoda de obținere a mulțimii mari de date care va fi utilizată în această secțiune a fost descrisă în secțiunea 2.4.2. Intenția mea nu este de a utiliza întreaga mulțime în procesul de clasificare. Prima dată încerc să creez grupuri cu scopul de a reduce dimensiunea urmărind paşii prezentați în acest capitol. Am utilizat metoda bazată pe LVQ cu un prag egal cu 0.15 şi o rată de învățare egală cu 0.4 pentru a crea grupele din primul pas. Am obținut un număr de 11258 vectori reprezentativi, deci având o reducere de 31% a mulțimii inițiale. Utilizând această mulțime de vectori reprezentativi am antrenat clasificatorul pentru a putea selecta vectorii suport. După antrenare am selectat acei vectori care au valoarea mai mare în modul decât un prag egal cu 0.2. Astfel în al doilea pas am obținut un număr de 10952 vectori de intrare care sunt împărțiți aleator în mulțimea de antrenament de 5982 vectori şi cea de test de 4970 vectori. În ceea ce urmează voi prezenta rezultatele obținute utilizând această mulțime redusă cu 33% față de cea inițială. Dacă scalabilitatea este păstrată ca cea pentru prima mulțime testată, când rezultatele sunt de obicei cu 1-2% mai mici dacă lucrăm cu mulțimea redusă comparativ cu mulțimea completă sper că rezultatele obținute vor fi cu 1-2% mai bune dacă voi lucra cu întreaga mulțime de date.

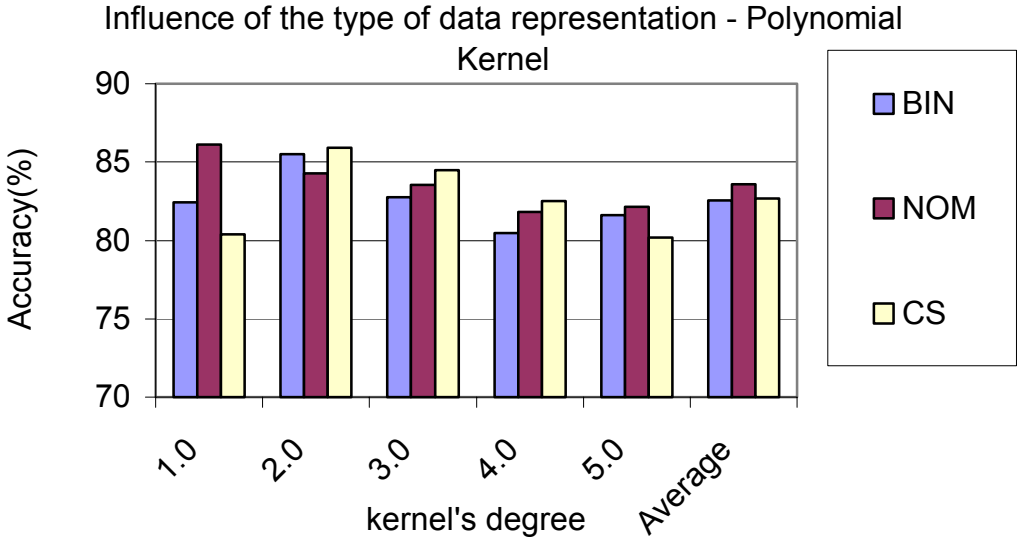

Figura 7.4 – Influența tipului de reprezentare a datelor pentru nucleul polinomial

Am rulat câteva teste pentru nucleul polinomial şi câteva pentru cel Gaussian pentru toate cele trei tipuri de reprezentare a datelor de intrare. În Figura 7.4 sunt prezentate rezultatele obținute pentru nucleul polinomial. Rezultatele sunt prezentate pentru o dimensiune a vectorului de 3000 trăsături relevante. Pentru selecția acestora am utilizat metoda SVM_FS. Chiar dacă am lucrat cu o dimensiune redusă rezultatele sunt semnificativ mai bune comparativ cu rezultatele obținute când am lucrat cu o mulțime completă de date de o dimensiune mai mică (7083 elemente). Aceasta se întrâmplă deoarece în această mulțime numărul de trăsături este mai mare iar acuratețea creşte când antrenăm pe mult mai multe date.

8 Concluzii

8.1 Principalele contribuții originale ale lucrării

Lucrarea prezintă munca autorului în domeniul clasificării de documente, în special în clasificarea documentelor text şi a celor web. Din punct de vedere al contribuției autorului la dezvoltarea acestui domeniu se pot remarca următoarele:

În **capitolul 1** autorul prezintă o imagine de ansamblu asupra acestei teze. Se prezintă procesul de clasificare automată a documentelor cu toate părțile componente şi se pune accentul pe părțile în care autorul şi-a adus contribuții.

În **capitolul 2** autorul realizează o sinteză a metodelor care s-au consacrat în domeniul extragerii de cunoştințe din baze de date, documente text şi documente web punând accentul pe documentele text şi web. De asemenea în acest capitol sunt prezentate cunoştințele de bază necesare acestui domeniu şi de asemenea metodele folosite de autor pentru pregătirea mulțimilor de antrenare şi testare care vor fi folosite în prezentarea rezultatelor.

În **capitolul 3** autorul prezintă bazele matematice ale algoritmului utilizat în procesul de clasificare de documente – Support Vector Machine. De asemenea, autorul prezintă aici metoda utilizată pentru implementarea algoritmului de clasificare bazat pe vectori suport – Seqential Minimal Optimization. Tot aici este prezentată o modalitate care permite algoritmului să lucreze şi cu exemple neetichetate, care în ultimul timp sunt tot mai numeroase. La sfârşitul acestui capitol autorul propune o metodă nouă pentru corelarea parametrilor nucleelor SVM care conduce la o îmbunătățire a acurateței clasificării şi simplifică modalitatea de selectare a acestor parametri. Autorul propune o metodă pentru corelarea gradului nucleului polinomial cu deplasamentul care trebuie adăugat acestuia, respectiv corelează parametrul care apare în nucleul Gaussian cu dimensiunea vectorilor de intrare folosiți în nucleu. Această dimensiune reprezintă numărul de trăsături distincte care apar în vectorii de intrare şi au ponderile diferite de zero.

În **capitolul 4** autorul prezintă patru metode pentru selectarea trăsăturilor relevante. Prima metodă prezentată se bazează pe selecția aleatoare (RAN) şi este folosită datorită simplității şi pentru a justifica necesitatea utilizării unor metode mult mai puternice. A doua metodă prezentată, Câştigul Informațional (IG), este o metodă utilizată în mod curent în literatura de specialitate şi a fost utilizată aici pentru a putea avea un punct de comparație pentru următoarele noi metode prezentate. O metodă bazată pe SVM (SVM_FS) este ce-a de-a treia metodă prezentată. Această metodă se bazează pe nucleul liniar şi beneficiile corelării parametrilor nucleelor obținând rezultate mai bune şi mai rapide. Pentru ultima metodă autorul propune o combinație între rigurozitatea matematică a metodei bazate pe nuclee - SVM şi un algoritm inspirat din teoria evoluției – algoritmul genetic (GA_FS). Această nouă metodă face procesul de selecție a trăsăturilor mult mai rapid fără să se piardă din performanțele trăsăturilor selectate. De asemenea, autorul testează influența reprezentării datelor de intrare asupra acurateței clasificării. Sunt testate astfel trei tipuri de reprezentare a datelor: Binară, Nominală şi Cornell Smart. La sfârşit autorul prezintă unele rezultate comparative care arată că în cazul clasificării în mai multe clase cele mai bune rezultate se obțin când se alege un număr mic dar relevant de trăsături. După selectarea trăsăturilor relevante, autorul arată că utilizarea unui număr de trăsături între 2.5% şi 7% din numărul total de trăsături acuratețea clasificării este semnificativ mai bună (cu un maxim de 87.11% pentru metoda SVM_FS cu nucleul polinomial şi reprezentarea Cornell Smart a datelor de intrare). Dacă se creşte numărul de trăsături mai mult de 10% acuratețea clasificării nu mai creşte sau uneori chiar şi descreşte (la 86.52% pentru 2488 trăsături sau la 85.36% pentru 8000 trăsături). Când se utilizează metoda SVM_FS pentru selecția trăsăturilor se obțin rezultate bune ale clasificării pentru un număr mai mic de trăsături selectate (85.28% pentru 475 trăsături reprezentând aproximativ 2.5% din numărul total de trăsături) reducându-se astfel substanțial timpii de antrenare. În general metodele

SVM_FS şi GA_FS obţin rezultate mai bune decât IG iar SVM_FS obţine rezultate bune împreună cu nucleul polinomial iar GA_FS împreună cu nucleul Gaussian.

O altă observaţie importantă este că nucleul polinomial obţine în general rezultate mai bune dacă se utilizează împreună cu reprezentarea nominală a datelor de intrare iar nucleul Gaussian obţine în medie rezultate mai bune când se utilizează împreună cu reprezentarea Cornell Smart a datelor. Pentru documentele text cele mai bune rezultate se obţin cu nucleul polinomial (obţinând chiar un maxim de 87.11%) în comparaţie cu nucleul Gaussian care a obţinut un maxim de doar 84.85%. Metoda GA_FS obţine cele mai bune rezultate pentru un număr mai mare de trăsături (8000). De asemenea autorul a arătat că timpul de antrenare a clasificării creşte doar cu 3 minute când numărul de trăsături creşte de la 475 la 1309 şi creşte cu 32 minute când acest număr creşte de la 1309 la 2488. După câte ştim, autorul este primul care propune o metodă de selecţie a trăsăturilor utilizând algoritmii genetici cu SVM pentru calculul funcţiei de performanţă şi o structură simplificată a cromozomului.

La sfârşitul acestui capitol sunt prezentate anumite cercetări folosind documente web. Autorul analizează influenţa numărului de trăsături asupra îmbunătăţirii acurateţei clasificării. Cele mai bune rezultate sunt obţinute tot pentru un număr mic de trăsături. Pentru clasificarea documentelor web s-au obţinut aceleaşi concluzii ca cele obţinute pentru clasificarea documentelor Reuters. De exemplu pentru nucleul polinomial cu gradul 1 şi reprezentarea Cornell Smart a datelor acurateţea clasificării are doar o mică descreştere, de la 86.89% pentru 500 trăsături la 84.86% pentru 25646 trăsături. Pentru nucleul Gaussian şi reprezentarea binară a datelor descreşterea este mult mai semnificativă (de la 86.63% pentru 500 trăsături la 68.91% pentru 25646 trăsături). Pentru documentele web maximul acurateţei clasificării a fost obţinut de nucleul Gaussian (87.70%) comparativ cu nucleul polinomial care a obţinut un maxim de doar 87.01%. Dacă luăm în considerare toate rezultatele obţinute pentru toate gradele nucleului şi toate tipurile de reprezentări, în medie nucleul polinomial obţine rezultate mai bune decât cel Gaussian. Valorile maxime s-au obţinut pentru un număr relativ mic de trăsături (500 – 1500 trăsături ceea ce înseamnă aproximativ 2 - 4% din numărul total de trăsături).

În **capitolul 5** autorul demonstrează că metodele propuse pentru corelarea parametrilor nucleului asigură o modalitate simplă şi rapidă pentru obţinerea celor mai bune rezultate. Autorul prezintă de asemenea o vizualizare bi-dimensională a rezultatelor clasificărilor obţinute cu scopul de a avea o modalitate simplă de analiză vizuală a clasificărilor efectuate. De asemenea autorul prezintă câteva rezultate comparative între implementarea proprie a algoritmului SVM şi implementarea „LibSvm" utilizată în mod frecvent în literatură. La sfârşitul capitolului sunt prezentate câteva rezultate obţinute utilizând algoritmul de clustering implementat.

În cazul nucleului polinomial există mai multe combinaţii între parametrii pentru care se obţin cele mai bune rezultate. Formula de corelare propusă de autor asigură în aproape toate cazurile că se obţin rezultatele cele mai bune (fără a trebui să se facă multe teste pentru a găsi combinaţia potrivită de parametri). Astfel autorul a propus pentru nucleul polinomial ca deplasamentul care se adaugă la acest nucleu să fie egal cu de două ori gradul nucleului ($b=2*d$).

Pentru nucleul Gaussian autorul a propus o formulă care asigură obţinerea în toate cazurile a celor mai bune rezultate. De asemenea autorul a arătat că în cazul clasificării documentelor text nu este o idee bună ca parametrul C din cadrul nucleul să fie egal cu numărul de trăsături din mulţimea de antrenament aşa cum este utilizat în mod constant în literatură. Utilizând formula propusă de autor se obţin în medie rezultate cu 3% mai bune pentru nucleul polinomial şi cu 15% mai bune pentru nucleul Gaussian. După câte ştim, autorul este primul care a propus o metodă de corelare între parametrii nucleului, atât pentru cel polinomial cât şi pentru cel Gaussian.

Utilizând ideea de a modifica nucleul Gaussian, rezultatele obţinute utilizând LibSvm cu parametrii nucleului corelaţi sunt mai bune în comparaţie cu rezultatele utilizând LibSvm cu nucleul standard (cu un câştig mediu al acurateţei clasificării de 24.26% pentru nucleul polinomial şi respectiv de 28.44% pentru nucleul Gaussian). Programul implementat de autor, „UseSvm",

obține rezultate mult mai bune pentru clasificarea documentelor text decât LibSvm, cu un câştig mediu de 25.57%.

În **capitolul 6** se prezintă o metodă hibridă de clasificare automată a documentelor bazată pe mai mulți clasificatori de bază. În acest capitol autorul prezintă mai multe metode adaptive pentru proiectarea unor metaclasificatori bazați pe clasificatori SVM. Astfel sunt investigate 3 abordări pentru construirea unui metaclasificator eficient. Pe baza rezultatelor anterioare s-au selectat 8 clasificatori SVM de bază. Între aceşti clasificatori diferă nucleul folosit, gradul acestuia şi tipul de reprezentare a datelor de intrare. Pe baza clasificatorilor selectați s-a calculat limita superioară pe care o poate atinge acest metaclasificator, care este de 94.21%. Una din metodele propuse pentru implementarea metaclasificatorului este o metodă neadaptivă – Votul Majoritar iar celelalte două sunt metode adaptive.

Cu Votul Majoritar acuratețea clasificării obținută este de doar 86.38%. Evident că documentele care sunt clasificate corect doar de un clasificator nu pot fi clasificate corect de această metodă. Rezultatul obținut este de cu 0.73% mai mic decât cea mai mare valoare obținută de către unul dintre clasificatorii de bază şi este mai mare decât media rezultatelor clasificărilor făcute de fiecare clasificator selectat (85.26%).

Metaclasificatorul bazat pe distanță euclidiană (SBED) obține cele mai bune rezultate acuratețea clasificării crescând până la 92.04% după 14 paşi (antrenare / testare). Această valoare este cu doar 2.17% mai mică decât limita superioară a metaclasificatorului. Această metodă este de asemenea şi mai rapidă. Ultima metodă prezentată este metaclasificatorul bazat pe cosinus (SBCOS), care încearcă să fie mult mai riguros deoarece încearcă să găsească întotdeauna cel mai bun clasificator la un moment dat. Ca o consecință, timpul de antrenare pentru SBCOS este în medie mai mare cu 20 minute comparativ cu SBED care răspunde în doar 22 minute.

Deoarece în lumea reală antrenarea clasificatorilor trebuie să se facă pe seturi foarte mari de documente în **capitolul 7** autorul dezvoltă o strategie pentru lucrul cu aceste mulțimi mari de date. Această strategie nu creşte exponențial timpul de antrenare când datele de intrare cresc şi nu are pierderi substanțiale în acuratețea clasificării când se încearcă reducerea acestora. Pentru aceasta se propune şi se implementează o metodă în trei etape care în prima etapă reduce numărul vectorilor din mulțimea de intrare. În cea de-a doua etapă, considerat etapă de învățare, se selectează doar acei vectori care sunt considerați ca fiind relevanți în procesul de clasificare. Cea de-a treia etapă este etapa propriu-zisă de clasificare în care se foloseşte o dimensiune redusă a mulțimii de intrare. Autorul observă că acuratețea clasificării a scăzut în medie cu 1% când mulțimea de date se reduce cu 40%.

La sfârşitul capitolului 7 sunt prezentate câteva experimente care arată că testele prezentate nu sunt semnificativ influențate de alegerea mulțimilor de antrenare şi testare. În acest scop autorul propune o altă metodă pentru selecția mulțimii de antrenament şi de test şi repetă testele. Se poate observa că rezultatele obținute cu noua grupare a datelor de intrare sunt apropiate de rezultatele obținute cu prima grupare a mulțimilor de date. Uneori prima grupare obține rezultate mai bune cu 1%, alteori a doua grupare obține rezultate mai bune. Întotdeauna diferența dintre aceste seturi nu este mai mare de 2%. In medie pe toate rezultatele efectuate diferența este aproape 0. Ca o concluzie rezultatele prezentate nu sunt influențate de selectarea seturilor de antrenare şi de testare. Cu alte mulțimi de date metodele de selecție a trăsăturilor caracteristice şi algoritmii de clasificare obțin rezultate comparabile. Aceste concluzii încurajează autorul să utilizeze clasificatorii implementați în domenii cum ar fi clasificarea documentelor web.

8.2 Contribuții generale şi dezvoltări ulterioare

Această teză, cu contribuțiile originale prezentate, poate fi utilă în cazul în care se doreşte dezvoltarea unui sistem bazat pe clasificarea automată a documentelor web, în special

reorganizarea automată a paginilor web întoarse de motoarele de căutare clasice. Astfel, reordonarea şi gruparea să poată fi făcută pe baza conținutului paginilor.

De asemenea această teză prezintă soluții pentru selectarea diferitelor nuclee şi reprezentări a vectorilor de intrare în funcție de tipul acestora. Astfel când avem date de tip text este indicat să utilizăm nucleul polinomial cu reprezentarea nominală a datelor de intrare şi un grad mic al nucleului. Dacă nu se obțin rezultate acceptabile, deoarece datele pot fi puternic suprapuse, atunci se poate încerca cu nucleul Gaussian cu reprezentarea Cornell Smart a vectorilor de intrare. Pentru a se putea obține rezultate bune rapid fără prea multe încercări se recomandă utilizarea nucleelor cu parametrii corelați după cum au fost prezentați în lucrare.

Este de asemenea indicat să se utilizeze mai mulți clasificatori combinați într-o metodă adaptivă pentru îmbunătățirea rezultatelor fără să crească semnificativ timpul de lucru. O idee interesantă pe care doresc să o abordez în viitor este utilizarea calculului paralel oferit de sistemele multiplrocesor pentru reducerea timpilor de clasificare în cadrul metaclasificatorului.

De asemenea sunt prezentate metode care fac ca algoritmii de clasificare să lucreze rapid şi cu mulțimi de date foarte mari fără a descreşte mult calitatea învățării.

În experimentele viitoare voi încerca să combin metodele de clasificare cu metodele de clustering cu scopul de a utiliza documente etichetate şi neetichetate într-un algoritm de clasificare hibrid. Ideea este de a schimba primul pas din algoritmul de clustering, când se specifică hiperplanul inițial prin alegerea unui procentaj de vectori suport, cu un pas de clasificare pentru găsirea acestui hiperplan. În acest pas se prezintă un număr mic de date etichetate pentru algoritmul de clasificare cu scopul de a găsi coeficienții α_i care vor fi utilizați ca valori inițiale pentru procesul de clustering. Aceasta ne oferă astfel posibilitatea de a utiliza în partea de antrenare mai multe date neetichetate.

O problemă majoră care apare la toți algoritmii de clasificare şi clustering este că ei devin greu de utilizat în situații reale. De exemplu, ei au o problemă când trebuie să lucreze cu noi documente pentru care nici o trăsătură caracteristică din noul document nu se găseşte în mulțimea de antrenament. Astfel trebuie să existe metode de actualizare a algoritmului pentru a lucra cu noi trăsături. Noul algoritm obținut va clasifica într-un mod oarecare documentul prin crearea unei reuniri între mulțimile de trăsături. Această problemă apare deoarece mulțimea documentelor de antrenament nu poate conține rădăcinile tuturor cuvintelor. Metodele de selecție a trăsăturilor aleg trăsăturile care sunt relevante pentru mulțimea de antrenament. După cum am văzut în această teză algoritmii de clasificare obțin rezultate mai bune când se utilizează un număr mic dar relevant de trăsături. Astfel antrenarea cu câteva trăsături creşte numărul de documente care nu vor putea fi ulterior clasificate. Ca o dezvoltare ulterioară doresc să realizez teste cu familii de cuvinte şi să utilizez ca trăsătură doar un reprezentant din acea familie. Astfel, numărul de trăsături se va reduce semnificativ şi numărul de fişiere care vor putea fi clasificate ulterior va creşte. Această metodă poate creşte acuratețea clasificării şi poate fi utilizată în pasul de selecție a trăsăturilor. Pentru a putea obține aceasta se poate utiliza baza de date WordNET care conține o parte din familiile de cuvinte pentru limba engleză. De asemenea, pentru creşterea numărului de documente care vor putea fi ulterior clasificate, se pot folosi metode bazate pe sinonime şi pe problematica polisemiei cuvintelor. Astfel, păstrând doar un cuvânt din mai multe sinonime putem reduce numărul de trăsături şi creşte numărul de documente care pot fi clasificate ulterior. De asemenea, detectând sensul cuvântului în propoziție putem evita clasificarea documentelor diferite în aceeaşi categorie. Baza de date WordNet poate furniza astfel de informații, dar deocamdată pentru un număr redus de documente.

9 Bibliografia tezei

[Ack97_1] Ackerman, M., *The DO-I-Care Agent: Effective Social Discovery and Filtering on the Web*, Proceedings of RIAO'97: Computer-Assisted Information Searching on the Internet, pages 17-31, 1997.

[Ack97_2] Ackerman, M., Starr, B., Pazzani, M., *DO I Care? – Tell Me What's Change on the Web*, In Proceedings of the American Association for Artificial Intelligence Spring Symposium on Machine Learning, 1997.

[Ack97_3] Ackerman, M, Billsus, D., Gaffney, S., Hettich, S., Khoo, G., Kim, D., Klefstad, R., Lowe, C.,. Ludeman, A., Muramatsu, J., Omori, K., Pazzani, M., Semler, D., Starr, B., Yap, P., *Learning Probabilistic User Profiles: Applications to Filtering Interesting Web Sites, Notifying Users of Relevant Changes to Web Pages, and Locating Grant Opportunities*, AI Magazine 18(2) pages 47-56, 1997.

[Alb04] Albanese, M., Picariello, A., Sansone, C., Sansone, L., *A Web Personalization System based on Web Usage Mining Techniques*, In Proceedings of the World Wide Web Conference 2004, New York pp. 288-289, 2004.

[And04] Andronico, P., Buzzi, M., Leporini, B., *Can I Find What I'm Looking For?*, In Proceedings of the World Wide Web Conference 2004, New York, pp. 430-431, 2004.

[Bal97] Ballard, D., *An Introduction to Neural Computation*, the MIT Press, 1997.

[Bar02] Barbat, B., *Agent oriented intelligent systems*, Romania Academy Publishing House, Bucharest, 467 pages, 2002 (in Romanian).

[Bar03] Barbu, C., Marin, S., *Information Filtering Using the Dynamics of the User Profile*, In Proceedings of The 16th International FLAIRS Conference, 2003.

[Bei02] Beil, F., Ester, M., Xu, X., *Frequent Term-Based Text Clustering*, In SIGKDD02 Exploration: Newsletter on the Special Interest Group on Knowledge Discovery & Data Mining, ACM Press, Alberta Canada, 2002.

[Ber99] Berners-Lee, T., *Weaving the Web*, Orion Business Book, 1999.

[Ber02] Berendt, B., Hitho, A., Syumme, G., *Towards Semantic Web Mining*, Springer-Verlag Berlin Heidelberg, ISWC 2002, pp. 264-278, Berlin, 2002.

[Bha00] Bhatia, S., *Selection of Search Terms Based on User Profile*, ACM Explorations, pages. 224-233, 1998.

[Bra94] Brazdil, P. B. Gama, J., and Henery, B., *Characterizing the applicability of classification algorithms using meta-level learning*. Proceedings of the 7th European Conference on Machine Learning (ECML-94)(83-102), 1994.

[Bos92] Boser, B. E., Guyon, I. M., Vapnik, V., *A training algorithm for optimal margin classifiers*, in proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, pages 144-152, Pittsburgh ACM Pres, 1992

[Cha00] Chakrabarti, S., *Data mining for hypertext: A tutorial survey*, Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM Explorations 1(2), pages. 1-11,2000.

[Cha03] Chakrabarti, S., *Mining the Web. Discovering Knowledge from Hypertext Data*, Morgan Kaufmann Publishers, USA, 2003.

[Cha05] Charlesworth, I., *Integration fundamentals*, Ovum, 2005.

[Che98] Chen, L., Sycara, K., *WebMate: A Personal Agent for Browsing and Searching*, Proceedings of the 2nd International Conference on Autonomous Agents, pages 132-139, 1998.

[Che00] Chen, H., Dumais, S., – *Bringing Order to the Web: Automatically Categorizing Search Results*, In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 145-152, 2000.

[Chi03] Chih-Wei, H., Chih-Chang, C., Chih-Jen, L., *A Practical Guide to Support Vector Classification*, Department of Computer Science and Information Engineering National Taiwan University, 2003 (Available at http://www.csie.ntu.edu.tw/~cjlin/papers/ guide/ guide).

[Chr02] Christoph, K., Veit, B., *Visual Representation and Contextualization of Search Results*, *List and Matrix Browser*, In Proceedings of International Conference on Dublin Core and Metadata for e-Communities, pp. 229-234, 2002.

[Coo99] Cooley, R., Mobasher, B., Srivastava, J., *Data preparation for mining World Wide Web browsing patterns*, Journal of Knowledge and Information Systems 1(1), pages 5-32, 1999.

[Cov91] Cover, T. M.,. Joy, T. A., *Elements of Information Theory*, Jhon Wiley & Sons Interscience Publication, 1991.

[Cro99] *Introduction to Data Mining and Knowledge Discovery*, Tow Crows Corporation, a short introduction to a new book ,1999 (Available at http://www.twocrows.com/booklet.htm).

[Dav06] Davies J., Studer R., Warren P., *Semantic Web Technologies Trends and Research in Ontology-based Systems*, John Wiley & Sons, Ltd, 2006.

[Dee90] Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R., *Indexing by latent semantic analysis*, Journal of the American Society for Information Science, 41, pages. 391-407, 1990.

[Der00] Dertouzos, M., *What will be: How the New World of Information Will Change Our Lives*, Technical Publisher, Bucharest, 2000 (Translation in Romanian by F. G. Filip et al.).

[Die95] Dietterich, T. G., Bakiri, G., *Solving multi-class learning problems via error-correcting output codes*, Journal of Artificial Intelligence Research, 1995

[Dim00] Dimitrova, N., Agnihotri, L., Wei, G., *Video Classification Based on HMM Using Text and Face*, Proceedings of the European Conference on Signal Processing, Finland, 2000.

[Dou04] Douglas, H., Tsamardinos, I., Aliferis C., *A Theoretical Characterization of Linear SVM-Based Feature Selection*, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada,2004.

[Dum00] Dumais, S., Hao Chen, *Hierarchical Classification of Web Content*, Proceedings of the 23rd international ACM SIGIR Conference on Research and Development in Information Retrieval, pages 256-263, 2000.

[Fen04] Fensel, D., *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*, Second Edition, Springer –Verlag Berlin Heidelberg, 2004.

[For04] Forman, George, *A Pitfall and Solution in Multi-Class Feature Selection for Text Classification,* Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004.

[Gab04] 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.

[Geo99] Goecks, J., Shavilk, J., *Automatically Labeling Web Pages Based on Normal User Action*, Proceedings of the International Joint Conference on Artificial Intelligence. Workshop on Machine Learning for Information Filtering, pages 573-580, 1999.

[Gue00] Guerra-Salcedo, C., Chen, S., Whitley, D., Smith, S., *Fast and Accurate Feature Selection Using Hybrid Genetic Strategies*, CEC00, Proceedings of the Congress on Evolutionary Computation, CEC00, July 2000.

[Gun03] Gunnain, R., Menzies, T., Appukutty, K., Srinivasan, A., *Feature Subset Selection with TAR2less*, Available from http://menzies.us/pdf/03tar2less.pdf, 2003

[Gol89] Goldberg, G., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, New York, 1989.

[Gru93] Gruber, T., *A Translation approach to portable ontologies*. Knowledge Acquisition 5, pages 199-220, 1993.

[Hun03] Hung, C., Wermter, S., Smith, P., *Hybrid Neural Document Clustering Using Guided Self-Organization and WordNet*, Published by the IEEE Computer Society, IEEE 2003.

[Hof99] Hofmann, T., *Probabilistic Latent Semantic Analysis*, In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, pages. 289-296, UAI 1999.

[Hol75] Holland, J., *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[Ian00] Ian H., Witten, E. F., *Data Mining, Practical Machine Learning Tools and Techniques with Java implementation*, Morgan Kaufmann Press, 2000.

[Jai00] Jaideep, S., Cooley, R., Mukund, D., Pang Ning Tan, *Web - Usage Mining: Discovery and Applications of Usage Patterns from Web Data*, Technical Report, Department of Computer Science and Engineering University of Minnesota, 2000 (Available at http://maya.cs.depaul.edu/~classes/ect584/papers/srivastava.pdf).

[Jai01] Jaiwei, H., Micheline K., *Data Mining. Concepts and techniques*, Morgan Kaufmann Press, 2001.

[Jeb00] Jebara, T., Jaakkola, T., *Feature selection and dualities in maximum entropy discrimination*, In Uncertainty in Artificial Intelligence 16, 2000.

[Jeb04] Jebara, T., *Multi Task Feature and Kernel Selection for SVMs*, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004.

[Jim04] Romng, J., Huan, L., *Robust Feature Induction for Support Vector Machine*, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004.

[Joa99] Joachims, T., *Making large-scale support vector machine learning practical*, In B. Scholkopf, C. J. C. Burges, A.J. Smola (Eds): *Advances in kernel methods: Support vector learning*, MIT Press, 1999, pp. 169-184.

[Jur03] Jurafsky, D, Pradhan, S, Ward, W., Hacioglu, K., Martin, J., – *Shallow Semantic Parsing using Support Vector Machines*, Proceedings of the Human Technology Conference / North America chapter of the Association of Computational Linguistics, Boston, 2003.

[Kai02] Kai, Yu, Schwaighofer, A., Volker, T., Wei-Ying, M., HongJing, Z., *Collaborative Ensemble Learning: Combining Collaborative and Content Based Information Filtering,* Technical Report Siemens AC CT IC, Munich, 2002.

[Kai03] Kai, Yu, Schwaighofer, A., Volker, T., Wei-Ying M., HongJing Z., C*ollaborative Ensemble Learning: Combining Collaborative and Content Based Information Filtering via Hierarchical Bayes*, Proceeding of the 19th Conference, Uncertainty in Artificial Intelligence, pages 616-623, 2003.

[Kan02] Kanungo, T., Mount, D.M., Netanyahu, N.S., Pitko, C.D., Wu, A., *An Efficient* k-*Means Clustering Algorithm: Analysis and Implementation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, col. 24, no. 7, July 2002.

[Kim00] Kim, G., Kim, S., *Feature Selection Using Genetic Algorithms for Handwritten Character Recognition*, Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, Amsterdam, 2000.

[Kit98] Kittler, J., Hatef, M., Durin, R.P.W., Mates, J., *On Combining Classifiers*, IEEE Transactions and Pattern Analysis and Machine Intelligence, 1998.

[Koh97] Kohonen, T., *Self-Organizing Maps*, Second edition, Springer Publishers, 1997.

[Kum01] Kummamuru, K., Krishnapuram, *A clustering algorithm for asymmetrically related data with its applications to text mining*, In Proceedings of CIKM, pages 571-573, 2001.

[Kum04] Kummamuru, K., Lotilikar, R., Roy, S., Singal, K., Krishnapuram, R., *A Hierarchical Monothetic Document Clustering Algorithm for Sumarization and Browsing Search Results*, In Proceedings of the Thirteenth International World Wide Web Conference, pages 658-665, 2004.

[Law99] Lawrence, S., Giles, C. L., *Accessibility of information on the Web*,Nature,400, pages 107-109, 1999.

[Law01] Lawrie, D., Croft, W. B., Rosenberg, A., *Finding topic words for hierarchical summarization*, In proceedings of SIGIR, pages 349-357, 2001.

[Lin02_1] Lin, W.-H., Houptmann, A., *News Video Classification Using SVM-based Multimodal Classifier and Combination Strategies*, International Workshop on Knowledge Discovery in Multimedia and Complex Data, Taiwan, 2002.

[Lin02_2] Lin, W.-H., Jin, R., Houptmann, A., *A Meta-classification of Multimedia Classifiers,* International Workshop on Knowledge Discovery in Multimedia and Complex Data, Taiwan, May 2002.

[Lug98] Luger, G. F., Stubblefield, W. A., *Artificial Intelligence*, Addison Wesley Longman, Third Edition, 1998.

[Lym03] Lyman, P., *How Much Information?* School of Information Management and System, University of California at Berkeley, http://www.sims.berkeley.edu/research/projects/how-much-info-2003.

[Mit97] Mitchell, T., *Machine Learning*, McGraw Hill Publishers, 1997.

[Mla98] Mladenic, D., *Feature Subset Selection in Text Learning*, Proceedings of the 10th European Conference on Machine Leraning (ECML-98), pages 95-100, 1998.

[Mla99] Mladenic, D., Grobelnik, M. *Feature selection for unbalanced class distribution and naïve bayes*, In Proceedings of the 16th International Conference on Machine Learning ICML, p.258-267,1999.

[Mla04] Mladenic, D., Brank, J., Grobelnik, M., Milic-Frayling, N., *Feature Selection Using Support Vector Machines* The 27th Annual International ACM SIGIR Conference (SIGIR2004), pp 234-241, 2004.

[Mob96] Mobasher, B., Jain, N., Han, E.-H., Strivastava, J., *Web Mining: Pattern Discovery from World Wide Web Transactions*, Technical Report, 1996.

[Mor03_1] **Morariu, D.**, Curea, G., *Learning using the Support Vector Concept*, Proceedings of Scientific Workshop „The Science Challenge in XXI Century", organized by Land Forces Military Academy Sibiu, ISBN 973-8088-85-2, pages 29-36, Sibiu, December 2003.

[Mor03_2] Curea, G., **Morariu D**., *Structure of Web data using XML*, Proceedings of Scientific Workshop „The Science Challenge in XXI Century", organized by Land Forces Military Academy Sibiu, ISBN 973-8088-85-2, pages 21-28, Sibiu, December 2003.

[Mor05_1] **Morariu, D.**, *Web Information Retrieval*, 1st PhD Report, University „Lucian Blaga" of Sibiu, February, 2005, http://webspace.ulbsibiu.ro/daniel.morariu/html/Docs/Report1.pdf.

[Mor05_2] **Morariu, D.**, *Classification and Clustering using SVM*, 2nd PhD Report, University „Lucian Blaga" of Sibiu, September, 2005, http://webspace.ulbsibiu.ro/ daniel.morariu/html/Docs/Report2.pdf

[Mor06_1] **Morariu, D.**, Vintan, L., *A Better Correlation of the SVM Kernel's Parameters*, Proceedings of the 5th RoEduNet IEEE International Conference, ISBN (13) 978-973-739-277-0, Sibiu, June, 2006.

[Mor06_2] **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.

[Mor06_3] **Morariu, D.**, Vintan, L., Tresp, V., *Evolutionary Feature Selection for Text Documents using the SVM*, Proceedings of the 3rd International Conference on Machine Learning and Pattern Recognition (MLPR'06), ISSN 1503-5313, vol.15, pp. 215-221, Barcelona, Spain, October, 2006.

[Mor06_4] **Morariu, D.**, Vintan, L., Tresp, V., *Meta-classification 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.

[Mor06_5] **Morariu, D.**, *Relevant Characteristics Extraction*, 3rd PhD Report, University „Lucian Blaga" of Sibiu, October, 2006, http://webspace.ulbsibiu.ro/daniel.morariu /html/Docs/Report3.pdf

[Mor06_6] **Morariu, D.**, Vintan, L., Tresp, V., *Evaluating some Feature Selection Methods for an Improved SVM Classifier*, International Journal of Intelligent Technology, Volume 1, no. 4, ISSN 1305-6417, pages 288-298, December 2006

[Mor07_1] **Morariu, D.**, Vintan, L., *Kernel's Correlation for Improving SVM in Text Documents' Classification*, (Accepted) "Acta Universitatis Cibiniensis", Technical Series, "Lucian Blaga" University of Sibiu, 2007

[More05] Morello, D., *The human impact of business IT: How to Avoid Diminishing Returns*, published in Information Age magazine, Australian Computer Society, 2005.

[Nel00] Nello C., Shawe-Taylor, J., *An Introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000.

[Ord03] Ordones, C., *Clustering Binary Data Streams with K-means*, Conference of Data Mining and Knowledge Discovery, San Diego, 2003.

[Pla99_1] Platt, J., *First training of support vector machines using sequential minimal optimization*. In B. Scholkopf, C.J.C. Burges, and A. J. Smola, editors, Advances in Kernel Methods – Support Vector Learning, pages 185-208, Cambridge, MA, 1999, MIT Press.

[Pla99_2] Platt J., *Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods*, In Advances in Large Margin Classifiers, A. Smola, P. Bartlett, B. Scholkopf, D. Schuurmans, eds., MIT Press, Cambridge, pages 61-74, 1999.

[Sch02] Schoslkopf Bernhard, Smola Alexander, *Learning with Kernels, Support Vector Machine*, MIT Press, London, 2002.

[Siy01] Siyang, G., Quingrui, L., Lin, M., *Meta-classifier in Text Classification*, http://www.comp.nus.edu.sg/~zhouyong/papers/cs5228project.pdf, a research group, 2001.

[Str00] Strivastava, J., Cooley R., Deshpande M., Tan Pang-Ning, *Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data*, ACM SIGKDD Explorations, pages 12-23, 2000.

[Ray00] Raymond, K., Hendrik, B., *Web mining research: A survey*, In SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM Press, pages 1-15,2000.

[Vap95] Vapnik, V., *The nature of Statistical learning Theory*, Springer New York, 1995.

[Vap01] Vapnik, V., Asa, Ben-Hur, Horn, D., Sieglmann H. T., *Support Vector Clustering*, Journal of Machine Learning Research 2, pages 125-137, 2001.

[Yan97] Yang, Y., Pedersan, J.O., *A Comparative Study on Feature Selection in Text Categorization*, Proceedings of ICML, 14th International Conference of Machine Learning, pages 412-420, 1997.

[Yu03] Yu, H., Yang, J., Han, J., *Classifying Large Data Sets Using SVM with Hierarchical Clusters*, In SIGKDD03 Exploration: Newsletter on the Special Interest Group on Knowledge Discovery & Data Mining, ACM Press, Washington, DC, USA, 2003

[Wah99] Wahba, G., *Support Vector Machine, reproducing kernel Hilbert space and the randomized GACV*. In B. Scholkopf, C. J. C. Burgers, and A. J. Smol;a, editors, Advances in Kernel Methods – Support Vector Learning, pages 69-88, Cambridge, MA, 1999, MIT Press.

[Whi94] Whitely, D., *A genetic Algorithm Tutorial, Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, 1994.

[Wiz04] Wojciech, W., Krzysztof, W., Wolciech, C., *Periscope – A System for Adaptive 3D Visualization of Search Results*, Association for Computing Machinery , p.29-40, 2004.

Referințe web

[Aol] search.aol.com (Web directories, accessed in December 2006).

[Dmoz] www.dmoz.org (Open Directory Project - Web directories, accessed in December 2006).

[Excite] www.excite.com (Web directories, accessed in December 2006).

[Google] www.google.com (search engine, accessed in December 2006).

[Ir] http://www.cs.utexas.edu/users/mooney/ir-course/ Information Retrieval java Application, accessed in December 2006.

[kartoo] www.kartoo.com (graphical representation of search results and monitoring a specific site, accessed in December 2006).

[LibSvm] http://www.csie.ntu.edu.tw/~cjlin/libsvm - accessed in December 2006.

[LookSmart] http://www.looksmart.com/ (Web directories with preclassified Web pages) – accessed in December 2006

[SurfMind] www.surfmind.com/Web (Web directories and searching into a specific domain, accessed in November 2006).

[Surfwax] www.surfwax.com (help user to find different synonyms grouped after domain for every search word, accessed in December 2006).

[SparseMatrix] www.cs.utk.edu/~dongarra/etemplates/node372.html - presenting an optimal modality to store sparse matrix.

[Periscope] http://periscope.kti.ae.poznan.pl/ (graphical representation of the search results) – accessed in December 2006.

[Reu00] Misha Wolf and Charles Wicksteed- Reuters Corpus: http://www.reuters.com/researchandstandards/corpus/ Released in November 2000 accessed in June 2005

[Rdf] www.w3.org/rdf.

[Yahoo] www.yahoo.com (search engine and Web directory, accessed in December 2006).

[Vivisimo] www.vivisimo.com (2 levels hierarchical representation of the search results, accessed in December 2006).

[WebCr] www.Webcrawler.com (one level hierarchical representation of the search results, accessed in December 2006).

[WebMate] www.cs.cmu.edu/~softagents/Webmate (the proactive agent that learn the user profile to increase the quality of the search results) – accessed in December 2006.

[Weka] www.cs.waikato.ac.nz/~ml/weka/ - accessed in December 2006.

[WordNet] http:\\www.cogsci.princeton.edu\obtain – online lexical system – accessed in December 2006.