ŠIAULIŲ UNIVERSITETAS
TECHNOLOGIJOS FAKULTETAS
ELEKTRONIKOS KATEDRA

Arūnas Šatas

TEKSTO TURINIO ANALIZĖ DIRBTINIAIS NEURONŲ TINKLAIS

Magistro darbas

Vadovas
doc. dr. G. Daunys

ŠIAULIAI, 2006

ŠIAULIŲ UNIVERSITETAS
TECHNOLOGIJOS FAKULTETAS
ELEKTRONIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. G. Daunys
2006 06

TEKSTO TURINIO ANALIZĖ DIRBTINIŲ NEURONŲ TINKLAIS

Magistro darbas

Vadovas
doc.dr.G. Daunys
2006 06

Atliko
RM4 gr. stud.
A. Šatas
2006 06

Recenzentas
ŠU Technologijos fakulteto elektronikos katedros
doc. dr. D. Miniotas
2004 06

ŠIAULIAI, 2006

Šatas A.Textual Analysis Using Artificial Neural Networks. Master thesis of electonics / research advisor dr. G. Daunys, Šiauliai University, Technological Faculty, Electronics department. – Šiauliai, 2006.

SUMMARY

The theme of Master project is a posibility to use arificial neural networks for textual analysis and automatic categorization of textual documents in editorial programs. The task of the work was to analyze diferent methods of text clasification using diferent neural networks (SOM, Feed Forward, Learning Vector Quantization, etc.). There are much researchers who works on text clasification and artificial neural networks, but there is no practical fitting of such research. In this work I tried to find posibilities and dificulties of practical use of text clasificaion. I find that very important thing is initial amount and quality of information and not all neural networks fits for solving text categorization problems.

TURINYS

PRIEDAI

LENTELĖS

PAVEIKSLAI

ĮŽANGA

Šiandieniniame pasaulyje, kuriame nuolat daugėja elektroniniu būdu pateikiamos tekstinės informacijos, darosi vis sunkiau ją apdoroti ir suprasti. Visame pasaulyje išplitus ir toliau vis sparčiau tebeplintant internetui, bei kitai žiniasklaidai, informacijos radimui vis plačiau naudojami įvairūs teksto kategorizavimo metodai. Be jų šiuo metu negalima įsivaizduoti nei vienos internete veikiančios paieškos sistemos. Spaudoje (tiek elektroninėje, tiek „popierinėje“) pateikiami straipsniai yra kategorizuoti – tvarkingai suskirstyti pagal temas. Tačiau didėjant informacijos apimčiai reikia ir naujų metodų, įgalinančių straipsnių kategorizavimą automatizuoti. Visame pasaulyje yra kuriami ir naudojami įvairūs metodai. Jie dažnai pritaikomi interneto paieškai, elektroniniam paštui, įvairių įmonių bei organizacijų dokumentų tvarkymui ar medicinos įstaigų elektroninių kartotekų sudarymui. Pagrindinis teksto klasifikavimo tikslas – suskirstyti straipsnius į kelias (ar keliasdešimt) iš anksto apibrėžtų kategorijų. Keletas plačiausiai paplitusių metodų yra Naive Bayes (NB), Support Vector Machines (SVM), k-nearest Neighbor (kNN), bei dirbtinių neuronų tinklai.

Automatinis teksto klasifikavimas taip pat naudingas dėl to, kad tokių būdu taupomi kaštai. Juk tą patį darbą žmogus, atlikdamas rankiniu būdu, sugaišta nemažai laiko, kuris taip pat šiais laikais yra gana brangus, be to žmonės negali dirbti be pertraukų po 24 valandas per parą. Kita vertus, perdavus teksto kategorizavimo darbą automatinėms sistemoms, iškyla jų tikslumo problema. Šiame baigiamąjame magistro darbe bus bandoma įvertinti galimybę, spaudos redakcinėje sistemoje panaudoti automatinio teksto kategorizavimo metodus. Tyrimai bus atliekami naudojantis dirbtinių neuronų tinklų metodais.

1. Tekstinės informacijos kategorizavimas

1.1 Kategorizavimo panaudojimo sritys

Teksto klasifikavimas (kategorizavimas) – tai natūralios kalbos tekstų priskirimas tam tikroms iš anksto apibrėžtoms kategorijoms, remiantis tų tekstų turiniu. Remiantis teksto kategorizavimo metodais yra sukurti identifikatoriai, leidžiantys nustatyti teksto žanrą ar autorių (dažnai naudojami nežinomo autoriaus nustatymui), kalbos identifikatoriai, leidžiantys nustatyti kokia kalba parašytas tekstas, taip pat teksto kategorizavimo priemonės, leidžiančios straipsnius kategorizuoti pagal temas, bei jų turinį. Visos šios priemonės leidžia sutaupyti nemažai laiko ir kitų išteklių.

Be anksčiau paminėtų panaudojimo sričių, visi šie matodai labai plačiai taikomi šiuolaikinėse informacinėse technologijose, tokiose kaip informacijos paieška internete, personalizuotos (skirtos tam tikrai vartotojų grupei ar atskiram vartotojui) informacijos pateikimas, nepageidautinų elektroninių laiškų filtravimas. Ypač aktyvūs tyrimai šiuo metu vyksta būtent nepageidautinų elektroninių laiškų („spamo“) filtravimo srityje.

Šiuolaikinėse redakcinėse sistemose automatinis dokumentų kategorizavimas dar nėra plačiai paplitęs, nes dažnai laikraščių ar žurnalų redakcijose šį darbą atlieka žmonės. Tačiau didėjant informacijos apimtims ir norint sutaupyti laiko (ypač dideliuose interneto portaluose) labai naudinga būtų automatinius metodus panaudoti ir čia.

1.2 Įvadas į teksto kategorizavimą

Teksto kategorizavimas (arba teksto klasifikavimas) – tai atsitiktinės funkcijos $\Phi': D \times C \rightarrow \{T,F\}$ (kuri apibrėžia kaip dokumentai turi būti suklasifikuoti) aproksimavimas funkcijos $\Phi: D \times C \rightarrow \{T,F\}$(vadinamos klasifikavimo funkcija) reikšmėmis. Šiose išraiškose $C = \{c_1,\ldots,c_{|C|}\}$ – iš anksto apibrėžtų kategorijų rinkinys, o D – klasifikuojamų dokumentų rinkinys. Jei $\Phi'(d_j,c_i) = T$ (true), tuomet

d_j bus vadinamas teigiamu pavyzdžiu kategorijai c_i, jei $\Phi'(d_j,c_i) = F$ (false) – tai bus neigiamas pavyzdys kategorijai c_i.

Kategorijos yra tik simboliniai pavadinimai ir paprastai jokios papildomos informacojos jose nėra. Jos savyje neturi tokios informacijos kaip publikavimo data, dokumento tipas ar publikacijos šaltinis. Klasifikavimas vykdomas remiantis tik ta informacija, kurią turi pats klasifikuojamas dokumentas savo turinyje.

Teksto kategorizavimas yra gana subjektyvus dalykas, nes keli klasifikuotojai (ar klasifikatoriai) gali tas pačias publikacijas priskirti skirtingoms kategorijoms. Pavyzdžiui straipsnis apie krepšininko Darjušo Lavrinovičiaus automobilio vagystę gali būti priskirtas ir sporto, ir kriminalinėms naujienoms, priklausomai nuo to, kokia bus aproksimuojanti funkcija $\Phi:D\times C\rightarrow\{T,F\}$. Šį straipsnį skirtingi klasifikuotojai (ar klasifikatoriai) gali priskirti bet kuriai iš minėtų kategorijų arba abiems iš karto. Taip pat gali nepriskirti nei vienai iš jų.

Priklausomai nuo situacijos, teksto kategorizavimas gali būti vienos reikšmės arba daugiareikšmis. Pirmuoju atveju tik vienai kategorijai c_i gali būti priskirtas tik vienas dokumentas d_j, antruoju atveju d_j gali būti priskirtas bet kuriai kategorijai, esančiai ribose $0 \leq n_j \leq |C|$. Specialus vienos reikšmės kategorizavimo būdas yra dvinaris teksto kategorizavimas, kai d_j gali būti priskirtas arba duotąjai kategorijai c_i, arba šios kategorijos subkategorijai c_i'. Daugiareikšmis kategorizavimas pagal kategorijas $C=\{c_1,\ldots,c_{|C|}\}$paprastai gali būti apibrėžtas kaip nepriklausomas dvinaris kategorizavimas pagal kategorijas $\{c_i,c_i'\}$, kai $i = 1,..., |C|$. Tokiu atveju kategorijai c_i klasifikavimo funkcija bus apibrėžiama $\Phi_i: D \rightarrow\{T,F\}$, o jos aproksimuojama funkcija $\Phi_i': D \rightarrow\{T,F\}$.

Kuriant klasifikavimo sistemą, galime išskirti tris pagrindinius žingsnius [1]:

1. Dokumentų indeksavimas;
2. Klasifikatoriaus mokymas;
3. Apmokyto klasifikatoriaus įvertinimas;

1.3 Dokumentų indeksavimas

Dokumentų indeksavimas – tai dokumentų d_j suskirstymas į kompaktiškas atvaizdavimo formas pagal jų turinį taip, kad klasifikatorių kuriantis algoritmas ar pats klasifikatorius galėtų juos tiesiogiai atpažinti. Paprastai tekstas d_j yra atvaizduojamas kaip terminų svorių $d_j = (\omega_{1j},..., \omega_{|T|j})$ vektorius. Šiuo atveju T yra rinkinys terminų (dar vadinamų požymiais), kurie aptinkami bent kartą keliuose iš pasirinktų tekstų ir svoris $0 \leq \omega_{kj} \leq 1$ nustato požymio t_k svarbą dokumentui d_j. Paprastai k reikšmės būna tarp 1 ir 5.

Paprastai indeksavimo metodas būna apibrėžiamas indeksavimui parinktų terminų ir metodo, pagal kurį skaičiuojami terminų svoriai. Pirmiausiai būna išrenkami dažniausiai tekstuose pasitaikantys žodžiai arba jų kamienai (kad kaitoma galūnė neturėtų įtakos kategorizavimui). Tuomet yra atmetami „stop žodžiai“ tokie kaip jungtukai (kitose kalbose dar ir artikeliai), bei neutralūs žodžiai, kurie pasitaiko visuose tekstuose, bet negali būti traktuojami kaip tam tikros temos požymiai (tai veiksmažodžiai, būdvardžiai ir tt.). Dažnai naudinga indeksuojant tekstus, rankiniu būdu pridėti tipiškas frazes, apibūdinančias tą temą (pavyzdžiui kategorizuojant tekstus apie automobilius pagal gamintojus, galima įterpti frazes „prancūziškas automobilis“ ar „europoje gaminamos transporto priemonės“).

Apskaičiuojant terminų svorius naudojami statistiniai metodai. Vienas populiariausių metodų remiasi formule tf * idf [2] kuri reiškia du skaičiavimo būdus:

1. Kuo dažniau dokumente d_j randamas terminas t_k, tuo jis svarbesnis šiam dokumentui (terminų dažnumo kriterijus);
2. Kuo didesniame kiekyje dokumentų randamas terminas t_k, tuo mažesnę įtaką jis daro jų išskyrimui (atvirkščio dokumentų dažnumo kriterijus);

Kad apmokymo laikas sutrumpėtų ir klasifikatorius veiktų tiksliau, teksto kategorizavime dažnai parenkamas ne T terminų skaičius, bet mažesnis (tačiau iš anksto apibrėžtas). Tai vadinama požymių atrinkimu. Kitaip sakant, kiekvinam terminui t_k yra apskaičiuojamas tam tikras teigiamas ar neigiamas svoris koreliacijai

su c_i, ir tik geriausiai dokumentui tinkantys terminai panaudojami to dokumento atvaizdavimui.

1.4 Klasifikatoriaus mokymas

Teksto klasifikatorius kategorijai c_i yra automatiškai sukuriamas apmokymo algoritmo, kuris remiasi iš anksto suklasifikuotų dokumentų sąvybėmis, priskiriančiomis dokumentą kategorijoms c_i ar c_i‘. Šis algoritmas garantuoja, kad naujas dokumentas, turintis panašius požymius bus priskirtas kategorijai c_i. Kad sukurtume tokį klasifikatorių kategorijų rinkiniui C, turime turėti tokį dokumentų rinkinį Ω, kad reikšmė Φ‘(d_j,c_i) būtų apibrėžta kiekvienam (d_j,c_i). Kuriant klasifikatorių, dokumentų rinkinys yra padalinamas į tris dalis – apmokymo rinkinį T_r, suderinimo (patvirtinimo) rinkinį V_a ir testavimo rinkinį T_e. Pirmasis rinkinys T_r sukuria klasifikatorių. Antrasis rinkinys V_a naudojamas tiksliam sukurtos sistemos suderinimui. Testavimo rinkinys T_e galutinai suderina sistemą ir nustato klasifikatoriaus efektyvumą. Abiem atvejais (V_a ir T_e), nustatant klasifikatoriaus efektyvumą tikrinamas sistemos išėjimo ir iš anksto parinktų dokumentų atitikimo laipsnis.

1.5 Apmokyto klasifikatoriaus įvertinimas

Pagrindiniai rodikliai, įvertinantys klasifikatorių yra apmokymo naudingumo koeficientas (arba vidutinis laikas klasifikatoriui Φ_i sukurti iš turimo dokumentų Ω rinkinio), klasifikavimo naudingumo koeficientas (vidutinis laikas, reikalingas suklasifikuoti dokumentus pagal Φ_i reikšmes), ir efektyvumas (vidutinis Φ_i tikslumas atpažystant tekstinius dokumentus). Efektyvumas ko gero yra svarbiausias kriterijus, nes dažnai apmokymo ir klasifikavimo laikas (ypač apmokymo) yra aukojamas tam, kad gaunami rezultatai būtų geresni. Teksto klasifikavime efektyvumas dažniausiai matuojamas atitikimo (π) ir atkūrimo (ρ) kombinacija. Taip pat naudojamas tikslumo matas (τ). Šiuos matus galime apskaičiuti pasinaudoję lentele Nr.1.

1.1 lentelė

Efektyvumo matų nustatymo lentelė

		Tikroji reikšmė	
		Taip	Ne
Klasifikatoriaus sprendimas	Taip	TP	FP
	Ne	FN	TN

TP – dokumentai, teisingai priskirti kategorijai;

FP – dokumentai klaidingai priskirti kategorijai;

FN – dokumentai klaidingai atmesti iš kategorijos;

TN – dokumentai teisingai atmesti iš kategorijos;

Remiantis lentelės Nr.1 duomenimis galime apskaičiuoti efektyvumo parametrus:

$\pi = TP/(TP+FP)$ (1.1)

$\rho = TP/(TP+FN)$ (1.2)

$\tau = (TP+TN)/n$, kur $n = TP+FP+FN+TN$ (1.3)

Pavieniai parametrai π ar ρ netinka sistemos įvertinimui, nes jei klasifikatorius priskirs visoms kategorijoms reikšmę „Taip“, tai atkūrimas (ρ) bus 100%, o atitikimas (π) bus labai žemas [3], todel, kad tiksliai įvertintume efektyvumą turime rasti harmoninę reikšmę:

$$F_1 = 2 \pi \rho / \pi + \rho \quad (1.4)$$

Kai efektyvumas yra skaičiuojamas kelioms kategorijoms, atskirų kategorijų įvertinimas gali būti atliekamas dviem būdais: mikroįvertinimas (kiekvienai kategorijai skaičiuojamas proporcingai pagal apmokymo rinkinių skaičių), arba makroįvertinimas (visoms kategorijoms skaičiuojamas vienodai).

2. Raktinių žodžių svorių nustatymas

Statistiniai raktinių žodžių (terminų) svorio nustatymo metodai nagrinėja kaip konkretus terminas straipsnyje gali lemti visų dokumentų pasiskirstymą kategorijose. Terminas, kuris yra išskirtinis kažkuriai kategorijai, gali suskirstyti

atsitiktinius tekstus į tam tikrus rinkinius. Kad įvertintume termino svarbumą konkrečiai kategorijai, turime nustatyti ryšį tarp to termino dažnumo visame dokumentų rinkinyje ir konkrečioje temoje. Raktinių žodžių pasiskirstymą galima atvaizduoti jų priklausymu arba nepriklausymu kategorijai (2.1 lentelė), kuris bus panaudotas tolesniuose poskyriuose.

2.1 lentelė

Terminų atitikimas kategorijoms

		Dokumentai		
		Priklauso kategorijai	Nepriklauso kategorijai	Terminų rinkinys
Terminai	Yra	r (A)	n-r (B)	n
	Nėra	R-r (C)	N-R-n+r (D)	N-n
		R	N-R	N

r – kategorijai priklausantys dokumentai, kuriuose yra nagrinėjamas terminas;

n – visi rinkinyje esantys dokumentai, kuriuose yra nagrinėjamas terminas;

R – skaičius dokumenų, esančių išskirtoje kategorijoje;

N – skaičius visų rinkinyje esančių dokumentų;

2.1 Atvirkštinis dokumentų dažnis (IDF)

Bendra terminų svorio nustatymo schema yra vadinama atvirkštiniu dokumentų dažnumu. Dokumentų rinkinyje kai kurie terminai bus randami daugumoje dokumentų, o kai kurie – tik viename ar keliuose. Žodžiai, randami visuose tekstuose nėra reikšmingi, o žodžiai, pasitaikantys tik keliuose straipsniuose turi kur kas didesnę reikšmę. Būtent pagal juos dokumentai gali būti skirstomi į kategorijas. Teksto atpažinimo sistema turi išskirti tam tikrus terminus, kurie pasitaiko tik tam tikros specifikos dokumentuose.

Remiantis 2.1 lentele galime šį metodą aprašyti matematiškai:

$$IDF_t = \log_2(N / n); \quad (2.1)$$

Dažnai teksto atpažinime naudojami ir kiti terminai:

1. Dokumentų dažnumas – dokumentų d_j, kuriuose randamas konkretus terminas t_k dažnumas dokumentų rinkinyje Ω;
2. Termino dažnumas – to paties termino t_k pasikartojimo dažnumas viename dokumente d_j;

Kadangi termino dažnumas tiesiogiai priklauso nuo dokumento dydžio (kuo ilgesnis dokumentas tuo daugiau tų pačių žodžių), reikia normalizuoti šį dydį. Tam tikslui yra skaičiuojamas dokumento dydžio vektorius (visų dokumentų vektoriai keliami kvadratu, sumuojami ir iš jų ištraukiama kvadratinė šaknis), o po to termino dažnumas yra dalinamas iš gautojo vektoriaus.

2.2 Liktinis atvirkščias dokumentų dažnis (RIDF)

Šis metodas yra atvirkštinio dokumentų dažnio atskiras atvejis, kuriuo remiantis svoriai skaičiuojami sumuojant atvirkštinį dokumentų dažnį ir jo išankstinę tikimybę pagal Poison modelį [8].

$$RIDF_t = idf_t + \log_2(1 - p(0;\lambda_t)) \quad (2.2)$$

Čia $\lambda_t = cf_t / N$ – vidutinis termino t_k pasikartojimo skaičius ;

$1 - p(0;\lambda_t)$ – Poison tikimybė, kad t_k bent kartą randamas dokumente;

2.3 Artimų dokumentų dažnis (RDFThresh)

Tai metodas, kurį galime traktuoti kaip dokumentų dažnumo dvejetainį atvejį. Jis naudoja tiesioginę informaciją atpažinimui terminų, kurie dažniausiai pasitaiko tik tam tikros specifikos dokumentuose. Čia remiamasi tuo, kad terminai, dažniau pasitaikantis konkrečioje kategorijoje yra svarbesni už rečiau pasitaikančius.

$$RDF_t = r \quad (2.3)$$

r – kategorijai priklausantys dokumentai, kuriuose yra nagrinėjamas terminas.

2.4 Informacijos įgyjimas (Information Gain - IG)

Tai matavimo vienetas iš informacijos teorijos, kuris matuoja entropijos skirtumus kategorijos prognozavime, remiantis žodžio buvimu arba nebuvimu analizuojamame dokumente. Remiantis 2.1 lentele galime aprašyti šį metodą matematiškai:

$$IG_t = -\Pr(rel)\log\Pr(rel) + \Pr(t)\Pr(rel \mid t)\log\Pr(rel \mid t) + PR(t)\Pr(rel \mid t)\log\Pr(rel \mid t) =$$
$$= -\frac{R}{N}\cdot\log\frac{R}{N} + \frac{r}{N}\cdot\log\frac{r}{n} + \frac{R-r}{N}\cdot\log\frac{R-r}{N-n} \qquad (2.4)$$

2.5 Bendra (abipusė) informacija (DFDiff)

Tai dar vienas metodas iš informacijos teorijos. Jis remiasi vienos atsitiktinės reikšmės neapibrėžtumo mažėjimu, žinant apie kitą reikšmę. Metodas naudojamas nustatyti termino išsidėstymui dokumentų rinkinyje. Taip pat jis gali būti naudojamas nustatant ryšį tarp turimo termino ir tyriamos temos. Remiantis 2.1 lentele aprašome šį metodą matematiškai:

$$F1/MI_t \approx \log\frac{A\times N}{(A+C)\times(A+B)} = \log\frac{r/R}{n/N} \qquad (2.5)$$

2.6 χ^2 chi kvadratas

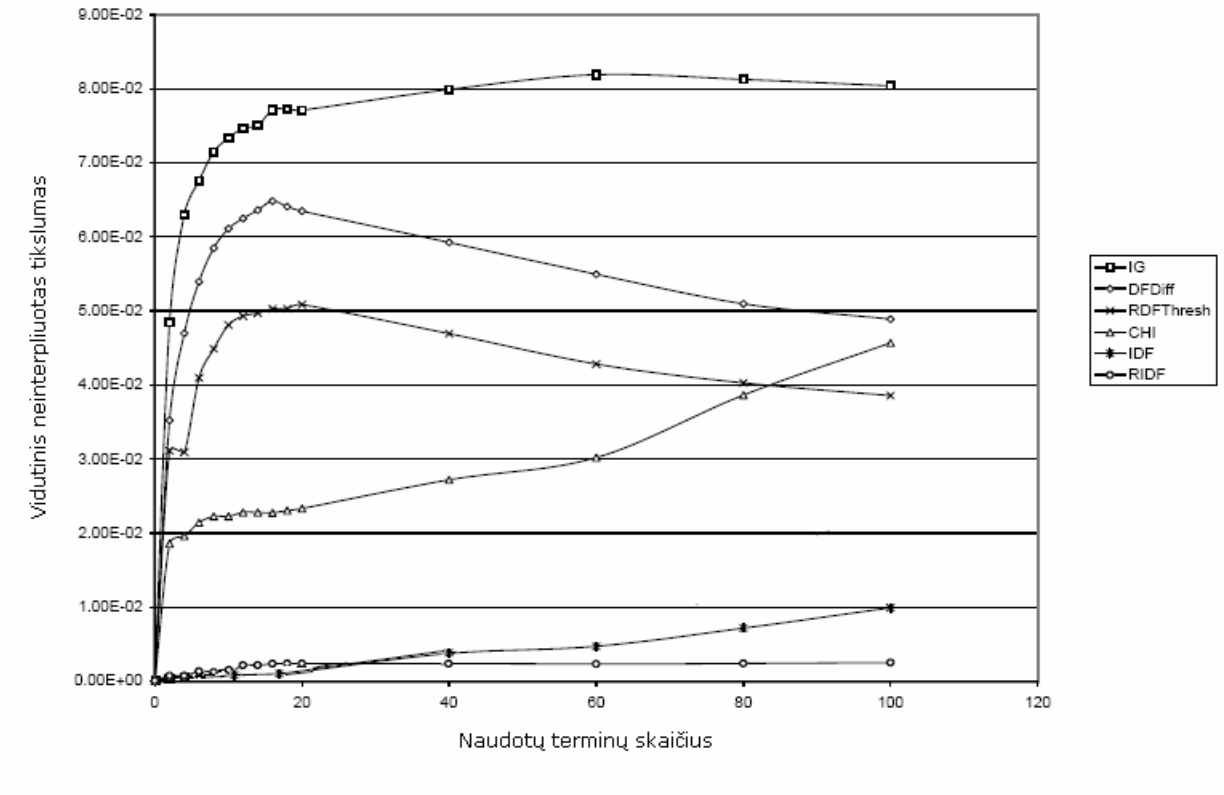
Chi kvadratas yra panašus į Bendrą (abipusę) informaciją, kadangi taip pat matuoja dviejų kintamųjų abipusę priklausomybę. Šiuo atveju skaičiuojamas skirtumas tarp matuojamo ir prognozuojamo terminų dažnumo. Jei skirtumas yra didelis, tai galime vertinti, kad panašumas tarp terminų yra mažas ir jie yra nepriklausomi vienas nuo kito. Taigi šiuo atveju matuojame termino

nepriklausomybės kategorijai lygį. Vėlgi remiantis 2.1 lentele galime metodą aprašyti:

$$\chi^2 = \frac{N \cdot (AD - CB)^2}{(A + C) \cdot (B + D) \cdot (A + B) \cdot (C + D)} = \frac{N \cdot (rN - nR)^2}{R \cdot n \cdot (N - R) \cdot (N - n)} \qquad (2.6)$$

2.7 Svorių nustatymo metodų palyginimas

Remiantis [9] tyrimais buvo nustatyta, kad ne visi metodai vienodai tiksliai suskirsto terminus. Geriausius rezultatus duoda IG metodas, prasčiausius – RIDF (pav. 2.1).



Pav. 2.1 Terminų svorių nustatymo metodų tikslumo palyginimas [9]

2.8 Požymių išrinkimas

Požymių išrinkimas dažnai naudojamas neinformatyvių žodžių pašalinimui iš dokumentų. Tokiu būdu didinamas kategorizavimo efektyvumas ir supaprastinamas skaičiavimas [3]. Kita priežastis naudoti požymių išrinkimą – tai leidžia išvengti nereikalingų didelių požymių vektorių (sumažinamas duomenų dydis). Vietos taupymas ir duomenų valdymo supaprastinimas tampa labai svarbus kai dirbama su dideliais tekstų rinkiniais. Yra du pagrindiniai būdai požymiams atrinkti – tai svarbių terminų įtraukimas ir nereikšmingųjų atmetimas.

Galime išskirti keletą požymių atrinkimo algoritmų:

1. Imti požymius po vieną ir su jais klasifikuojant išrinkti geriausius. Tik kartais šis algoritmas nėra visiškai tinkamas, nes vienas požymis gali duoti blogą rezultatą, o kelių požymių kombinacija klasifikuoti visai be klaidų;
2. Nagrinėti požymius po du ar daugiau. Šis metodas tinka jei požymių nėra daug;
3. Požymių atrinkimas su paskatinimu remiasi tuo, kad geriausi požymiai pasitaiko dažniau negu blogi. Tuoment atskiriama m dažniausiai pasitaikančių požymių ir iš jų perrenkama n reikalingų požymių;
4. forward selection – atrenkamas vienas geriausias požymis ir tada bandomos jo kombinacijos su likusiais, po to atrenkamas antras ir tt. Šis būdas vadinamas nuosekliu pridėjimu. Taip pat galimas ir nuoseklus atmetimas;
5. Genetiniai algoritmai;

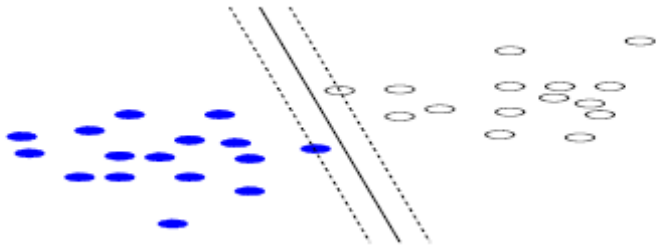
Kai kurie mokslininkai mano, kad požymių išrinkimas yra žalingas [4], nes tokiu būdu prarandama informacija, tačiau kitos tyrimų grupės šią metodiką naudoja gana plačiai. Toks nuomonių skirtumas tik parodo, kad požymių išrinkimas yra tikrai svarbi tyrimų kryptis.

3. Kategorizavimo metodai

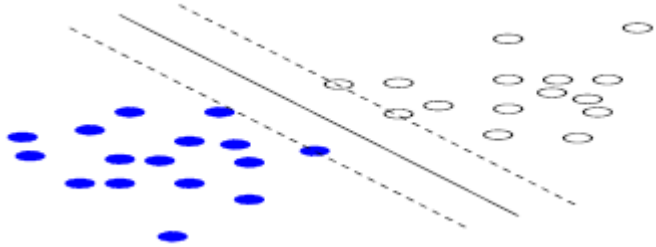
Tekstinės informacijos kategorizavimui naudojami keli metodai. Jų pasirinkimas apsprendžiamas norimo gauti tikslumo, veikimo greičio, bei skaičiavimų paprastumo. Labiausiai paplitę klasifikavimo metodai yra Atraminių vektorių mašinos (Support vector machines), k-Nearest Neighbour klasifikatoriai, Tiesinis mažiausių kvadratų metodas, Naive Bayes klasifikatoriai, Sprendimų medžiai (Decision trees) bei dirbtinių neuronų tinklai.

3.1 Atraminių vektorių mašinos (SVM)

Atraminių vektorių mašinos yra vienas tiksliausių ir labiausiai paplitusių metodų [5]. Šiuo atveju turime vektorinę erdvę, kur uždavinys yra rasti skiriamąjį paviršių, kuris geriausiai išskirtų turimą informaciją į dvi klases. Tam tikslui turime išvesti ribą tarp šių klasių (Pav. 3.1 ir Pav. 3.2).

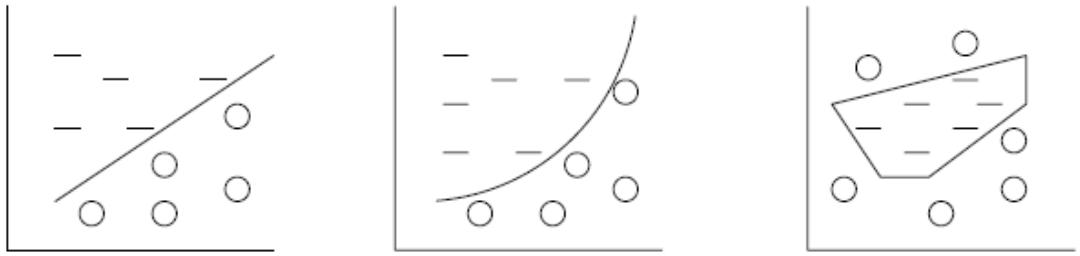


Pav. 3.1 Sprendimų linija (paryškinta) su siauresne sprendimo riba



Pav. 3.2 Sprendimų linija (paryškinta) su platesne sprendimo riba

Pav. 3.1 ir Pav 3.2 taškai, kuriuos kerta punktyrinės linijos yra atraminiai vektoriai (Support vectors). Paprastumo dėlei šiame darbe bus aptarta tik dviejų dimencijų erdvė ir tiesinis taškų atskyrimas, nors šis metodas gali būti naudojamas ir daugiadimencinėje erdvėje, bei su įvairių formų skiriamaisiais paviršiais (Pav. 3.3).



Pav. 3.3 Skirtingi atraminių vektorių mašinų modeliai

Paryškintos linijos iliustracijose (Pav 2.1 ir Pav 2.2) yra galimi skiriamieji paviršiai (sprendimų linijos). Jie abu teisingai atskiria duomenis į dvi grupes. Lygiagrečios punktyrinės linijos rodo ribas, kuriose gali kisti skiriamieji paviršiai išvengiant neteisingo duomenų suklasifikavimo. Atstumai tarp dviejų lygiagrečių linijų vadinami ribomis. SVM uždavinys yra rasti skiriamąjį paviršių, kuriam esant, riba tarp taškų būtų pati didžiausia.

Matematiškai skiriamasis paviršius aprašomas taip:

$$\vec{\omega} \cdot \vec{x} - b = 0 \qquad (3.1)$$

$\vec{x}$ yra sutartinis taškas, o vektorius $\vec{\omega}$ ir konstanta b gaunami iš sistemos apmokymo duomenų rinkinio. Jei apmokymo rinkinį aprašysime $D = \{(y_i, \vec{x_i})\}$ kur $y_i \in \{\pm 1\}$, o $\vec{x}$ (+1 jei turime teigiamą pavyzdį ir -1 jei turime neigiamą pavyzdį), tai SVM uždavinys yra rasti $\vec{\omega}$ ir b, kad būtų tenkinamos žemiau aprašytos sąlygos:

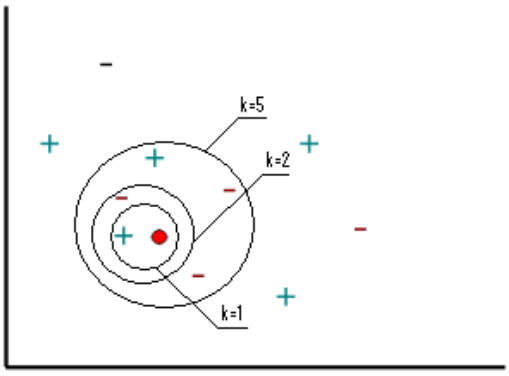
$\vec{\omega} \cdot \vec{x} - b \geq +1$ kai $y_i = +1$; (3.2)

$\vec{\omega} \cdot \vec{x} - b \leq -1$ kai $y_i = -1$; (3.3)

Įdomi SVM sąvybė yra ta, kad skiriamasis paviršius yra apibrėžiamas tik tų taškų, kurie yra nutolę nuo skiriamosios linijos tiksliai atstumu $\frac{1}{|\vec{\omega}|}$. Šie taškai yra vadinami atraminiais vektoriais ir yra vieninteliai elementai apmokymo duomenų rinkinyje, kurie turi įtaką. Net jei pašalintume visus likusius duomenis iš apmokymo rinkinio, algoritmas vis tiek įsimintų tą pačią atskyrimo funkciją. Tai ko gero unikali SVM sąvybė skirianti šį metodą nuo kitų. Thorsten Joachim [4] šį metodą pirmas pritaikė teksto kategorizavimui ir palygino rezultatus su kitais metodais. Šių tyrimų duomenimis – SVM stipriai lenkė kitus autoriaus bandytus metodus.

3.2 k-Nearest neighbor klasifikatorius (kNN)

k-Nearest neighbor klasifikatorius taip pat yra plačiai paplitęs ir naudojamas daugelyje tyrimų statistinis metodas. kNN algoritmas yra gana paprastas – padavus į sistemą klasifikuojamą dokumentą, klasifikatorius randa k artimiausių dokumentų ("kaimynų") ir naudoja šių dokumentų kategorijas naujojo dokumento svorių nustatymui. Apskaičiuotas panašumo į artimiausius "kaimynus" laipsnis ir nustato kuriai kategorijai dokumentas turi būti priskirtas. Svoriai apskaičiuojami remiantis atstumais nuo artimisusių "kaimynų" (kuo mažesnis atstumas – tuo didesnis svoris). Dokumento priskyrimas kategorijai priklauso nuo skaičiaus k. Pavyzdžiui jei k = 1 (Pav. 3.4), objektas bus priskirtas pliusams, jei k = 2 – objekto klasės nustatyti negalime, jei k = 5 – objektas bus priskirtas minusams.



Pav.3.4 kNN naujo objekto klasifikavimo priklausomybė nuo k

kNN kategorizavimo taisyklė matematiškai aprašoma taip:

$$y(\vec{x}, c_j) = \sum_{\vec{d}_i \in kNN} sim(\vec{x}, \vec{d}_i)\, y(\vec{d}_i, c_j) - b_j \qquad (3.4)$$

Čia $y(\vec{d}_i, c_j) \in \{0,1\}$ yra dokumento d_i priskyrimo kategorijai c_j funkcija (1, jei dokumentas priskiriams kategorijai ir 0 jei ne). $sim(\vec{x}, \vec{d}_i)$ yra panašumo tarp testinio dokumento $\vec{x}$ ir apmokymo dokumento $\vec{d}_i$ funkcija. b_j yra kategorijos slenkstis. Kitaip sakant, jei panašumas tarp dokumentų yra mažesnis (atstumas didesnis), negu nustatytas slenkstis – dokumentas į kategoriją negali patekti. Kiekvienai kategorijai slenkstis b_j yra nustatomas automatiškai su antruoju – suderinimo duomenų rinkiniu V_a (skyrius 1.4). Optimaliausią slenksčio reikšmę gauname tada, kai gaunama geriausia F_1 reikšmė (skyrius 1.5).

3.3 Tiesinis mažiausių kvadratų metodas (LLSF)

LLSF yra kartografinis metodas, kurį pasiūlė Yiming Yang [6] dar 1994 m. Šiuo atveju turime įėjimų/išėjimų poras apmokymui. Tai mokymo algoritmas su mokytoju. Įėjimo vektorius yra dokumentas, skirtas sistemos apmokymui, o išėjimo vektorius – atitinkama kategorija. Matematinė algoritmo išraiška yra:

$$F_{LS} = \arg\min_F \| FA - B \|^2 \qquad (3.5)$$

Čia matricos A ir B žymi apmokymo duomenis (įėjimo/išėjimo reikšmes), o matrica F_{LS} yra rezultatas, pagal kurį, priklausomai nuo įėjimo dokumentų požymių sužymimos išėjimo kategorijos. Panašiai kaip ir kNN atveju, čia sistema automatiškai "įsimena" slenkstį, skiriantį vieną kategoriją nuo kitos.

3.4 Naïve Bayes klasifikatorius (NB)

NB klasifikatorius yra tikimybinis metodas, kuris yra gana plačiai išnagrinėtas įvairioje literatūroje [7]. Pagrindinė šio metodo idėja – skaičiuojamos tikimybės, kad tam tikri raktiniai žodžiai bus prisikiriami tam tikroms kategorijoms, bei kategorizuojamų dokumentų tikibybės patekti į tam tikrą kategoriją. Šiuo atveju nenaudojami žodžių junginiai kaip požymiai, naudojami tik atskiri žodžiai. Kiekvienas raktinis žodis priskiriamas kiekvienai kategorijai su tam tikra tikimybe. Tokiu būdu, pagal požymius, esančius dokumente, galime nustatyti tikimybę, kad pasirinktas dokumentas pateks į atitinkamą kategoriją. Naudojant Naïve Bayes klasifikatorius vienas dokumentas yra priskiriamas vienai ir tik vienai kategorijai.

Tarkime, kad turime rinkinį kintamųjų $X = \{x_1,x_2,...,x_d\}$ ir mums reikia nustatyti tikimybę, kad įvykis C_j įvyks. $C = \{c_1,c_2,...,c_d\}$. Šiuo atveju X atitinka klasifikuojamų dokumentų rinkinį, o C – kategorijų rinkinį. Pasinaudodami Bayes taisykle gauname:

$$p(C_j| x_1,x_2,...,x_d) = p(x_1,x_2,...,x_d|C_j)\, p(C_j) \qquad (3.6)$$

Čia $p(C_j| x_1,x_2,...,x_d)$ yra tikimybė, kad X priklauso C_j. Iš čia gauname:

$$p(X \mid C_j) = \prod_{k=1}^{d} p(x_k \mid C_j) \qquad (3.7)$$

Toliau galime perrašyti taip:

$$p(C_j \mid X) = p(C_j)\prod_{k=1}^{d} p(x_k \mid C_j) \qquad (3.8)$$

Pasinaudoję Bayes taisykle, naują kintamąjį X priskiriame kategorijai C_j jei apskaičiuota tikimybė yra didžiausia.

4. Dirbtinių neuronų tinklų panaudojimas kategorizavimui

Dirbtinių neuronų tinklai imituoja smegenų veiklą, todėl turi sąvybę mokytis. Neurono matematinis modelis:

$$s = \omega_0 + \sum_{i=1}^{p} \omega_i x_i \qquad (4.1)$$

Čia ω_0 - pradinis svoris; ω_i -svorio pokytis; x_i – įėjimo reikšmė;

Šiuo metu dirbtinių neuronų tinklų metodai taikomi vis plačiau, be to naudojama daugybė jų mokymo algoritmų, kurie skiriami į dvi stambiausias klases – mokymą su mokytoju ir mokymą be mokytojo. Pirmuoju atveju sistemai duodami tiek įėjimo duomenys, tiek ir norimos gauti reikšmės. Su šiuo duomenų rinkiniu dirbtinių neuronų tinklas nusistato svorius tarp visų savo mazgų taip, kad kitą kartą padavus vien įėjimo duomenis jis juos galėtų atpažinti. Dažniausiai naudojami vienasluoksniai ir daugiasluoksniai perceptronai, tiesioginio sklidimo bei kiti tinklai. Mokymo be mokytojo algoritmas yra kiek kitoks – šiuo atveju duodami tik įėjimo duomenys ir tinklas pats „išmoksta“ tų duomenų struktūrą (čia populiariausi SOM arba Kohonen tinklai).

4.1 Perceptronas

Vienasluoksnis perceptronas (angl. single layer perceptron) – tai tiesiog vienas neuronas. Tuo tarpu daugiasluoksnis perceptronas (angl. multilayer perceptron) – tai daug neuronų, išdėstytų sluoksniais. Vieno sluoksnio neuronų išėjimai yra jungiami su kito sluoksnio įėjimais. Tinklą gali sudaryti tik du sluoksniai (tinklas be paslėptų sluoksnių) ir trys ar daugiau sluoksnių (tinklas su įėjimo sluoksniu, išėjimo sluoksniu ir su likusiais paslėptais sluoksniais). Dauguma dirbtinių neuronų tinklų priklauso šioms kategorijoms. Jei tinklas turi bent vieną paslėptą sluoksnį – juo galima spręsti bet kokio sudėtingumo atpažinimo uždavinius, priklausomai nuo neuronų skaičiaus.

Tiek vienasluoksnių, tiek ir daugiasluoksnių perceptronų apmokymo principas yra toks pat. Spręsdami klasifikavimo (kategorizavimo) uždavinį, pirmausiai turime rasti tinklo koeficientus (svorius) ω_i. Turime perceptroną:

$$g(x) = x_1\omega_1 + x_2\omega_2 + \omega_0 \qquad (4.2)$$

Perceptroną reikia apmokyti. Turime turėti rinkinį mokymo vektorių $(x_{11},...,x_{1p},t_1),...,(x_{n1},...,x_{np},t_n)$. Čia x_{ji} – įėjimo reikšmės, o t_j – tikslai (angl. targets). Apibrėžkime nuostolių funkciją, kurią reikia minimizuoti:

$$c = \sum_{j=1}^{n} (t_j - f(\omega_1 x_{j1} + ... + \omega_p x_{jp} + \omega_0))^2 \qquad (4.3)$$

Čia: n – mokymo duomenų kiekis;

p – požymių skaičius;

x_j – mokymo vektorius;

t_j – pageidautinas išėjimas;

Dabar reikia inicializuoti tinklą, t.y. sugeneruoti tinklo mazgams atsitiktinius svorius. Tada ieškome rajono, kuriame yra teisingas variantas ir toliau dirbame jame panaudodami mažiausių kvadratų metodą. Jei funcija turi daug minimumų, tai atsakymą gali būti sunku rasti. Pasinaudojame Niutono metodu:

$$\omega_{t+1} = \omega_t - \eta(\delta_c/\delta_\omega) \qquad (4.4)$$

Čia η – mokymo žingsnis. Žingsnį galime tiek didinti tiek ir mažinti, priklausomai nuo to kaip sekasi priartėti prie reikiamos reikšmės. Pradedant mokyti reikia priskirti pradinius svorius. Vienasluoksniui perceptronui galima duoti ir nulius, nes jam šios reikšmės nėra svarbios.

Jei duomenys yra normaliai pasiskirstę ($\mu x = Ex$ – vidurkis, $\sigma^2 = E(x - \mu)^2$ – variacija, σ – dispersija), tai normalinis tankis yra:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}(x-\mu)\sigma^{-2}(x-\mu)} \qquad (4.5)$$

Bendru atveju p-matėje erdvėje normalinis tankis yra:

$$f(x) = \frac{1}{(2\pi)^{p/2} \mid \Sigma \mid^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \qquad (4.6)$$

Čia Σ – kovariacinė matrica. x ir μ šiuo atveju yra vektoriai.

Jei turime dvi klases su vidurkiais μ_1 ir μ_2 ir kovariacines matricas Σ_1 ir Σ_2, gauname tokią diskriminacinę funkciją:

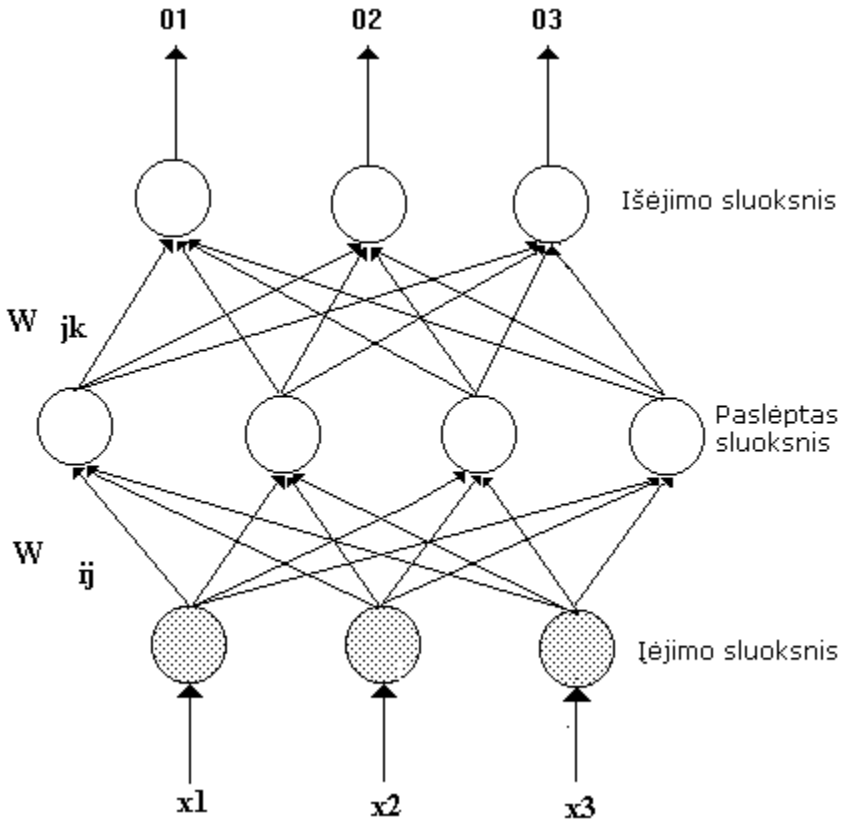
$$G(x) = -1/2(x - \mu_1)^T \Sigma_1^{-1}(x - \mu_1) - ½ \ln(|\Sigma_1|) + ½(x - \mu_2)^T \Sigma_2^{-1}(x - \mu_2) - ½ \ln(|\Sigma_2|) \qquad (4.7)$$

Ši funkcija vadinama kvadratine diskriminacine funkcija. Jei g(x)>0, tai x priklauso pirmąjai klasei, jei g(x)<0 – antrąjai.

Naudojant gradientinį mokymo algoritmą galimi du režimai – stochastinis ir totalinis. Pirmu atveju skaičiuojame gradientą kiekvienam mokymo vektoriui atskirai ir darome pataisymus. Totalinį režimą turime kai skaičiuojame gradientą visiems vektoriams kartu (imame vidurkį) prieš darydami pataisymą (tai kartais vadinama *batch mode*). Vienasluoksniams perceptronams geriau naudoti totalinį, o daugiasluoksniams – stochastinį režimus.

4.2 Tiesioginio sklidimo (feed forward) tinklai

Tiesioginio sklidimo tinklas – tai tas pats daugiasluoksnis perceptronas, todėl jam tinka ankstesniame skyriuje aprašyti mokymo algoritmai. Jis sudarytas iš daugybės paprastų neuronų išdėstytų sluoksniais. Kiekvienas neuronas yra sujungtas su visais ankstesnio sluoksnio neuronais (Pav. 4.1)



Pav. 4.1 Trijų sluoksnių tiesioginio sklidimo neuronų tinklas

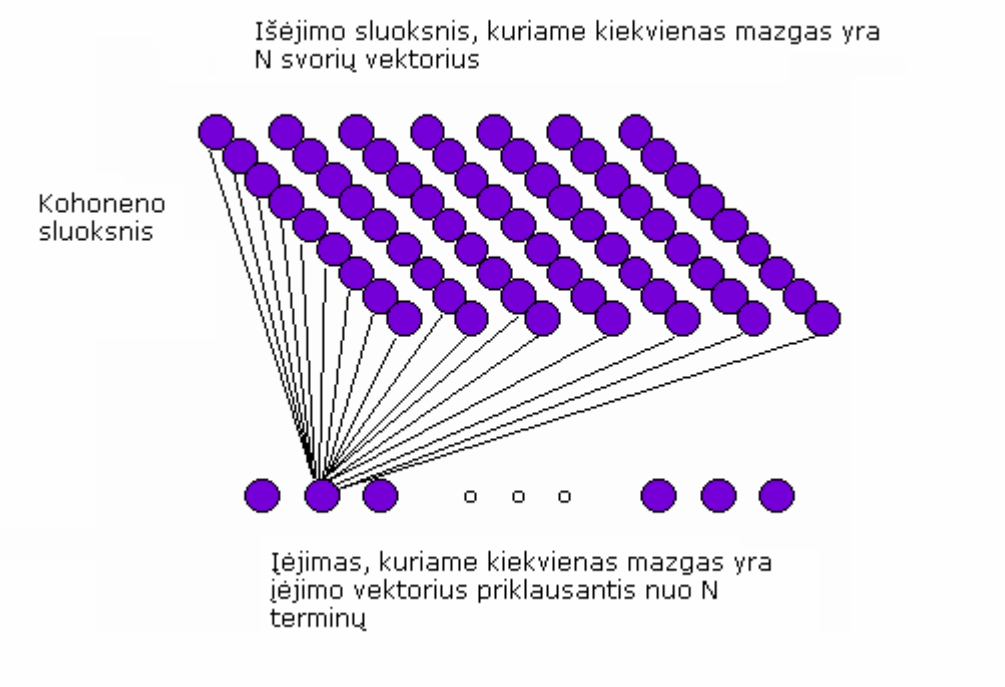
Tinkle ryšiai tarp neuronų nėra vienodi. Jie turi savo svorius. Būtent šiais svoriais ir yra koduojama informacija, esanti tinkle. Dažnai tinkle esantys neuronai dar vadinami mazgais. Šiame tinkle duomenys paduodami į įėjimo sluoksnį ir sklinda tinklu per visus sluoksnius iki išėjimo.

Tiesioginio sklidimo tinklo darbas gali būti skirstomas į dvi fazes – apmokymo ir klasifikavimo.

Apmokymo metu naudojamas mokymo su mokytoju algoritmas, t.y. tinklui duodamos ne tik įėjimo reikšmės, bet ir tikslų (angl. targets) reikšmės.

4.3 SOM tinklai

Teksto kategorizavime galima naudoti ir saivaime besiorganizuojančius žemėlapius (SOM) dar kitaip vadinamus Kohoneno žemėlapiais (Pav. 4.2). Šiuo atveju reikalingi tik pradiniai įėjimo duomenys, o tikslai nėra nurodomi. Šie tinklai, neturėdami nurodytų tikslų tiesiog suskirsto duomenis į tam tikrar grupes – klasterius, kuriuos po to galima sužymėti (angl. labeling) tinklui duodant žinomą pavyzdį.



Pav. 4.2 SOM neuronų tinklas

Kaip matome Pav. 4.2, SOM tinklas yra sudarytas iš dviejų sluoksnių – įėjimo sluoksnio ir išėjimo („žemėlapinio“) sluoksnio, kuris turi dvimatę struktūrą. Įėjimo sluoksnis naudojamas duomenų padavimui. Šiame sluoksnyje neuronų skaičius sutampa su turimu požymių (žiūr. skyrių 2.8) skaičiumi. Tinkle visi išėjimo neuronai yra sujungti su visais įėjimo neuronais ir tie ryšiai turi skirtingą stiprumą (svorius), kurie ir saugo informaciją apie tinklo veikimą. Iš pradžių išėjomo reikšmės yra inicializuojamos atsitiktinėmis reikšmėmis. Tuoment kiekvienas

įėjimo neuronas yra lyginamas su kiekvienu išėjimo neuronu ir randami tokie, kurių svoriai artimiausi. Tuomet „laimėjęs“ išėjimo sluoksnio neuronas laikomas turinčiu mažiausią Euklidinį atstumą tarp išėjimo sluoksnio neurono svorių vektoriaus ir įėjimo sluoksnio atitinkamo neurono svorio vektoriaus. Tokiu būdu įėjimo duomenys nukreipiami į konkrečią išėjimo sluoksnio vietą. Pasibaigus tinklo mokymui kai visi įėjimo neuronai jau būna susieti su tam tikrais išėjimo neuronais (paprastai po šimtų ar net tūkstančių mokymo ciklų), susidaro tam tikra struktūra, vadinama Kohoneno (arba SOM) žemėlapiais. Kohoneno žemėlapiu šis tinklas vadinamas todėl, kas suomių mokslininkas Kohonen pirmasis tyrė SOM panaudojimo galimybes teksto kategorizavime [10].

SOM algoritmas tekstinės informacijos kategorizavimui yra toks:

1. Inicializuojami įėjimo bei išėjimo sluoksnių neuronai ir svoriai tarp jų: imame N dažniausiai tekstuose sutinkamų terminų ir sukuriame N dydžio įėjimo sluoksnį. Sudarome išėjimo sluoksnį iš M neuronų. Tuomet priskiriame atsitiktinius svorius ω_{ij} visiems jų tarpusavio ryšiams.
2. Tinklui „rodome“ visus klasifikuojamus dokumentus iš eilės: visus dokumentus atvaizduojame kaip N dydžio įėjimo sekas, kuriose 1 bus tose vietose, kurios atitinka tekste esantį terminą ir 0 – tose, kurios atitinka žodį, nesantį duotąjame tekste. Kiekvienas dokumentas tinklui „rodomas“ keletą (ar net kelis šimrtus) kartų.
3. Apskaičiuojame atstumus tarp visų neuronų: skaičiuojame Euklidinį atstumą tarp įėjimo neurono d_j ir kiekvieno išėjimo neurono j.

$$d_j = \sum_{i=0}^{N-1} (x_i(t) - \omega_{ij}(t))^2 \qquad (4.8)$$

Čia $x_i(t)$ yra 1 arba 0, priklausomai nuo to, ar i-tasis terminas yra paduodamas tinklui t laiko momentu, ω_{ij} – svorių vektorius, nurodantis išėjimo sluoksnuo neurono j vietą tiriamų dokumentų rinkinyje.

4. Išrenkamas “nugalėjęs” išėjimo sluoksnio neuronas j* ir atnaujinami svoriai artimiausiems j* neuronams:

$$\omega_{ij(t+1)} = \omega_{ij}(t) + \eta(t)(x_i(t+1) - \omega_{ij}(t)) \qquad (4.9)$$

Po šio svorų atnaujinimo išėjimo sluoksnis suskirstomas tam tikrais regionais, kuriose atsiranda tik artimos įėjimų reikšmės. Čia η(t) yra klaidos mažėjimo koeficientas (0< η(t)<1), kuris mokant tinklą nuolat mažėja.

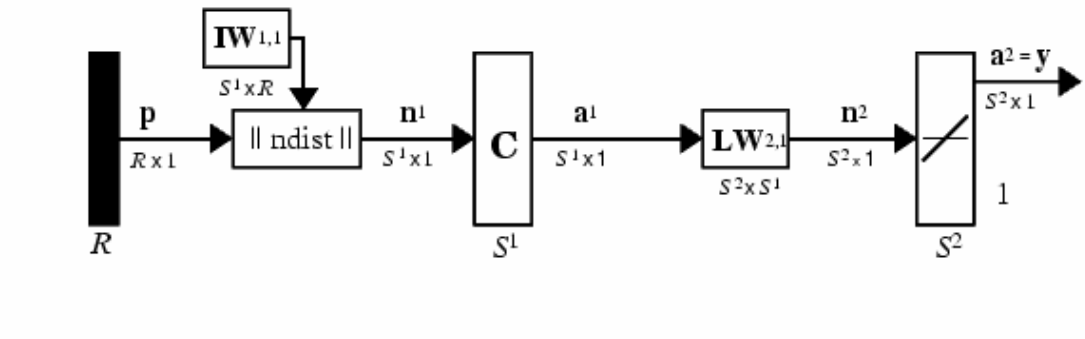
5. Regionų sužymėjimas tinkle: kai tinklas jau būna apmokytas, kiekvienai kategorijai parenkame įėjimo sekas, kuriose yra vienintelis 1, atitinkantis didžiausią svorį turintį terminą. Tokiu atveju išėjimo sluoksnio regionai yra sužymimį pagal temas.

Kad šis algoritmas veiktų tiksliau, galima kiekvienam terminui parinkti tam tikrą slenksti (pavyzdžiui žodis tekste turi pasikartoti mažiausisi du kartus). Tokiu būdu yra išvengiama atsisiktinio žodžių pasitaikymo su tema nesusijusiuose tekstuose.

SOM metodas yra tinkamas, kai turime didelį kiekį duomenų ir galime sudaryti pakankamai dideles įėjimo sekas. Kitais atvejais šis metodas nėra tikslus.

4.4 LVQ tinklai

LVQ (angl. Learning vector quantization) tinklai yra mišrus SOM tinklų ir tiesinių tinklų variantas (Pav.4.3).



Pav. 4.3 LVQ tinklo struktūrinė schema

Pav. 4.3 paveikslėlyje R žymimas įėjimo reikšmių skaičius, S^1 – lenktyniaujančio tinklo neuronų skaičius, S^2 – tiesinio neuronų tinklo mazgų skaičius.

Kaip matome, LVQ tinkas turi pirmą lenktyniaujančių neuronų sluoksnį ir antrą – tiesinį sluoksnį. Pirmasis sluoksnis įėjimo duomenis klasifikuoja panašiai kaip SOM tinklas, o antrasis sluoksnis priskiria pirmojo sluoksnio suformuotus regionus vartotojo apibrėžtoms tikslinėms reikšmėms. Klasės, kurias suformuoja pirmasis lenktyniaujantis sluoksnis, vadinamos subklasėmis, o antrojo sluoksnio suformuotos klasės – tikslų klasėmis. Abiejuose sluoksniuose yra po vieną neuroną kiekvienai klasei.

5. Automatinio kategorizavimo metodų naudojimas redakcinėje sistemoje

Pasaulio spaudos kompanijos naudoja daug įvairių redakcinių sistemų savo leidiniams parengti. Kadangi visuose spaudos leidiniuose straipsniai skirstomi į tam tikras temas (neskaitant kai kurių specializuotų leidinių), kategorizavimo uždavinys yra gana svarbus. Jei jį būtų galima pilnai automatizuoti – būtų sutaupoma nemažai laiko ir žmogiškųjų išteklių atliekant šį darbą.

5.1 Redakcinių sistemų apžvalga

Nemažai kompanijų pasaulyje siūlo gana universalias sistemas, į kurias integruota ne tik redakcinė programa, bet ir visos įmonės darbą padedantys organizuoti įrankiai įskaitant buhalterinę programą, CRM (ryšių su klientais valdymo įrankis), projektų planavimo priemonės ir tt. Tokią universalią sistemą „Ad Management“ siūlo kompanija Anygraaf. Ji gana patogi įmonės vadovui, galinčiam nesunkiai kontroliuoti visus savo darbuotojus, bet nelabai pritaikyta reporterių bei redaktorių poreikiams.

Wilkenson Scoop kompanija taip pat kuria nemažai programinės įrangos redakcijoms, bet jie produktus išskiria kur kas griežčiau. Jų produktas „SCOOP NewsPlanner“ yra pritaikytas tiek Macintosh, tiek ir Microsoft Windows platformoms. Pagrindinės sistemos galimybės – dienos darbų planavimas, straipsnių kūrimas ir redagavimas, ilgalaikis darbų planavimas, resursų parinkimas konkrečiam darbui ir tt. Scoop produkcijos trūkumas – papildomiems uždaviniams spręsti reikia diegti apildomas Scoop šeimos programas.

Dar viena įmonė, siūlantį panašaus pobūdžio produkciją – QPS. Tai jau labiau specializuota redakcinė sistema su daugybe reikalingu funkcijų, įgalinančių formatuoti ne tik tekstinę informaciją, bet ir iliustracijas, antraštes ir kitus grafinius elementus panašiai kaip naudojantis grafinėmis programomis – tokiomis kaip Adobe Fotoshop ar CorelDraw.

Internetinės žiniasklaidos redagavimui yra sukurta ir atviro kodo programų, tokių kaip „SendStory“. Jos automatiškai generuoja XML failus, suteikia galimybę importuoti tekstus ir publikuoti HTML formatu.

UAB „Informacijos alėja“ yra sukūrusi redakcinę sistemą „News Processor“, kuri yra ko gero labiausiai pritaikyta lietuviškoms spaudos įmonėms. Šiuo metu šią sistemą naudoja tokie leidiniai kaip dienraštis „Klaipėda“, „Šiaulių kraštas“, „Santarvė“ ir kiti.

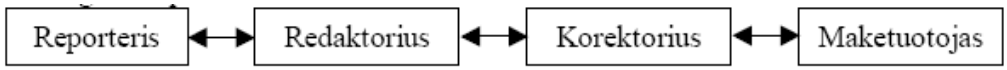
Deja, visose šiose sistemose nėra numatyta automatinio kategorizavimo galimybių, todėl šis darbas yra atliekamas rankiniu būdu.

5.2 Redakcinės sistemos darbų srautas

Kiekvienoje leidybinėje redakcijoje yra tam tikros pozicijos darbuotojai: reporteriai, redaktoriai, korektoriai, maketuotojai, vyriausias redaktorius. Gali būti dar kokių nors pozicijų, bet idealiausiu atveju turėtų būti penkių pozicijų darbuotojai.

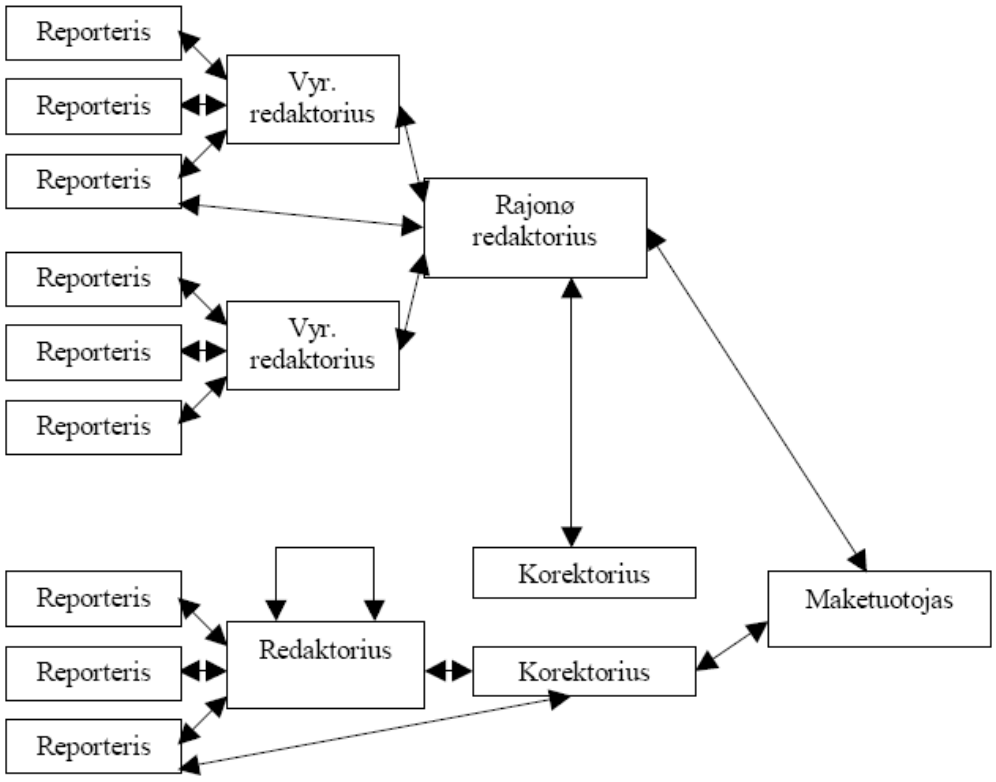
Taigi redakcinėje sistemoje reporteriai kuria publikacijas, jas siuntinėja redaktoriams. Redaktoriai – korektoriams arba tiesiai maketuotojams. Korektoriai – maketuotojams. Maketuotojai eksportuoja iš sistemos duomenų bazės publikacijų

tekstus į maketavimo programas suprantamo formato tekstus, kuriuos susikelia į būsimo laikraščio maketą. Galimas atgalinis ryšys: pvz. redaktorius grąžina publikaciją atgal reporteriui ir pan. Tokios sistemos schema pavaizduota pav. 5.1



Pav. 5.1 Supaprastintas redakcinės sistemos darbų srautas

Bet tikrovėje taip nebūna kaip pavaizduota Pav. 5.1. Dažnai grandinėlė nėra tokia graži, ji būna šakota (Pav. 5.2), turi susidvejinusių asmenybių (redaktorius-reporteris, korektorius-reporteris), ir nebūtinai turi būti "hierarchinė" priklausomybė kaip Pav. 5.1



Pav. 5.2 Realus redakcinės sistemos darbų srautas

5.3 News Processor redakcinė programa

Šios sistemos paskirtį galima aprašyti keliais punktais:

- Apjungti visos redakcijos sistemos darbą į vieningą sistemą leidžiančią kiekvienam darbuotojui nuo reporterio, korektoriaus, maketuotojo iki redaktoriaus visiškai atlikti darbus susijusius su leidinio gamybos procesu.
- Kurti, įkelti ir redaguoti, tikrinti ir skiemenuoti tekstus.
- Vienu metu ruošti kelius leidinius.
- Lengvai publikuoti leidinius internete.
- Kaupti redakcijos tekstų archyvą bei lengvai juo naudotis.
- Visapusiškai kontroliuoti leidinio ruošimo darbą, stebint publikacijų kūrimo eigą.
- Visiems redakcijos darbuotojams bendrauti tarpusavyje žinučių pagalba, redaktoriams skirti užduotis ir kontroliuoti jų vykdymą.
- Leisti bet kokiam redakcijos darbuotojui dirbti ne redakcijos patalpose, prisijungti prie sistemos per interneto tinklą.

News Processor paketą sudaro:

- NP PubEdit - tekstų redaktorius;
- NP DocExplorer - dokumentų tvarkymo sistema;
- NP Admin - NewsProcessor valdymo programa;

Tai gana universalus įrankis, bet skirtas išimtinai tik redakcijos darbui. Paketo galimybės labai plačios:

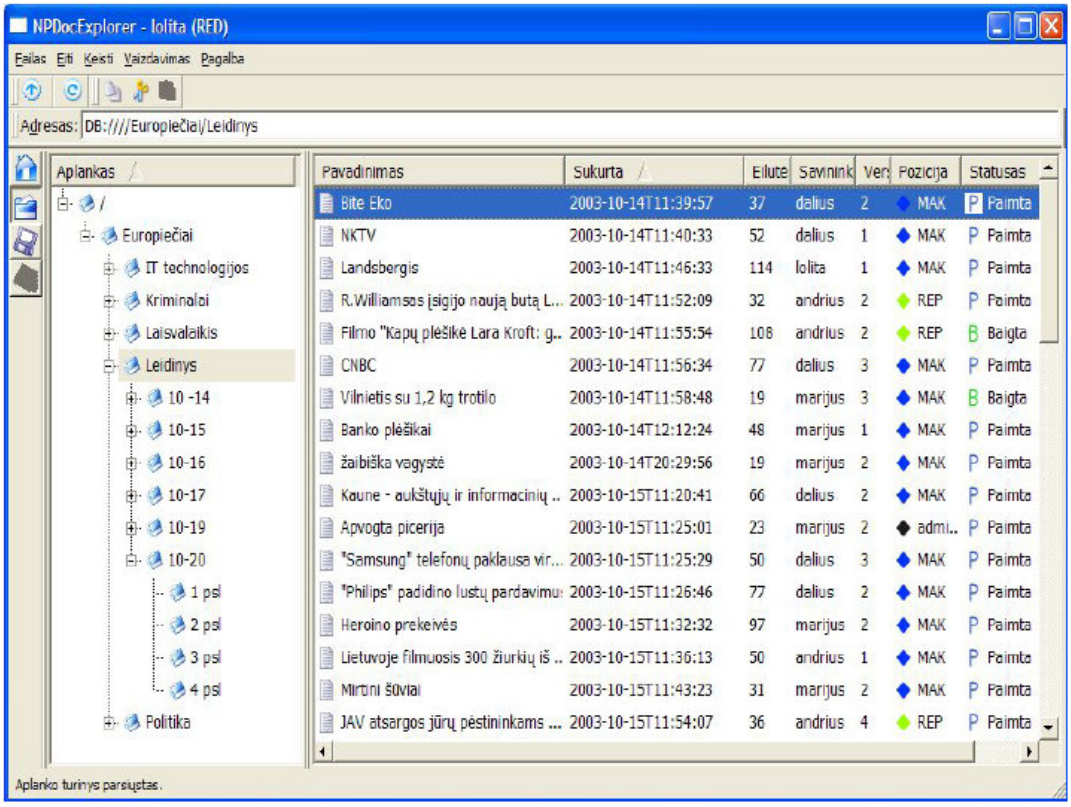
- NP programos veikia Windows, Mac ir Linux operacinėse sistemose.
- Sistema plečiama iki bet kokio dydžio redakcijos.
- Sistema greitina leidinio paruošimą, mažina klaidų kiekį leidinio ruošimo metu.
- Sistema iš karto formuoja lanksčią interneto svetainę.
- Lietuvių kalbos rašybos tikrinimas ir skiemenavimas.
- Daugiakalbės sąsajos.

- Reporterių programos nereikalauja naujausios technikos, veikia net su PI 150Mhz procesoriais.
- Kiekvienas reporteris ir redaktorius turi savo darbinę erdvę, asmeninę šiukšlių dėžę duomenims.
- Kiekvienas vartotojas turi savo žinučių dėžutę.
- Straipsniai turi versijas ir jų kontrolės mechanizmus.
- Lengvai valdomi ir konfigūruojami dokumentų srautai.
- Reporteriai gali dirbti ir tinkle, ir be tinklo.
- Lanksti vartotojų teisių sistema.
- Galimybė persijungti iš grupės į grupę, iš rolės į rolę.
- Katalogų medis paprastai navigacijai tarp duomenų.
- Administratoriaus nurodomos sisteminės dokumentų kategorijos.
- Lanksti dokumentų paieška.
- Greita paieška centrinėje duomenų bazėje esančiuose dokumentuose.
- Dokumentų rikiavimas ir filtravimas pagal bet kurį norimą parametrą.
- Pastabos straipsniams.
- Pilna straipsnių būsenų istorija.
- Serverio platforma - UNIX, PostgreSQL.
- Lanksti grupavimo ir rolių sistema (vartotojas grupėje, grupė grupėje).
- Prisijungimas prie serverio interneto tinklu.

Pagrindiniai šia programa atliekami darbai yra:

- Leidinių planavimas

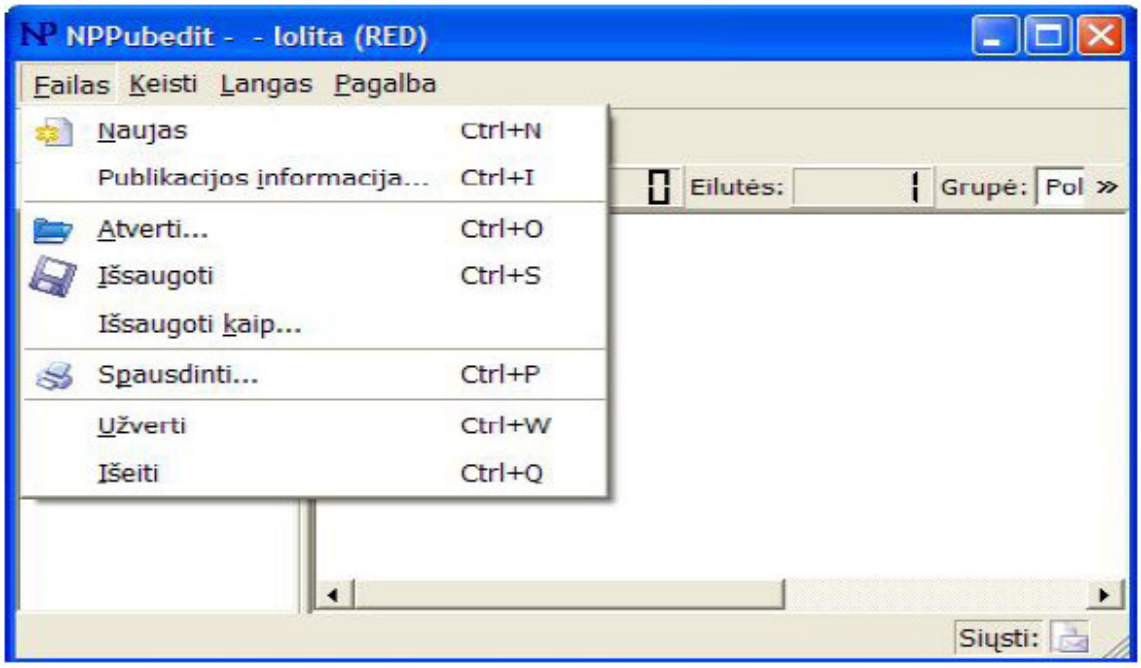
Su sistema NewsProcessor galima ruošti kelius leidinius. Tam atsakingas už leidinio planavimą darbuotojas su programa DocExplorer (Pav. 5.3) sukuria atitinkamus katalogus, kurių niekas neturi teisės ištrinti.



Pav. 5.3 Programos DocExplorer vartotojo sąsaja

- Publikacijų ruošimas

Publikacijas reporteriai kuria su PubEdit programa (Pav. 5.4). Programa galima dirbti tiek biuro tinkle, internetu, tiek neturint jokio ryšio su Centrine duomenų baze.

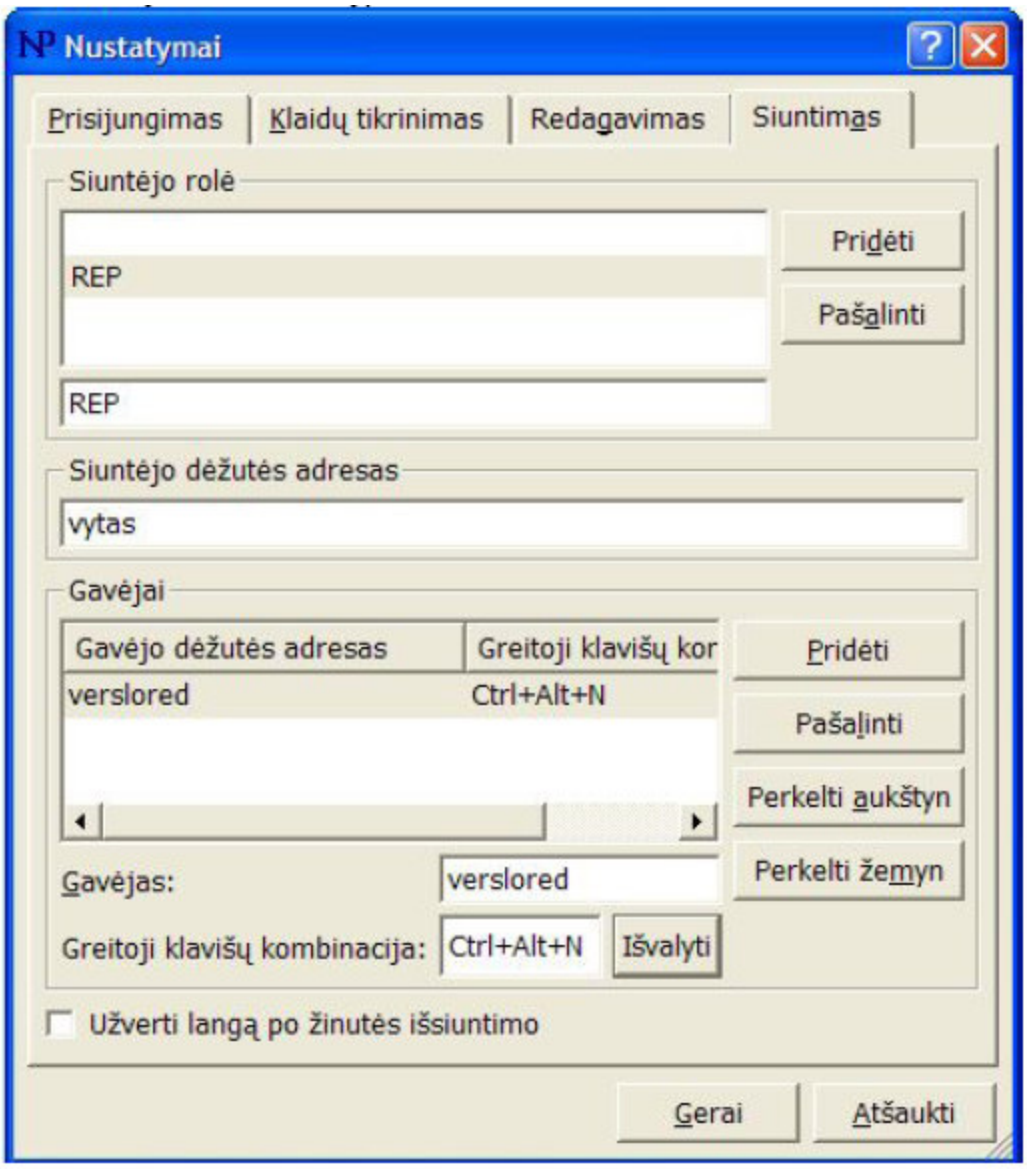


Pav. 5.4 PubEdit programos vartotojo sąsaja

Prisijungimo parametrai:

Norint dirbti tinkle ar internetu pirma reikia sukonfigūruoti PubEdit programą. Svarbiausia įvesti savo vardą sistemoje bei slaptažodį. PubEdit programa iš karto pasiūlys grupes, kuriose galite dirbti, bei roles, į kurias teises jums suteikė NP administratorius.

Pavyzdys (Pav. 5.5). Vytas yra reporteris Verslo grupėje ir redaktorius Politikos grupėje. Įvedęs PubEdit programoje prisijungimo parametrų lange “vytas “ bei savo slaptažodį, jis galės persijunginėti ir dirbti vienu metu redaktoriaus darbus (redaguoti kitų reporterių publikacijas), kitu metu – reporterio (kurti ir redaguoti tik savo publikacijas).



Pav. 5.5 Prisijungimas prie sistemos

- Publikacijos rašymas

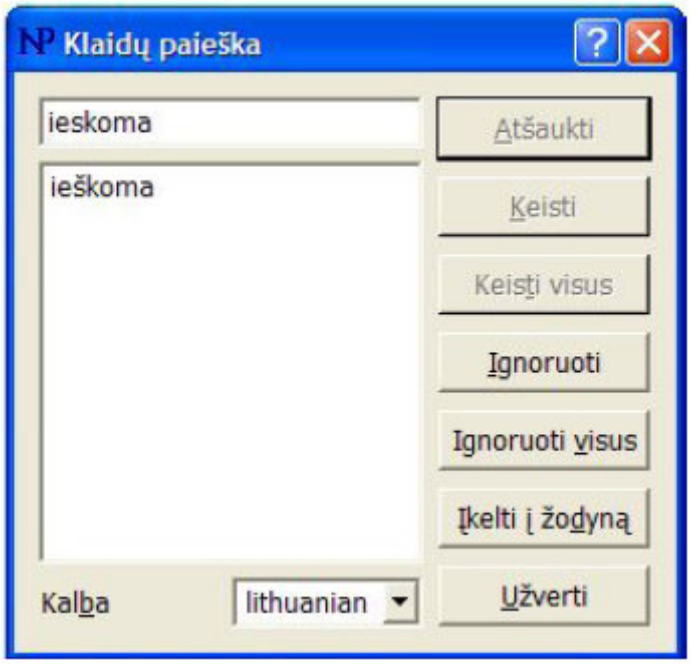
Rašyti publikaciją galima su bet kuriuo jūsų mėgstamu tekstų redaktoriumi. Tačiau įkelti publikaciją į Centrinę duomenų bazę, ir išsiųsti ją reikiamam darbuotojui galima tik su PubEdit programa prijungta prie interneto arba kito kompiuterių tinklo.

Publikaciją parašytą su bet kuriuo tekstų redaktoriumi ir bet kokiu kompiuteriu, iš bet kokio disko ar diskelio, galite įkelti komandomis Copy-Paste.

Parašytą ar įkeltą publikaciją BŪTINA sutvarkyti, aprašyti jos dalis (antraštė, tekstas, autorius, fotografas, žodžiai pagal kuriuos publikaciją bus galima greitai rasti ateityje ir kt.).

Publikacijos rašymo metu, galima matyti kiek simbolių yra joje, bei kiek apytiksliai spaudos eilučių ji užima.

PubEdit programa gali tikrinti įvairių kalbų gramatiką. Norėdami patikrinti savo parašytos publikacijos gramatines klaidas, turite pastatyti kursorių į publikacijos pradžią ir pasirinkti komandą *Ieškoti klaidų* (Pav. 5.6).



Pav. 5.6 Klaidų taisymas

Pabaigę publikaciją galite ją išsaugoti savo kompiuterio diske (sau) arba Centrinėje duomenų bazėje savo darbinėje erdvėje. Išsiųsti redaktoriui galima tik tas publikacijas, kurios yra išsaugotos Centrinėje duomenų bazėje.

Pabaigtą ir aprašytą publikaciją reikia išsiųsti savo redaktoriui. Redaktorius peržiūrėjęs publikaciją, galės arba ją siųsti stilistams, arba iš karto talpinti į galutiniam leidiniui paruoštą ir publikacijai skirtą katalogą.

Jei redaktorius nuspręs, kad publikaciją reikia tvarkyti, jis galės ją grąžinti autoriui. Autorius tokiu atveju gaus redaktoriaus pranešimą su komentaru.

Publikacijos būseną (grąžinta, paimta, atiduota maketavimui) visi darbuotojai taip pat gali matyti DocExplorer programoje.

Leidinių parengimui yra naudojama programa DocExplorer (Pav.5.3).

Skyriaus redaktorius, kuris yra atsakingas už atitinkamą skyrių, mato savo skyriaus darbuotojų publikacijas ir jų būsenas, gali išleidimui paruoštas publikacijas sukelti į reikiamos dienos reikiamą skyrių ir puslapį.

Skyriaus redaktorius mato visas jam reporterių atsiųstas publikacijas, sprendžia, ką su jomis daryti - spausdinti, atidėti, gražinti pataisymams ir kt. Vyr. redaktoriaus rolę turintis darbuotojas mato visas publikacijas bei skyrių redaktorių darbo eigą, gali komentuoti, ir pagal leidimus redaguoti publikacijas.

Kiekvienas vartotojas turi savo atskirą dėžutę saugoti publikacijoms – namai. Darbuotojas turintis tam tikras teises gali pasiekti kai kurių vartotojų namus, tačiau jis matys tik publikacijas be katalogų. Kai publikacijų labai daug, galima jas filtruoti pagal pavadinimą, datą, autorių ir kitas savybes apibūdinančias publikacijas. Užėję ant savybės dešiniuoju pelės klavišu pasirenkame filtravimą. Jei parinkti keli filtravimai, komanda vyks IR principu t.y. bus atrinktos publikacijos, kurioms būdingos abi filtruotos savybės.

Darbo langą galima koreguoti. Jei trukdo darbui tam tikros publikacijos savybės, jas galima pašalinti. Tereikia užeiti ant jų ir dešiniu pelės mygtuku nustatyti kokias savybes norite matyti. Jei nematote pilno savybės aprašymo, užeikite ant savybės ir du kartus paspauskite pelės kairįjį mygtuką.

Pagrindines komandas reikalingas dirbti su NP DocExplorer galite rasti pažymėję publikaciją ir spustelėję dešinįjį pelės mygtuką. Statuso sąrašas suteiks informaciją apie publikaciją t.y. kada ji buvo išsiųsta ir kokiam asmeniui, ar ji buvo taisyta, grąžinta ir pan.

Redaktorius ar atsakingas asmuo surinkęs visas publikacijas į bendrą katalogą gali jas eksportuoti, tarkim, tiesiai į internetinį puslapį.

Redaktorius planuoja leidinį t.y. kuria su NPDocExplorer katalogų hierarchiją maketuotojams ir ten dėlioja publikacijas. Labai svarbios yra dokumentų kategorijos, kurios yra naudojamos sisteminti NP sistemos dokumentus į tam tikras

temas – kategorijas. Dokumentas gali priklausyti nebūtinai vienai kategorijai. Jei dokumentų kategorijos yra naudojamos, tai labai palengvina dokumentų paieška.

Kategorizavimas yra atliekamas rankiniu būdu, todėl būtų labai naudinga šį darbą automatizuoti, taip palengvinant redaktoriaus darbą.

6. Eksperimentai ir rezultatai

Siekiant ištirti galimybę panaudoti dirbtinius neuronų tinklus redakcinių sistemų darbe, buvo atlikti keli eksperimentai, kurių metu buvo bandoma įvertinti kategorizavimo tikslumą, ryšį tarp raktinių žodžių skaičiaus ir sistemos galimybės atpažinti tekstinius dokumentus, bei palyginti keli skirtingi neuroniniai tinklai ir jų tinkamumas kategorizavimo uždaviniams spręsti. Tyrimus buvo atlikti SOM, LVQ ir tiesioginio sklidimo tinklais pasinaudojant Matlab paketu.

6.1 Duomenų surinkimas eksperimentams

Sprendžiant kategorizavimo uždavinį reikia turėti rinkinį tekstinių dokumentų, kuriuos galime panaudoti dirbtinių neuronų tinklų mokymui ir testavimui. Bandymams buvo parinkti du skirtingo dydžio tekstinių dokumentų rinkiniai. Naudojama informacija buvo imama iš lietuviškos elektroninės žiniasklaidos – Lietuvos rytas (www.lrytas.lt), Delfi (www.delfi.lt), Kompiuterija (www.kompiuterija.lt). Eksperimentams naudoti straipsniai buvo publikuoti nuo 2006 m. Vasario mėnesio iki 2006 m. Gegužės mėnesio.

Pirmąjį dokumentų rinkinį sudarė penki šimtai devyniasdešimt keturi straipsniai, suskirstyti į aštuonias kategorijas:

1. Auto – 21 straipsnis;
2. IT – 30 straipsnių;
3. Kultūra – 100 straipsnių;
4. Lietuvos diena – 100 straipsnių;
5. Pasaulis – 100 straipsnių;
6. Sportas – 100 straipsnių;

7. Sveikata – 43 straipsniai;
8. Verslas – 100 straipsnių;

Šiame etape buvo tiriamas neuroninio tinklo tikslumas, skirtingose kategorijose esant skirtingam straipsnių skaičiui.

Antrąjį dokumentų rinkinį sudarė tūkstantis du šimtai straipsnių, taip pat suskirstytų į aštuonias kategorijas:

1. Auto – 150 straipsnis;
2. IT – 150 straipsnių;
3. Kultūra – 150 straipsnių;
4. Lietuvos diena – 150 straipsnių;
5. Pasaulis – 150 straipsnių;
6. Sportas – 150 straipsnių;
7. Sveikata – 150 straipsniai;
8. Verslas – 150 straipsnių;

Šiame etape buvo tiriamas neuronų tinklų kategorizavimo tikslumas, visose kategorijose esant vienodam duomenų kiekiui.

6.2 Raktinių žodžių išskyrimas

Kad turimus straipsnius būtų galima suskirstyti į ankstesniame skyriuje aprašytas kategorijas, reikėjo parinkti kiekvieną kategoriją atitinkančius raktinius žodžius. Tam tikslui, pasinaudojant programa (1 PRIEDAS) buvo išskirti 230 dažniausiai visame dokumentų rinkinyje pasitaikančių žodžių. Kad būtų išvengta kaitomos galūnės įtakos, buvo imami tik žodžių kamienai. Analogiškai buvo išskirti atskirose kategorijose dažniausiai besikartojantys žodžiai. Palyginus pagrindinį raktinių žodžių rinkinį su visų kategorijų dažniausiai pasikartojančių žodžių rinkiniais, buvo išskirti du komplektai specifinių terminų, tinkančių kiekvienai kategorijai. Pirmąjame rinkinyje kiekvienai kategorijai buvo parinkta po aštuonis žodžius (6.1 lentelė), o antrąjame – po penkiolika (6.2 lentelė).

6.1 lentelė

Raktinių žodžių rinkinys N=8

Auto	IT	Kultūra	Lietuvos diena
Automobil	Kompiuter	Kino	Frakc
Moter	Internet	Dain	Piliet
Mašin	Sistem	Muzik	Valstieč
Model	Kompanij	Festival	Socialdemokrat
Draudim	Paslaug	Euroviz	Liberal
Gamyb	Vartot	Parod	Liaudinink
Transport	Microsoft	United	Dp
Lenktyn	Google	Meninink	Nato
Pasaulis	Sportas	Sveikata	Verslas
Polic	Rungtyn	Lig	Infliac
Pareigūn	Ekip	Sveikat	Rink
Pirmadien	Krepšin	Vaik	Valdant
Rinkim	Žaidėj	Maist	Pardavim
Deryb	Žalgir	Paukšč	Gryn
Šiaur	Lyg	Grip	Mlrd
Iran	Turnyr	Krauj	Apyvart
Karin	Pirmenyb	Medicin	Jukos

6.2 lentelė

Raktinių žodžių rinkinys N=15

Auto	IT	Kultūra	Lietuvos diena
Automobil	Kompiuter	Film	Seim
Kain	Program	Kino	Partij
Motor	Internet	Dain	Pirminink
Moter	Sistem	Teatr	Frakc
Amerik	Tarnyb	Kultūr	Piliet
Mašin	Duomen	Muzik	Demokrat
Model	Kompanij	Konkurs	Adamk
Amž	Paslaug	Festival	Bns
Parduot	Vartot	Euroviz	Valstieč
Draudim	Saugum	Praneš	Socialdemokrat
Duj	Parduot	Parod	Liberal
Doler	Doler	United	Liaudinink
Gamyb	Kinij	Žiūrov	Algird
Transport	Microsoft	Dram	Dp
Lenktyn	Google	Meninink	Nato
Pasaulis	Sportas	Sveikata	Verslas
Vyriausyb	Tašk	Žmog	Viktor
Įvyk	Komand	Žmon	Akcij
Įstatym	Rungtyn	Moter	Infliac
Lenkij	Pergal	Lig	Įmon
Polic	Ekip	Sveikat	Pajam
Pareigūn	Serij	Moksl	Bank
Pirmadien	Krepšin	Organizacij	Valdant
Rinkim	Žaidėj	Atvej	Pardavim
Valdž	Žalgir	Vaik	Gryn
Vokietij	Lyg	Maist	Naft
Deryb	Turnyr	Pavyzd	Mlrd
Šiaur	Pirmenyb	Paukšč	Apyvart
Iran	Futbol	Grip	Mažeik
Naujien	Rinktin	Krauj	Jukos
Karin	Titul	Medicin	Rink

Kadangi su šiais raktinių žodžių rinkiniais nei vienu neuronų tinklu nebuvo gautas pakankamas tikslumas, buvo perskaičiuota kiek procentaliai šie žodžiai kartojasi visose kategorijose ir atmesti tie, kurie nesudarė bent trisdešimties procentų vienoje kategorijoje. Liko septyniasdešimt trijų žodžių rinkinys – *automobil-, moter-, amerik-, mašin-, model-, draudim-, duj-, gamyb-, transport-, lenktyn-, kompiuter-, internet-, kompanij-, paslaug-, vartot-, doler-, kinij-, microsoft, google, film-, dain-, teatr-, kultūr-, muzik-, festival-, euroviz-, united, žiūrov-, dram-, meninink-, partij-, frakcij-, demokrat-, adamk-, valstieč-, socialdemokrat-, liberal-, liaudinink-, algird-, lenkij-, polic-, pareigūn-, rinkim-, valdž-, vokietij-, šiaur-, karin-, pergal-, ekip-, krepšin-, žaidėj-, žalgir-, turnyr-, pirmenyb-, futbol-, rinktin-, lig-, sveikat-, paukšč-, grip-, krauj-, medicin-, infliacij-, bank-, valdant-, pardavim-, gryn-, naft-, apyvart-, mažeik-, jukos, organizacij-, nato.*

Taigi gavome tris rinkinius neuronų tinklo apmokymui iš kurių sudarytos trys mokymo sekos:

1. Šešiasdešimt keturių žodžių;
2. Šimto dvidešimties žodžių;
3. Septyniasdešimt trijų žodžių;

Paskutinis rinkinys nuo pirmų dviejų skyriasi tuo, kad jame kiekvieną kategoriją atstovauja skirtingas raktinių žodžių skaičius.

6.3 Duomenų parengimas Matlab programai

Kad Matlab programiniu paketu sukurtume ir apmokytume dirbtinių neuronų tinklus, reikėjo sudaryti mokymo vektorius – dvejetaines skaičių sekas tinklo įėjimams ir tinklo tikslams (išskyrus SOM tinklą). Tam tikslui pasinaudojant programa (2 PRIEDAS) buvo sukurtas tekstinis failas, kurio kiekviena eilutė atitiko vieną straipsnį. Jei straipsnyje buvo randamas žodis iš raktinių žodžių sąrašo, tai eilutėje buvo įrašomas vienetas, jei nerandama – nulis (žiūr. skyrių 4.3). Tokiu būdu buvo gauti du įėjimo duomenų rinkiniai – penkių šimtų devyniasdešimt keturių

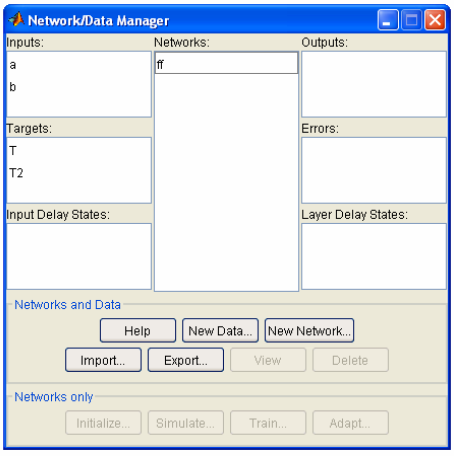
eilučių ir tūkstančio dviejų šimtų eilučių. Kadangi iš anksto paruoštų dokumentų pavadinimai atspindėjo jų priklausomybę vienai ar kitai kategorijai, tai ta pati programa pagal pavadinimus sukūrė tikslų (angl. targets) sekas abiems įėjimo duomenų rinkiniams. Matlab pakete naudojantis šiais duomenimis, dokumentai buvo tyriami SOM ir LVQ tinklais tiesiogiai ir tiesioginio sklidimo tinklais pasinaudojant Matlab paketo įrankiu nntools.

6.4 Rezultatai naudojant SOM tinklą

Patys prasčiausi rezultatai buvo gauti naudojant SOM tinklus (3 PRIEDAS). Kadangi duomenų kiekis buvo mažas, tinklas prastai įsiminė duomenų struktūrą ir su visais mokymo rinkiniais nepavyko bent kiek tiksliau sužymėti išėjimo sluoksnyje regionų. Be to, šis metodas reikalavo labai daug laiko tinklo apmokymui. Buvo išbandytas mokymo algoritmas su vienu šimtu ciklų, po to su vienu tūkstančiu ciklų ir galų gale su septyniais tūkstančiais penkiais šimtais ciklų. Nei vienu atveju mokymas nepasiteisino iš ko galima daryti išvada, kad pradinių duomenų kiekis šio tipo tinklams buvo nepakankamas.

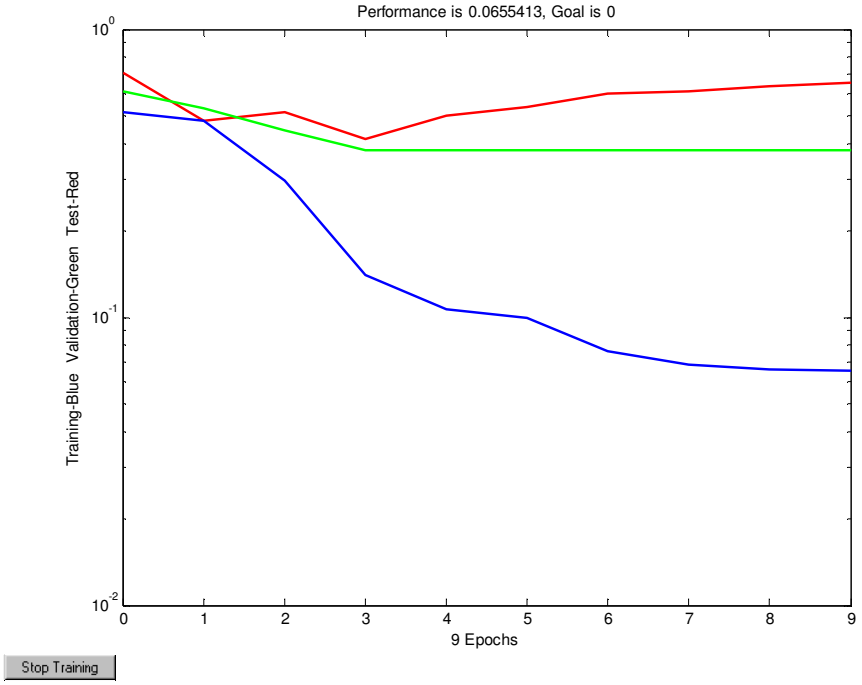
6.5 Rezultatai naudojant tiesioginio sklidimo tinklą

Tiesioginio sklidimo neuronų tinklas buvo modeliuojamas Matlab paketo įrankiu nntools (Pav. 6.1).



Pav 6.1 nntools įrankis neuroninių tinklų mokymui

Panaudojus šešiasdešimt keturių raktinių žodžių sekas su pirmuoju mokymo duomenų rinkiniu buvo gauti prasti patvirtinimo ir testavimo rezultatai (Pav. 6.2).



Pav. 6.2 MSE kitimas mokymo metu 64x594

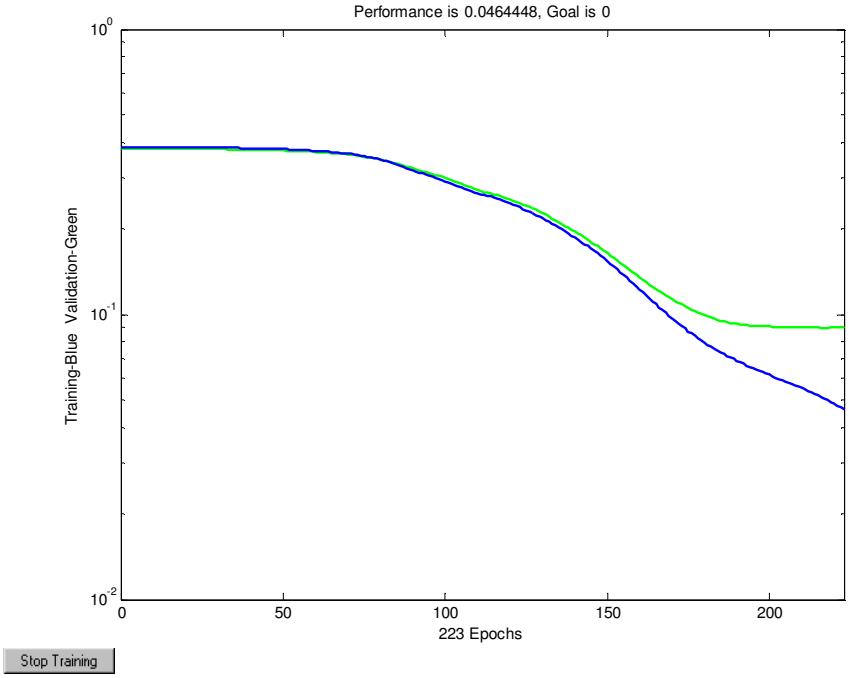
Tikrųjų reikšmių palyginimas su tinklo duotais rezultatais pateiktas 6.3 lentelėje.

6.3 lentelė

Patvirtinimo duomenų sutapimų lentelė 64x594

		Realios kategorijos							
		Auto	IT	Kultūra	Lietuva	Pasaulis	Sportas	Sveikata	Verslas
Tinklo reikšmės	Auto	3	5	10	8	13	21	7	14
	IT	7	4	7	8	15	9	5	23
	Kultūra	1	3	5	23	7	16	3	6
	Lietuva	1	6	20	7	25	14	0	20
	Pasaulis	6	5	31	8	0	18	4	18
	Sportas	3	0	12	21	31	10	3	10
	Sveikata	0	4	6	9	1	3	3	9
	Verslas	0	3	9	16	8	9	5	0

Bandymas buvo pakartotas su šimto dvidešimties raktinių žodžių sekomis naudojant tūkstančio dviejų šimtų dokumentų rinkinį. Šiuo atveju nebuvo naudojamas testavimas – tik mokymo ir patvirtinimo (angl. validation) funkcijos (Pav. 6.3).



Pav. 6.3 MSE kitimas mokymo metu 120x1200

Tikrųjų reikšmių palyginimas su tinklo duotais rezultatais pateiktas 6.4 lentelėje.

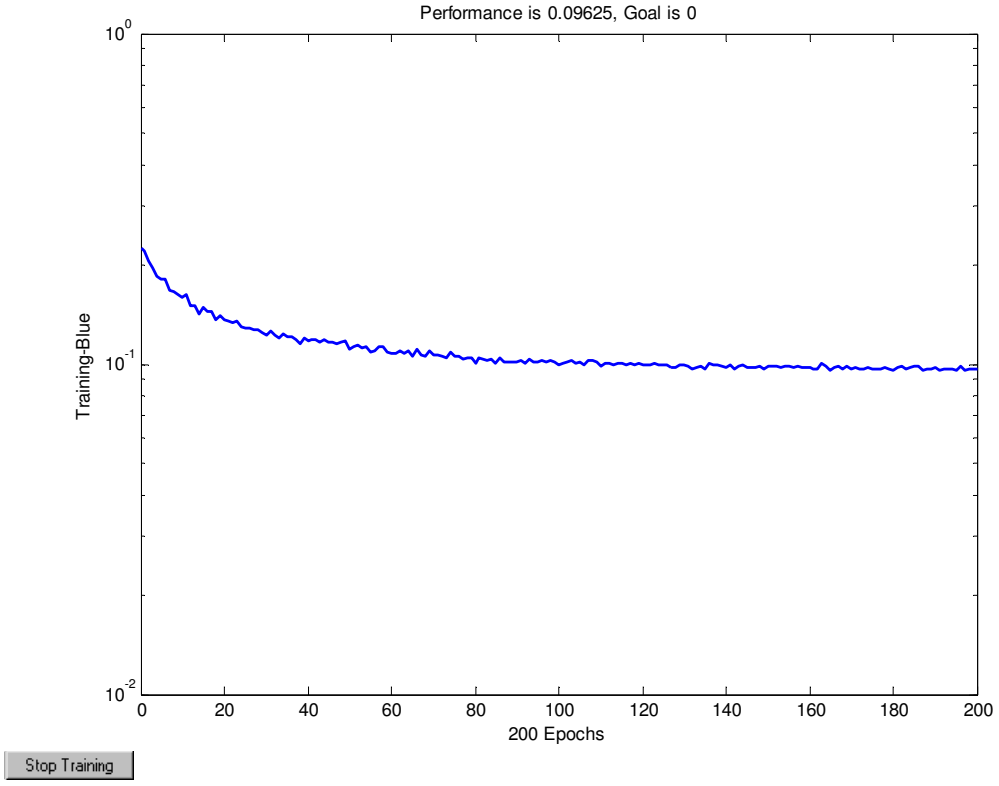
6.4 lentelė

Patvirtinimo duomenų sutapimų lentelė 120x1200

		Realios kategorijos							
		Auto	IT	Kultūra	Lietuva	Pasaulis	Sportas	Sveikata	Verslas
Tinklo reikšmės	Auto	72	14	13	11	20	12	11	30
	IT	10	69	11	15	24	15	20	21
	Kultūra	9	10	74	15	13	19	12	6
	Lietuva	18	16	18	70	12	10	9	18
	Pasaulis	12	15	12	14	10	11	8	18
	Sportas	4	9	10	11	25	68	1	25
	Sveikata	12	11	2	0	20	5	78	18
	Verslas	13	6	10	14	26	10	11	14

6.6 Rezultatai naudojant LVQ tinklą

Trečiasis bandytas neuronų tinklo modelis buvo LVQ tinklas (4 PRIEDAS). Pirmas atvejis – šešiasdešimt keturių raktinių žodžių rinkinys ir penki šimtai devyniasdešimt keturi tekstiniai dokumentai. Mokymo grafikas pateiktas Pav. 6.4



Pav. 6.4 MSE kitimas mokymo metu LVQ 64

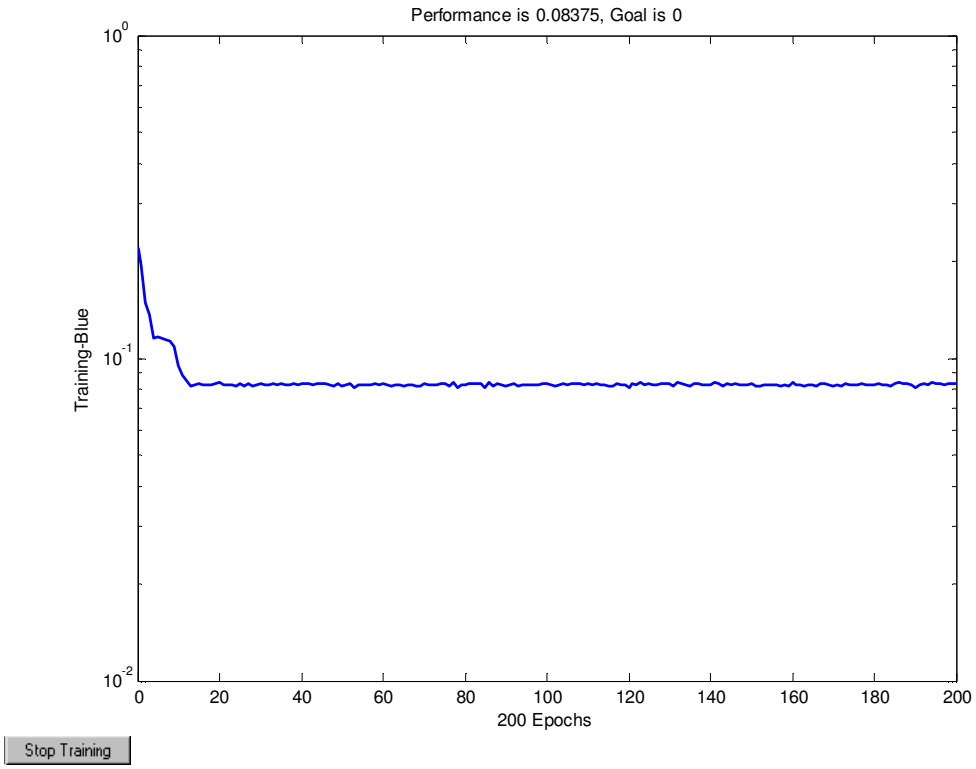
Tikrųjų reikšmių palyginimas su tinklo duotais rezultatais pateiktas 6.5 lentelėje.

6.5 lentelė

Patvirtinimo duomenų sutapimų lentelė LVQ 64

		Realios kategorijos							
		Auto	IT	Kultūra	Lietuva	Pasaulis	Sportas	Sveikata	Verslas
Tinklo reikšmės	Auto	30	9	2	5	3	3	8	4
	IT	11	20	3	4	4	6	5	5
	Kultūra	3	3	19	9	6	3	1	4
	Lietuva	4	5	8	21	5	12	6	10
	Pasaulis	0	2	7	4	12	7	2	4
	Sportas	6	9	11	10	15	29	6	8
	Sveikata	6	10	7	3	7	2	18	6
	Verslas	0	1	12	6	12	1	8	15

Antras bandymas – šimtas dvidešimt raktinių žodžių ir penki šimtai devyniasdešimt keturi tekstiniai dokumentai. Mokymo grafikas pateiktas Pav. 6.5.



Pav 6.5 MSE kitimas mokymo metu LVQ 120

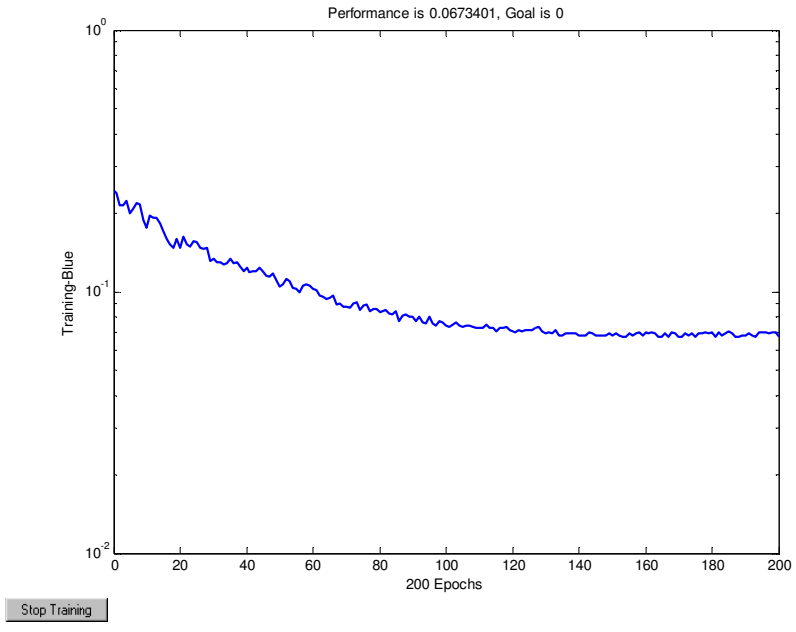
Tikrųjų reikšmių palyginimas su tinklo duotais rezultatais pateiktas 6.6 lentelėje.

6.6 lentelė

Patvirtinimo duomenų sutapimų lentelė LVQ 120

		Realios kategorijos							
		Auto	IT	Kultūra	Lietuva	Pasaulis	Sportas	Sveikata	Verslas
Tinklo reikšmės	Auto	38	10	1	3	3	5	13	8
	IT	13	26	5	4	3	3	11	6
	Kultūra	2	5	26	8	5	6	5	7
	Lietuva	12	11	9	26	14	5	4	15
	Pasaulis	1	4	14	12	14	18	11	8
	Sportas	9	6	3	4	10	26	6	13
	Sveikata	10	6	9	6	5	7	18	7
	Verslas	3	6	11	10	11	8	2	12

Trečiasis bandymas LVQ tinklu – septyniasdešimt trys perrinktos reikšmės ir tūkstantis du šimtai kategorizuojamų dokumentų. Mokymo grafikas pateiktas Pav. 6.6.



Pav. 6.6 MSE kitimas mokymo metu LVQ 73

Kaip matome iš grafiko – paklaidos šiuo atveju kur kas mažesnės už ankstesniuose bandymuose gautas paklaidas. Tačiau čia išryškėja kitas dalykas – tinklas labai gerai atpažino tik šešias iš aštuonių kategorijų, o dvi buvo visiškai klaidingai sukategorizuotos (6.7 lentelė).

6.7 lentelė

Patvirtinimo duomenų sutapimų lentelė LVQ 73

		Realios kategorijos							
		Auto	IT	Kultūra	Lietuva	Pasaulis	Sportas	Sveikata	Verslas
Tinklo reikšmės	Auto	71	1	0	1	7	2	2	10
	IT	0	76	0	1	14	0	3	14
	Kultūra	1	3	67	9	22	0	2	18
	Lietuva	1	1	1	50	8	0	1	5
	Pasaulis	0	0	0	0	0	0	0	0
	Sportas	0	0	0	3	13	73	0	15
	Sveikata	2	5	3	8	13	1	62	11
	Verslas	0	0	0	0	0	0	0	0

6.7 Gautų rezultatų įvertinimas

Remiantis gautais duomenimis galime daryti išvada, kad SOM tinklas su turimu duomenų kiekių yra netinkamas sprendimas. Geriausius rezultatus davė LVQ tinklas, bet iš jo išeitinių duomenų galime padaryti išvadą, kad ne visos iš anksto parinktos kategorijos buvo visiškai savarankiškos. Kadangi esant tikslesniam duomenų pateikimui (septyniasdešimt trijų raktinių žodžių variantas) kategorijos „Pasaulis“ ir „Verslas“ buvo kategorizuojamos klaidingai, galime spręsti, kad jos tiesiog yra kitų kategorijų subkategorijos arba jose esantys straipsniai apima labai plačią sritį. Antrasis variantas labiau tikėtinas, nes tinklas šių kategorijų straipsnius išskirstė kitose kategorijose beveik tolygiai.

IŠVADOS

Baigiamojo magistro darbo tikslas buvo išanalizuoti tekstinės informacijos kategorizavimo metodus pasinaudojant dirbtiniais neuronų tinklais ir ištirti jų galimybę panaudoti visa tai redakcinės sistemos kategorizavimo sistemos automatizavimui.

Gauti rezultatai parodė, kad skirtingomis aplinkybėmis įvairūs metodai nevienodai gerai atlieka kategorizavimo uždavinius. SOM tinklai šiam tikslui būtų tinkami tik iš anksto turint realiai veikiančią redakcinę sistemą ir didelį straipsnių archyvą. Tuomet tereikėtų sukurti papildomą automatinio kategorizavimo modulį tai sistemai. Tokias išvadas galima padaryti remiantis analizuota literatūra, nes šiame darbe praktiškai SOM tinklų išbandyti nepavyko.

Norint dirbtinius neuronų tinklus naudoti praktikoje reikėtų iš anksto labai tiksliai išskirti kategorijas panaudojant kuo didesnį kiekį pradinių duomenų, nes šiame darbe didinant pradinių tekstų skaičių rezultatai po truputį gerėjo tiek naudojant LVQ, tiek ir tiesioginio sklidimo tinklus. Be to kiekvienai žinisklaidos priemonei reikėtų kurti individualų produktą, atsižvelgiant į jos specifiką, nes geriausiai buvo sukategorizuotos temos, kurių straipsniai buvo paimti iš vieno šaltinio, o temos, kurioms straipsniai buvo renkami iš kelių šaltinių buvo atpažystamos kur kas prasčiau.

Naudota literatūra

[1] Fabrizio Sebastiani. Machine learning in automated text categorization. ACM Computing Surveys, 34(1):1-47, 2002

[2] Karen Sparck Jones and Peter Willett, editors. Readings in information retrievel. Morgan Kaufmann, San Mateo, US, 1997

[3] Yiming Yang. An evaluation of ststistical approaches to text categorization. Journal of information retrieval. 1-2(1): 69-90, 1999

[4] Thorsten Joachims. Text categorization with Support vector mashines: Learning with many relevant features European conference of Machines learning (ECML), 1998

[5] V. Vapnik. The Nature of Statistical Leaerning Theory. Springer. New York. 1995

[6] Y.Yang and C. G. Chute. An example-based mapping method for text categorization and retrieval. ACM Transaction on Information Systems (TOIS), 12(3):252-277, 1994

[7] Tom Mitchell. Machine learning. McGraw Hill. 1996

[8] K.W. Church. One term or two? In proceedings of the 18th Annual International ACM SIGIR Conference on Researchand Development in Information Retrieval, pages 310-318, Seattle, WA USA, 1995

[9] Nikolaos Nanas, Victoria Uren, Anne De Roeck. A Comparative Study of Term Weighting Methods for Information Filtering. Milton Keynes, UK. 2003

[10] H. Ritter, T. Kohonen. Self-organizing semantic maps.Biological Cybernetics. 61:241-254, 1989

PROGRAMOS RAKTINIŲ ŽODŽIŲ IŠRINKIMUI TEKSTAS

```
/*
 * Analyze.java
 *
 * Created on Antradienis, 2006, Gegиюлs 16, 15.06
 *
 */
package zodzioskaiciuokle;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Comparator;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import java.util.SortedMap;
import java.util.TreeMap;
/**
 *
 * @author Arunas
 */
public class Analyze {
    String inFile = "in.txt";
    String outFile = "out.txt";
    /** Creates a new instance of Analyze */
    public Analyze() {
    }
    private String readFromFile(File file) {
        long time1 = System.currentTimeMillis();
        StringBuffer sb = new StringBuffer();
        try {
            BufferedReader bufReader = new BufferedReader(new FileReader(file));
            String strLine = bufReader.readLine();
            while (strLine != null) {
                sb.append(strLine).append("\r\n");
                strLine = bufReader.readLine();
            }
            bufReader.close();
            long time2 = System.currentTimeMillis();
            System.out.print("Failo nuskaitymo laikas (ms): ");
            System.out.println(time2 - time1);
            //return sb.toString();
            return new String(sb.toString());
        } catch (IOException e) {
            System.out.println("IO klaida:" + e);
```

```
        }
        return null;
    }
    private void saveToFile(File file, String context, boolean append){
        BufferedWriter out = null;
        try {
            out = new BufferedWriter(new FileWriter(file, append));
            out.write(context);
            out.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
    public void analyzeText(){
        SortedMap map1 = new TreeMap();
        String a = this.readFromFile(new File(this.inFile));
        a = a.replaceAll("(?i)[,./\",,”“;:'`~!@()#$%–^&?!*+=-]", "");
        a = a.replaceAll("\n","");
        a = a.toLowerCase();
        String[] abc = a.split("\\s");
        int kk = 1;
        for (int i = 0; i < abc.length; i++) {
            if( map1.containsKey(abc[i]) == true ){
                int b = Integer.parseInt( map1.get(abc[i]).toString() );
                map1.put(abc[i], ++b);
            } else {
                map1.put(abc[i], kk);
            }
        }
        String save = "";
        Set entries = map1.entrySet();
        Iterator iterator = entries.iterator();
        int i = 0;
        while (iterator.hasNext()) {
            Map.Entry entry = (Map.Entry)iterator.next();
            save += entry.getKey()+"\t"+entry.getValue()+"\n";
            System.out.println(entry.getKey()+"\t"+entry.getValue()+"");
        }
        this.saveToFile(new File(this.outFile), save, false );
        System.gc();
    }
}
```

PROGRAMOS ĮĖJIMO DUOMENIMS PARUOŠTI TEKSTAS

```
<?php
function getFolderArray(){
	$failu_araray = array();
	$handele = opendir('./tekstai');
	if ($handele){
		while (false != ($file = readdir($handele))) {
			if ($file != '.' && $file != '..'){
				$failu_araray[]= $file;
			}
		}
	}
	return $failu_araray;
}
$handler = fopen('./kamienai.txt','r');
$kamienai = fread($handler,filesize('./kamienai.txt'));
fclose($handler);
$kamienu_masyvas = explode('|',$kamienai);
$fail_masyvas = getFolderArray();
$dvejetainis_kodas= array();
foreach ($fail_masyvas as $failas){
   foreach ($kamienu_masyvas as $kamienas){
   $dvejetainis_kodas[$failas] .= ' '.getdvejetainis($failas,$kamienas);
		}
}
function getdvejetainis($failas,$kamienas){
$handler = fopen('./tekstai/'.$failas,'r');
$failo_turinys = fread($handler,filesize('./tekstai/'.$failas));
fclose($handler);
		if (stristr($failo_turinys,$kamienas)) return 1;
	else return 0;
}
$failo_turinys = '';
foreach ($dvejetainis_kodas as $kodas){
$failo_turinys .= $kodas.'
';
}
$handler = fopen('./rezultatai.txt','w');
fwrite($handler,$failo_turinys);
fclose($handler);
print '<pre>';
print_r($dvejetainis_kodas);
print_r($kamienu_masyvas);
print '</pre>';
?>
```

SOM TINKLO MOKYMO PROGRAMOS TEKSTAS

```
[file, path]=uigetfile('*.txt', 'text file')    %Pasirenkamas failas
a=load([path file]);                            %Perskaitomi duomenys is failo
n=size(a,1);                                    %Nustatomas eiluciu skaicius duomenu masyve
net = newsom([zeros(64,1) ones(64,1)],[8 8], 'gridtop');  %Sukuriamas SOM tinklas
net.trainParam.epochs = 1000;                   %Mokymo ciklu skaicius
net.trainParam.show=1;                          %Kas kiek ciklu rodomi rezultatai
net=train(net,a');                              %Tinklo mokymas
Tp = sim(net,a');                               %Nustatoma, koks taskas i kuri klasteri pateko
T= vec2ind(Tp)                                  %Rezultatai is masyvo pakeiciami i indeksu vektoriu
plot(T, '*');
```

LVQ TINKLO MOKYMO PROGRAMOS TEKSTAS

```
a=xlsread('vektoriai_nauji.xls');                %Nuskaitomas mokymo rinkinio failas
NN=size(a,1);                                    %Nustatomas eiluciu skaicius duomenu masyve
N=round(NN/2);                                   %Failas padalinamas i dvi dalis mokymui ir patvirtinimui
T1=xlsread('vektoriu_kategorijos_n.xls');        %Nuskaitomas "taikiniu" failas
T1=T1(2:1201,3)';                                % Nustatomos duomenu nustatymo ribos
rr=rand(1, NN);                                  %ismaisomos reiksmes
[r, index]=sort(rr);
a=a(index,:);
T1=T1(index);
T = ind2vec(T1)
net = newlvq([zeros(73,1) ones(73,1)], 900, [ones(1,8)*0.125]); %Sukuriamas LVQ neuronu tinklas
net.trainParam.epochs = 200;                     %Mokymo ciklu skaicius
net.trainParam.show=1;                           %Kas kiek ciklu rodomi rezultatai
P=a(1:N, :)';
Tt=T(1:8,1:N);
VV.P=a(N+1:2*N, :)';
VV.T=T(1:8,N+1:2*N);
net = train(net, P, Tt,[],[],VV);       %Tinklo mokymas
Tp=sim(net, P);
Tr = vec2ind(Tp)
Tp1 = vec2ind(Tt)
figure                %Atvaizduojamas mokymo rinkinio atitikimas "taikiniams"
plot(Tp1, 'r*')
hold on
plot(Tr, 'bo')

TP=sim(net, VV.P);
TR = vec2ind(TP)
TP1 = vec2ind(VV.T)
figure                          %Atvaizduojamas patvirtinimo rinkinio atitikimas "taikiniams"
plot(TP1, 'r*')
hold on
plot(TR, 'bo')
```