VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis magistro darbas

Turinio filtras, paremtas daugialypės terpės failų klasifikavimu

Atliko: 2 kurso, 9 grupės studentas
Edgaras Mečkauskas (parašas)
Darbo vadovas:
dr. Linas Bukauskas (parašas)
Recenzentas:
..... (parašas)

Vilnius
2012

Turinio filtras, paremtas daugialypės terpės failų klasifikavimu

Santrauka

Šiame darbe pasiūlytas algoritmas, gebantis spręsti dviejų klasių problemą bei pasitelkiant tik tekstinį turinį skirtas analizuoti ir klasifikuoti tokius daugialypės terpės dokumentus kaip *HTML* puslapiai. Taip pat *Mozilla Firefox* įskiepio pagrindu sukurtas turinio filtras, klasifikuojantis pagal darbe pasiūlytą klasifikavimo algoritmą. Klasifikatoriui apmokyti pasitelktas *PHP* programavimo kalba realizuotas tiesinis atraminių vektorių algoritmas (SVM). Pagrindinės realizuoto turinio filtro savybės, išskiriančios jį iš daugumos rinkoje esančių analogiškų įrankių, tai galimybė klasifikuoti dar algoritmui nežinomus interneto puslapius bei priklausomai nuo parametrų blokuoti tik dalį arba visą internetinį dokumentą.

Raktiniai žodžiai : *HTML* dokumentas, daugialypė terpė, požymių išskyrimas, požymių atranka, dokumentų indeksavimas, internetinio turinio klasifikavimas, tiesinis atraminių vektorių klasifikatorius, turinio filtras.

Content filter based on classification of multimedia documents

Abstract

An algorithm, able to solve two class problem, designed to analyse and classify multimedia documents such as *HTML* pages by using textual content, is suggested in the paper. Moreover, content filter based on *Mozilla Firefox* extension was developed to classify web pages according to the algorithm. Linear support vector machine (SVM) was developed using *PHP* programming language in order to train the classifier. The main advantage of the content filter we developed which distinguishes it from other analogical tools existing in the market is its ability to classify web pages unfamiliar to the algorithm and to block a part or entire web document depending on setup.

Key words : *HTML* document, multimedia, feature extraction, feature selection, document indexing, web content classification, linear support vector machine, content filter.

Turinys

Įvadas

Šiais laikais internetas suteikė žmonėms galimybę prieiti prie daug daugiau informacijos nei tai buvo galima padaryti kada nors anksčiau ir tapo dažniausiai vartojamu informacijos šaltiniu. Tačiau virtuali erdvė atskleidė ir savo tamsiąją pusę. Dėl mažos kontrolės ir internetinio turinio reguliavimo daugybė nepageidaujamos informacijos yra lengvai prieinama internete. Tarp milijonų naudingų internetinių puslapių egzistuoja daugiau nei 500 000 internetinių puslapių platinančių pornografinį ar kitą vaikams žalingą turinį [48]. Todėl kuo daugiau asmenų, tame tarpe ir nepilnamečių, naudojasi internetu bei kuo daugiau informacijos yra pateikiama internete, tuo didesnė tikimybė, kad vartotojas susidurs su nepadoria, tik suaugusiems asmenims skirta, informacija. Ši informacija ypatingai žalinga vaikams ir paaugliams, todėl nereikia nė sakyti, kad yra būtina turėti galimybę apriboti priėjimą prie tokią informaciją platinančių internetinių portalų.

Šiuo metu rinkoje egzistuoja ne vienas įrankis, leidžiantis blokuoti suaugusiems skirtus internetinius puslapius, pavyzdžiui *Cyber Patrol, squidGuard, Websense, Net Nanny, FoxFilter, ProCon Latte* ir kt. Šie įrankiai nepilnamečių tėvams suteikia galimybę apsaugoti savo atžalas nuo nepadoraus turinio. Dauguma šių ir panašių apsaugos sprendimų veikimo principas pagrįstas raktažodžių, reguliariųjų išraiškų paieška, nuorodų ar *IP* adresų filtravimu, iš anksto paruoštais lyginamųjų nuotraukų archyvais ir t.t.

Nepaisant to, kad šie plačiai paplitę ir paprasti metodai puikiai veikia ir šiais laikais, nereikia nė abejoti, kad tokios priežastys, kaip tekstinio turinio puslapyje stoka (pvz. tik iš nuotraukų sudaryti puslapiai), tekstinių iškarpų apjungimas (pvz., paieškos variklių atliekama puslapių turinio agregacija, dėl kurios į saugų turinį gali įsimaišyti ir pornografinio turinio nuotraukos ar kita daugialypės terpės informacija, pvz., audio, video ir t.t.), sąmoningas tekstinio turinio iškraipymas (pvz., raidės pakeistos skaičiais ar pan.) arba pateikimas daugialypės terpės failų pavidalu (pvz., paveiksliukuose) ir t.t., gali apgauti tokias sistemas ir iškreipti klasifikavimo rezultatus.

Šiame straipsnyje aptarsime algoritmą, pagrįstą daugialypės terpės failų klasifikavimu ir skirtą aptikti nepageidaujamą turinį *HTML* formatu saugomuose internetiniuose puslapiuose. Priešingai nei dauguma šiuo metu paplitusių apsaugos priemonių algoritmas nereikalautų nuolat kiekviename kompiuteryje rankiniu būdu atnaujinti duomenų bazės, kurioje būtų saugomi adresai, nurodantys nepageidaujamus puslapius, raktiniai žodžiai, nuotraukos ar kita būtina informacija, ir dėl šios priežasties galėtų klasifikuoti dar nežinomus tinklalapius. Taip pat skirtingai nuo alternatyvių įrankių algoritmas leistų blokuoti ir įtartinas dokumento

dalis, o ne tik visą puslapį. Be to, yra siekiama, kad algoritmas būtų gana efektyvus bei pakankamai paprastas, kad būtų galima realizuoti tokiose riboto funkcionalumo aplinkose, kaip pvz. internetinių naršyklių įskiepiai.

1. Tekstinės informacijos klasifikavimas

Šiame darbe pirmiausia apibrėšime, kas tai yra tekstinės informacijos klasifikavimas, bei trumpai apžvelgsime su klasifikavimu susijusius procesus. Tuomet detaliau panagrinėsime du dažniausiai naudojamus automatinio klasifikavimo metodus, įskaitant duomenų indeksavimą, klasifikatoriaus sudarymą ir apmokymą bei kokybės vertinimą. Galiausiai apžvelgsime konkrečias problemas, susijusias su HTML dokumentų struktūrinėmis savybėmis ir klasifikavimu, bei pasiūlysime internetiniams puslapiams klasifikuoti skirtą algoritmą ir aptarsime realizuotą sistemą, jos savybes, privalumus bei klasifikavimo rezultatus.

1.1. Įvadas į teksto klasifikavimą

Tekstinės informacijos klasifikavimą galime apibūdinti kaip automatinį įvesties teksto priskyrimą kuriai nors kategorijai. Kategorijos gali būti apibrėžtos iš anksto (klasifikavimas) arba nustatytos automatiškai (grupavimas). Formaliai klasių aibę galime apibrėžti kaip $C = \{c_1, ..., c_{|C|}\}$, o duomenų aibę $D = \{d_1, ..., d_{|D|}\}$. Teksto kategorizavimas gali būti apibrėžtas kaip klasifikavimo funkcijos f radimas, kuri priskiria loginę reikšmę kiekvienai porai $(d_j, c_i) \in D \times C$, reikšmę $True$ nusakančią, kad duomenų pavyzdys d_j turėtų būti priskirtas kategorijai c_i, arba reikšmę $False$ – kad duomenys d_j nepriklauso kategorijai c_i. Funkcija f yra vadinama klasifikatoriumi.

Teksto klasifikavimas – tai gana subjektyvus procesas, kadangi keli skirtingi asmenys, klasifikuotojai (tuo labiau automatizuoti klasifikatoriai), gali tuos pačius dokumentus priskirti skirtingoms kategorijoms arba išvis nė vienai.

Priklausomai nuo situacijos teksto klasifikavimas gali būti skirstomas į potipius. Pavyzdžiui, klasifikavimas gali būti vienareikšmis arba daugiareikšmis, griežtas arba negriežtas, vienmatis arba hierarchinis ir t.t.

1.1.1. Vienareikšmis ir daugiareikšmis klasifikavimas

Vienareikšmio klasifikavimo atveju tik vienai kategorijai c_i gali būti priskirtas dokumentas d_j. Daugiareikšmio klasifikavimo atveju dokumentas d_j gali būti priskirtas bet kurioms kelioms kategorijoms iš aibės $C = \{c_1, ..., c_{|C|}\}$ [41]. Galima išskirti dažnai naudojamą vienos reikšmės kategorizavimo būdą – tai binarinis teksto klasifikavimas, kai dokumentas d_j gali būti priskirtas arba kategorijai c_i, arba jai priešingai kategorijai c_i'. Pats paprasčiausias binarinio klasifikavimo atvejo pavyzdys – tai teksto filtravimas, kurio metu priklausomai nuo temos dokumentai skirstomi į susijusius ir nesusijusius.

1.1.2. Griežtas ir negriežtas klasifikavimas

Griežtas klasifikavimas – tai konkrečios vienos ar kelių kategorijų priskyrimas konkrečiam klasifikuojamam dokumentui. Priešingai nei griežtu atveju negriežtas klasifikavimas vietoj to, kad priskirtų dokumentą prie konkrečios klasės ar klasių, potencialias klases sureitinguoja priklausomai nuo to, kiek kuri klasė yra labiau tikėtina [41]. Yra galimi du reitingavimo būdai: reitingavimas pagal kategoriją ir pagal duomenis. Kategorinis reitingavimas klases surikiuoja atsižvelgdamas į tikimybę, kad dokumentas priklauso konkrečiai klasei. Tokiu būdu, skirtingai nuo griežto klasifikavimo, gauname daugiau informacijos apie tai, kokia tikimybė, kad dokumentas priklauso vienai ar kitai klasei, bei iš surikiuoto klasių sąrašo galime patys nuspręsti, kuriai klasei labiausiai tikėtina dokumentas priklauso.

Kategorinis reitingavimas plačiai naudojamas automatiniame dokumentų indeksavimo procese, o reitingavimas pagal duomenis – dokumentų filtravime, kai reikia iš dokumentų rinkinio išrinkti labiausiai su konkrečia sritimi (kategorija) susijusius dokumentus.

1.1.3. Vienmatis ir hierarchinis klasifikavimas

Atsižvelgiant į klasių aibės pobūdį, tekstinį klasifikavimą galima suskirstyti į nehierarchinį (vienmatį) ir hierarchinį. Pirmuoju paprastuoju atveju visos kategorijos traktuojamos kaip nepriklausomos viena nuo kitos, tuo tarpu hierarchinio klasifikavimo atveju klasės tarpusavyję priklausomai nuo semantinių tarpusavio ryšių sudaro klasių medį [43].

Hierarchinį klasifikavimo uždavinį galima spręsti dviem būdais. Pirmuoju atveju kategorijų medis transformuojamas į vienmatę struktūrą iš medžio paėmus tik lapines klases, tokiu būdų suformuojamas daugiareikšmis klasifikavimo uždavinys. Pastaruoju atveju užtenka sudaryti tik vieną klasifikatorių, kuris nuspręstų, kurioms kategorijoms priskirti dokumentus. Galiausiai po šio klasifikavimo etapo yra būtinas papildomas apdorojimo metodas, kurio metu dokumentams priskiriamos aukštesnių hierarchijos lygmenų kategorijos. Antruoju atveju kiekviename klasių hierarchijos lygyje yra sukuriama po atskirą klasifikatorių. Kiekviename lygyje klasifikatorius parenka vieną ar kelias labiausiai tikėtinas kategorijas ir leidžiasi į žemesnius lygius, kuriuose jau kiti nauji klasifikatoriai nusprendžia, į kuriuos vaikinius mazgus bus leidžiamasi toliau link galutinių kategorijų. Tyrimais įrodyta, kad tokio pobūdžio hierarchinis klasifikavimas yra pranašesnis nei nehierarchinis [24, 23].

2. Teksto klasifikavimo etapai

Kiekvieną tekstinio klasifikavimo procesą galime suskirstyti į tris pagrindinius žingsnius [41]:

1. Dokumentų indeksavimas, transformavimas į klasifikatoriui priimtiną formą
2. Klasifikatoriaus apmokymas
3. Klasifikatoriaus darbo kokybės įvertinimas

Toliau plačiau aptarsime kiekvieną šių etapų.

2.1. Dokumentų indeksavimas

Neapdorota tekstinė informacija – tai tik paprasti, nestruktūrizuoti duomenys. Tekstiniai dokumentai tarpusavyje skiriasi ne tik juose naudojamų žodžių aibe, bet ir jų kiekiu. Dėl šių paprastų priežasčių jie dar nėra tinkamai paruošti apdoroti pasitelkiant įvairius klasifikavimo algoritmus. Todėl prieš pritaikant bet kokį klasifikavimo algoritmą privaloma apdoroti tekstinę informaciją ir transformuoti ją į priimtiną, struktūrizuotą formą. Dažniausiai tekstinei transformacijai atlikti naudojamas būdas – tai žodžių krepšelio (angl. bag of words) metodas. Šio metodo esmė yra transformuoti tekstinę įvesties informaciją (t.y. dokumentą) į termų svorių vektorių, kur termais gali būti žodžiai, frazės arba kita informacija apie dokumentą. Šis informacijos apdorojimo procesas sudarytas iš kelių dalių: termų išgavimo, apdorojimo bei jų selektyvaus parinkimo.

2.1.1. Termas

Egzistuoja keletas galimų būdų kaip apibūdinti termo savoką: leksiškai, semantiškai, sintaksiškai arba statistiškai. Iš visų būdų pats paprasčiausias ir dažniausiai naudojamas metodas – tai termai, pagrįsti individualiais žodžiais, o tekstinis dokumentas atvaizduojamas kaip atskirų žodžių vektorius. Be to, remiantis informacijos paieškos tyrimais termams apibrėžti rekomenduojama naudoti ne žodžius, bet žodžių šaknis [21], nors tai ir nevisada naudinga teksto kategorizavime, nes daugybė termų gali būti susieti į vieną tokiu būdu prarandant dalį svarbios informacijos [20]. Šis elementarus termo apibrėžimo būdas visiškai ignoruoja struktūrinius ryšius tarp žodžių, tokius kaip priklausomybė ar santykinė padėtis vienas nuo kito. Siekiant išspręsti šią problemą buvo atliktas ne vienas bandymas vietoj atskirų žodžių panaudojant frazes. Frazės gali būti sintaksinės arba statistinės. Sudarant

sintaksines frazes atsižvelgiama į sintaksinius sakinio suvaržymus ir struktūras, o statistines – į statistinius įvykius.

Sintaksiškai frazes galime apibrėžti kaip dviejų ar daugiau sintaksiškai sujungtų žodžių grupę, pvz. veiksmažodinės ar daiktavardinės frazės. D. Lewis [28] tyrinėjo sintaksinių frazių panaudojimo atvaizduojant dokumentų vektorius efektą lyginant su pavienių žodžių vektoriais. Palyginus šių dviejų metodikų gaunamus rezultatus nustatyta, kad vien frazių naudojimas nesuteikė jokio pranašumo lyginant su paprastais žodžių krepšelio metodais. Taip pat eksperimentais įrodyta, kad šių dviejų metodikų vartojimas kartu turėto menkos reikšmės klasifikavimo efektyvumui [27].

Statistinės frazės, dar vadinamos *n-gramais* (angl. n-gram), remiasi n vienas po kito einančių žodžių seka. Tokių statistinių frazių panaudojimas klasifikavime pademonstravo akivaizdų pranašumą [16, 31]. Šių frazių tyrimai ir toliau buvo vykdomi kitų autorių [13, 4, 29, 2]. Rezultatai kolkas negalutiniai, o tyrimai šia kryptimi dar aktyviai tebetęsiami.

2.1.2. Termų apdorojimas

Žodžių krepšelio metodas – tai paprastas, tačiau efektyvus būdas atvaizduoti tekstinį dokumentą. Tarkime, kad dokumentas d sudarytas iš k termų, tuomet dokumentą galime atvaizduoti kaip termų svorių vektorių: $d = (t_1, t_2, ..., t_k)$, kur svoris t_k nusako, kiek k-asis termas prisideda prie dokumento d semantinės prasmės [41]. Svoris gali būti loginis arba skaitinis. Loginis svoris nusako termo buvimą ar nebuvimą dokumente, o skaitinis svoris tiksliau nusako santykinį termo svarbumą.

Egzistuoja daugybė metodų termo svoriui apskaičiuoti. Kokį metodą geriausia pasirinkti priklauso tik nuo konkrečios sprendžiamos užduoties. Dažniausiai naudojamas svorio nustatymo metodas – $TF \cdot IDF$ [39]. $TF \cdot IDF$ sudarytas iš dviejų dalių: termų pasikartojimo (angl. term frequency) ir dokumento pasikartojimo (angl. document frequency). Termo pasikartojimas nusako, kiek kartų termas t pasikartoja dokumente d, ir žymimas $TF(t, d)$. Tuo tarpu, dokumento dažnumas nusako keliuose dokumentuose randamas termas t ir žymimas $DF(t)$. O $TF \cdot IDF$ apskaičiuojamas pasitelkiant tokią formulę:

$$TF \cdot IDF(t, d) = TF(t, d) \cdot \log \frac{|D|}{DF(t)}, \tag{1}$$

kur $|D|$ – visų dokumentų kiekis.

2.1.3. Požymių atranka

Tekstinio dokumento atvaizdavimo vektoriumi metodai veda prie labai didelės dimensijų erdvės vektorių, turinčių daugiau nei 10000, o dažniausiai ir dar daugiau. Kai kurie mašininio mokymosi algoritmai nesugeba apdoroti didelio požymių kiekio, o taip pat ir klasifikavimo proceso greitis atvirkščiai proporcingas galimų požymių kiekiui. Be to, kai apmokymo duomenų kiekis yra ribotas per daug turimų požymių gali sukelti persimokymo efektą – fenomeną, kurio metu klasifikatorius išmoksta ne bendras klasių charakteristikas, o specifinius požymius apie duomenų aibę. Dėl to klasifikatorius puikiai klasifikuoja apmokymo duomenis, tačiau daug blogiau klasifikuoja dar nematytus testinius duomenis. Tyrimais nustatyta, kad tekstinių apmokymo dokumentų kiekis apytikriai proporcingas termų kiekiui, reikalingam išvengti persimokymo fenomeno [41]. Todėl kai apmokymo duomenų nepakanka, persimokymo galima išvengti sumažinus požymių kiekį. Dėl to požymių atranka – tai būtinas žingsnis, naudojamas tekstinio dokumento indeksavimo procese.

Požymių atrankos metodais siekiama iš požymių aibės išrinkti tokį požymių poaibį, su kuriuo būtų galima pasiekti maksimalų efektyvumą, tuo pačiu stengiantis neprarasti klasifikavimo tikslumo. Literatūroje yra pasiūlyta daugybė būdų kaip iš dokumento išgauti labiausiai jį apibūdinančius indeksinius žodžius [40]. Paprasčiausi požymių atrankos metodai – tai požymių, kurie pasitaiko rečiau nei nustatytą minimalų kartų kiekį pašalinimas, arba pašalinimas požymių, kurie randami rečiau nei nustatytame kiekyje dokumentų. Taip pat egzistuoja daugybė sudėtingesnių funkcijų daug tiksliau skaičiuojančių požymių svorius nei pastarasis: pvz. *informacijos įgijimo* (angl. information gain), χ^2 *chi-kvadrato* (angl. chi-square) ar *skirtumų santykio* (angl. odds ratio) metodai. Dauguma šių funkcijų atsižvelgia į ryšius tarp atskirų termų bei konkrečių kategorijų. Termo įvertis, gautas šiais metodais, paprastai yra priklausomas nuo kategorijos, todėl siekiant apskaičiuoti galutinį įvertį visoms kategorijoms galima pasinaudoti vienu iš kelių būdų – sumos, svertinės sumos, arba lokalaus įverčio kiekvienoje individualioje kategorijoje įverčiu.

Pakankamai išsamų įverčių funkcijų tyrimą atliko Y. Yang ir J. Pederson [52]. Savo ataskaitoje jie ištyrė kelių požymių atrankos metodų stipriasias ir silpnasias puses. Eksperimentus atliko pasitelkę *k artimiausių kaimynų* bei (angl. Linear Least Square Fit) klasifikatoriumi ir priėjo išvados, kad informacijos įgijimo ir chi-kvadratų metodai yra efektyviausi įverčių skaičiavimo metodai termų atrankos užduočiai spręsti. Toliau plačiau aptarsime du požymių atrankos metodus: *chi-kvadrato* bei L.-S. Chen ir C.-W. Chang pasiūlytą *CDW* metodą [7].

χ^2 **chi-kvadratas.** Chi-kvadratų metodas skaičiuoja abipusę priklausomybę tarp termo t ir kategorijos c, kuri gali būti palyginama su X^2 skirstiniu su vienu laisvės laipsniu ekstremumams įvertinti. Siekiant apskaičiuoti χ^2 įverčius galime pasitelkti dvišalių termo t ir kategorijos c atsitiktinumų lentelę (1 lentelė), kur A – dokumentų kiekis klasėje c, kuriuose randamas termas t, B – dokumentų, nesančių klasėje c, kiekis, kuriuose randamas termas t, C – dokumentų kiekis klasėje c, kuriuose neegzistuoja termas t, D – dokumentų kiekis, nesančių klasėje c, ir kuriuose neegzistuoja termas ts, N – visų dokumentų kiekis.

$A = DF(t, c)$	$C = DF(\neg t, c)$
$B = DF(t, \neg c)$	$D = DF(\neg t, \neg c)$
$N = A + B + C + D$	

1 lentelė: χ^2 dvišalių atsitiktinumų lentelė

Galutinis χ^2 įvertis apskaičiuojamas pagal formulę:

$$\chi^2 = \frac{N \times (AD - CB)^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)} \tag{2}$$

Kuo labiau termas t ir kategorija c nepriklausomi vienas nuo kito, tuo labiau χ^2 įvertis artėja prie nulio.

CDW **metodas.** CDW metodas – tai L.-S. Chen ir C.-W. Chang pasiūlytas kategorijų skirtumų svorių (angl. categorical difference weights) metodas [7], įkvėptas M. Simeon ir R. Hilderman pasiūlyto kategorinio proporcingumo skirtumo (angl. categorical proportional difference) metodo [42], skirto požymiams išgauti sprendžiant daugiaklasio klasifikavimo problemas. Siekiant išspręsti CPD metodo problemas, buvo modifikuotas CPD algoritmas ir pasiūlyta nauja jo versiją – $MCPD$ (angl. modified CPD). Galiausiai autoriai pasiūlė $MCPD$ pagrindu sukurtą CDW metodą. CDW įvertis apskaičiuojamas dviem etapais: pirma gaunama $MCPD$ vertė, kuri po to panaudojama apskaičiuoti galutinę CDW (6 formulė) vertę. Pasitelkus 1 lentelėje pateiktais žymejimais, $MCPD$ įvertis, priklausomai nuo situacijos, apskaičiuojamas trimis skirtingais būdais (3-5 formulės):

$$MCPD = |A - B|, kai A = 0 \; arba B = 0 \; arba A = B \tag{3}$$

$$MCPD = \frac{A}{B}, kai A > B \tag{4}$$

$$MCPD = \frac{B}{A}, kai A < B \tag{5}$$

Iš 3-5 formulių galime padaryti išvadą, kad kuo dažniau termas pasikartoja konkrečioje vienoje klasėje, tuo $MCPD$ įvertis didesnis. Ir priešingai, jei termas vienodai pasikartoja abiejose klasėse – $MCPD$ reikšmė lygi nuliui. Galiausiai, turint $MCPD$ įvertį, galutinis CDW įvertis apskaičiuojamas pagal formulę:

$$CDW = \frac{MCPD}{A + B} \tag{6}$$

Lyginant su χ^2 metodu, CDW tokiems termams, kaip jungtukai, artikeliai ar kiti abiejose kategorijose dažnai pasitaikantys kalbos žodžiai priskiria mažesnį įvertį nei reikšmingesniems. Tuo tarpu χ^2 metodas šių bendrinių, informacijos nenešančių termų nėra linkęs eliminuoti, ir dėl šios priežasties yra būtinas papildomas termų apdorojimo procesas, pvz. termų eliminavimas pagal kalbos dalis. Išsamesnė šių dviejų atrankos metodų rezultatų palyginamoji lentelė pateikta priede nr. 1

Kiti požymių atrankos metodai. Be minėtų požymių atrankos metodų yra galimi ir kiti būdai pašalinti mažiausiai reikšmingus termus. Pavyzdžiui, toks metodas kaip *skiriamųjų žodžių* (angl. stop words) pašalinimas bei žodžių kamienų skaičiavimai gali būti sėkmingai pritaikyti siekiant sumažinti požymių kiekį. *Skiriamieji žodžiai* – tai tokie neutralūs žodžai kaip artikeliai, prieveiksmiai, prielinksniai, jungtukai, skaičiai ir kt., nustatomi ir pašalinami pasitelkiant kalbos dalies nustatymo algoritmus (angl. part-of-speech tagging) [34], tokiu būdu paliekant daugiau informacijos teikiančias kalbos dalis: daiktavardžius, veiksmažodžius ir t.t. Žodžio šaknies analizė – tai metodas, kurio tikslas apjungti žodžius, turinčius bendrą morfologinę šaknį ir vertinti juos kaip tą patį žodį. Viso to rezultatas – sumažintas galimų termų kiekis.

2.1.4. Kiti indeksavimo metodai

Be žodžių krepšelio metodo gali būti vartojami ir kiti alternatyvūs indeksavimo metodai. Pavyzdžiui *Darmstadt* indeksavimo metodas [15], kuris analizuoja ne tik statistinę žodžių ir frazių informaciją, bet ir apjungia struktūrinę dokumento informaciją, pavyzdžiui, žodžio vietą (pavadinime, santraukoje, išvadose ar kitur) dokumente.

Kiti tyrėjai taip pat bandė įtraukti ir sintaksinę informaciją. Pavyzdžiui, S. Ray ir M. Craven [36], indeksuodami sakinius iš biomedicininės literatūros panaudojo ne tik žodžius ir

frazes, bet ir jų sintaksines kategorijas (t.y. kalbos dalių žymes).

Tokių įmantresnių indeksavimo metodų tikslas – siekis įtraukti papildomos informacijos apie duomenis, kurią vargu ar galėjo įskaičiuoti standartiniai indeksavimo metodai, ir tokiu būdu pagerinti klasifikavimo kokybę.

2.2. Klasifikavimo metodai

Algoritmų, taikomų įprastų tekstų klasifikavimui, yra nemažai. Išsami jų apžvalga ir automatinio mokymosi perspektyvos pateiktos F. Sebastiani darbe [41]. Dažniausiai moksliniuose darbuose minimas ir plačiausiai vartojamas yra Naive Bayes tekstinis klasifikatorius. Lyginant su kitais statistiniais klasifikatoriais [53] Naive Bayes klasifikatorius yra vienas paprasčiausių, tačiau tuo pačiu yra ir pakankamai efektyvus. Šis klasifikatorius dėl savo paprastumo yra labiausiai ištirtas – jis minimas beveik visuose straipsniuose, kuriuose kalbama apie klasifikavimą, ir vartojamas ne tik tekstui, bet ir kitų daugialypės terpės dokumentų požymiams klasifikuoti [11]. Kitas, ne ką mažiau žinomas ir naudojamas metodas, skirtas klasifikuoti tekstinius dokumentus, yra V. Vapnik ir jo kolegų [3, 10, 46] pasiūlytas neparametrinis klasifikavimo metodas – atraminių vektorių algoritmas SVM (angl. Support Vector Machines). SVM klasifikatorius vietoje empirinės apsirikimo rizikos (klasifikavimo klaidų mokymo imtyse lygis) mažinimo konstruoja klasifikavimo taisyklę taip, kad tikroji klasifikavimo klaida, t. y., tikimybė, kad klasifikatorius suklys sutikęs jam nematytą testinį pavyzdį būtų minimali. Paprasčiausiu atveju metodas yra tiesinis savo parametrų atžvilgiu, be to, skirtas spręsti paprastą vienos klasės uždavinį – teisingai atskirti duomenis, priklausančius klasei nuo nepriklausančių. Kita, nė kiek nemažesnė algoritmų grupė – tai pavyzdžiais paremti algoritmai, iš kurių pats žinomiausias ir populiariausias yra k-artimiausių kaimynų metodas [9, 14], sprendimą apie stebimo objekto klasę ar klases priimantis pagal sprendimus tų mokymo imties elementų, kurie yra arčiausiai (apibrėžto atstumo arba panašumo mato prasme) nagrinėjamojo. Iš kitų, sudėtingesnių algoritmų, vertėtų išskirti tokias stambias grupes, kaip neuroniniais tinklais paremti algoritmai [32, 38, 50] ir sprendimų taisyklių ir sprendimų medžių algoritmai [30, 49]. Plačiau pristatysime porą dažniausiai tekstiniams dokumentams klasifikuoti naudojamų klasifikatorių – tai Naive Bayes bei atraminių vektorių klasifikatoriai.

2.2.1. Naive Bayes klasifikatorius

Statistiniai metodai yra viena seniausių mašininio mokymosi paradigmų, darančių prielaidą, kad analizuojant šablonus galima pasinaudoti statistiniu tikimybiniu modeliu [12].

Bayes teorema suteikia galimybę apskaičiuoti tam tikro įvykio X tikimybę, kai turima pradinių stebėjimų aibė. Apmokymo metu ar bandant nustatyti šablono X kategoriją yra skaičiuojama klasės aposteriorinė tikimybė arba hipotezė h, $P(h|X)$, kuri yra išreiškiama formule

$$P(h|X) = \frac{P(X|h)P(h)}{P(X)}, \tag{7}$$

kur $P(h) = \frac{|h|}{N}$ yra h apriorinė tikimybė ($|h|$ ir N atitinkamai yra šablonų skaičius klasėje h ir šablonų skaičius visose klasėse, darant prielaidą, kad visos hipotezės yra vienodai tikėtinos), $P(X|h)$ – X aposteriorinė tikimybė, kurią sąlygoja h, o $P(X)$ – X apriorinė tikimybė (lygi konstantai).

Ši statistiškai optimali klasifikavimo taisyklė yra visuotinai priimtas standartas, su kuriuo lyginamos kitų klasifikavimo algoritmų darbo efektyvumas. Naive Bayes klasifikatorius atvaizduoja kiekvieną dokumentą d kaip n-matį atributinių reikšmių vektorių $\langle t_1, t_2, ..., t_n \rangle$. Sakykime, kad egzistuoja C klasių $c_1, c_2, ..., c_{|C|}$, klasifikatorius daro prielaidą, kad nežinomas dokumentas d priklauso klasei, turinčiai didžiausią aposteriorinę (angl. a posteriori) tikimybę, t.y. d yra priskiriamas klasei c_i tada ir tik tada kai $P(c_i|d) > P(c_j|d)$ kiekvienam $1 \leq j \leq l$ ir $j \neq i$. Naudojant (7) lygtį gauname:

$$P(c_i|d) = \frac{P(d|c_i)P(c_i)}{P(d)} \tag{8}$$

Siekiant sumažinti skaičiavimo kaštus, klasifikatorius daro naivią ir supaprastintą prielaidą, kad n atributų yra nepriklausomi vienas nuo kito. Ši nepriklausomybė išreiškiama kaip

$$P(c_i|d) \infty P(c_i) \prod P(a_j|c_i), \tag{9}$$

kur $1 \leq j \leq n$, $P(d)$ yra konstanta visoms klasėms, $P(c_i) = \frac{|c_i|}{N}$, ir belieka surasti tik $P(d|c_i)$ maksimumą, kurį apskaičiuojame pagal formulę:

$$P(c_i|d) = \prod P(a_j|c_i), \tag{10}$$

kur $1 \leq j \leq n$. Tai žymiai sumažina skaičiavimų kaštus, kadangi skaičiuojamas tik klasių

pasiskirstymas. Bayes klasifikatoriai yra labai paprasti, jiems užtenka tik vieną kartą pereiti per duomenų bazės turinį, tuo pačiu jie gali greitai ir tiksliai atlikti pavestą užduotį didelėse duomenų bazėse. Darbo efektyvumas nė kiek nenusileidžia sprendimų medžių ir neuroninių tinklų darbo efektyvumui. Bayes klasifikatoriai turi mažą paklaidos tikimybę, tačiau yra galimos klaidos dėl (a) supaprastintų prielaidų ir (b) netinkamai parinktų apmokymo duomenų.

2.2.2. Atraminių vektorių klasifikatorius

Atraminių vektorių algoritmai (angl. support vector machines, *SVM*) yra tapę vienu moderniausiu įrankiu teksto kategorizavimo srityje [21]. *SVM* algoritmai perkelia apmokymo duomenų aibę į daugiamatę erdvę bei stengiasi surasti tokią hiperplokštumą, kuri atskirtų duomenų aibės taškus taip, kad riba tarp teigiamos ir neigiamos klasės duomenų būtų kuo didesnė, o klaidingo klasifikavimo atvejų būtų kuo mažiau. Artimiausi hiperplokštumos taškai vadinami atraminiais vektoriais. Šie taškai iš esmės nulemia hiperplokštumos poziciją.

Kaip pavyzdį aptarsime mums aktualiausią binarinį tiesinį *SVM* algoritmą. Tarkime, kad x yra duomenų taškas, w yra koeficiento vektorius, o b – konstanta. Tiesiškai atskiriamos erdvės hiperplokštumą galime apibrėžti tokia tiesine funkcija:

$$f(x) = wx + b, \tag{11}$$

kur $wx + b > 0$ teigiamiems duomenims, o $wx + b < 0$ – neigiamiems. Pasinaudosime y_i, kad apsibrėžtume duomenų taško x_i klasę, kur reikšmė 1 naudojama teigiamai klasei nusakyti, o -1 neigiamai. Siekiant minimizuoti klaidingų klasifikavimo sprendimų kiekį, $f(x)$ privalo atitikti tokią salygą:

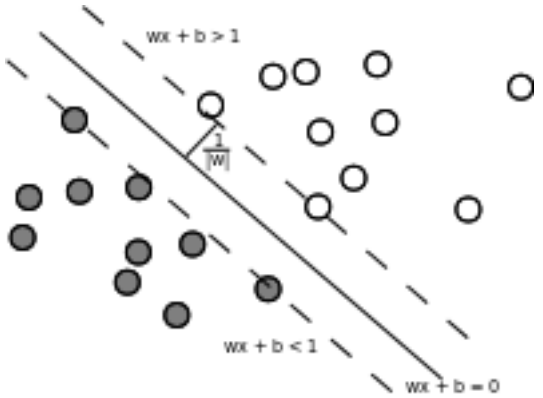
$$y_i f(x_i) \leq 0, kur\ i = 1, ..., n \tag{12}$$

Šis principas vadinamas empiriniu rizikos mažinimu (angl. empirical risk minimization). Kadangi *SVM* algoritmas klasifikavime labiau akcentuoja teisingų nei klaidingų sprendimų kiekį, todėl siekiama, kad duomenys abiejose hiperplokštumos pusėse būtų išsidėstę kuo toliau nuo tos pačios hiperplokštumos. Tai gali būti atlikta apribojant funkciją $f(x)$, kad atitiktų sąlygą:

$$y_i f(x_i) \leq 1, kur\ i = 1, ..., n \tag{13}$$

Atitinkamai, $wx + b > 1$ – teigiamiems duomenims ir $wx + b < 1$ – neigiamiems. Tokiu

atveju ribos tarp dviejų erdvės pusių plotis yra $\frac{2}{|w|}$. Yra siekiama, kad riba turi būtų kiek įmanoma platesnė. 1 pav. matome tiesinio SVM algoritmo rastą hiperplokštumą tarp teigiamų ir neigiamų duomenų pavydžių (1 pav.).



1 pav.: SVM klasifikatoriaus apskaičiuota hiperplokštuma h, apibrėžta funkcija $f(x) = wx + b = 0$. Atraminiai vektoriai – taškai išsidėstę arti h ne didesniu atstumu nei $\frac{1}{|w|}$.

Iki šiol SVM sprendinys buvo suvestas į optimizavimo uždavinį, t.y. kuo labiau maksimizuoti ribą $\frac{2}{|w|}$ naudojantis $y_i f(x_i) \leq 1$, $i = 1, ..., n$. Ši optimizavimo problema gali būti išspręsta pasitelkiant Lagranžo daugiklius. Daugiau informacijos apie metodus kaip rasti optimaliausius parametrus w ir b galima rasti J. Vert ir jo kolegų darbe [47].

Pasak T. Joachims [21], SVM klasifikatoriai klasifikavimo kokybe lenkia kitus klasifikatorius, tokius kaip Naive Bayes, k-artimiausių kaimynų bei sprendimų medžių algoritmus, bei turi keletą pranašumų prieš minėtus klasifikatorius.

Kaip minėta 2.1.3 skyriuje, teksto klasifikavime susiduriama su labai aukšto dimensijų kiekio duomenų vektoriais. Tuo tarpu viena svarbiausių SVM klasifikatorių savybių yra galimybė sėkmingai mokytis nepriklausomai nuo duomenų vektorių dimensijų skaičiaus, kadangi jie turi perpildymo apsaugą, kuri nepriklauso nuo požymių kiekio. Todėl SVM algoritmai yra vieni tų, kurie turi daugiausiai potencialo susidoroti su tokiais duomenų kiekiais.

Taip pat teksto klasifikavime egzistuoja tik labai mažai nereikšmingų duomenų. Tyrimų [21] duomenimis buvo nustatyta, kad netgi požymiai, kurie pasitelkiant požymių svarbumo įvertinimo matą $TF \times IDF$, ir buvo reitingo lentelės apačioje vis tiek savyje saugojo nemažai vertingos informacijos ir buvo ne mažiau svarbūs nei požymiai reitinguojami aukščiausiose pozicijose. Taip pat klasifikatorius pasinaudojęs šiais „mažiausiai vertingais“ požymiais pasiekė geresnę klasifikavimo kokybę nei klasifikatorius pasinaudojęs atsitiktinių požymių atranka. Todėl galima padaryti išvadą, kad geras klasifikatorius turi apjungti daug požymių, o agresyvios požymių atrankos dėka galima tik prarasti vertingus duomenis.

Kita problema, su kuria susiduriama tekstinių dokumentų klasifikavime, tai labai ne-

gausūs duomenų vektoriai. Kiekvieną dokumentą atitinkantis duomenų vektorius savyje saugo tik labai mažą dalelę įrašų, kurių svoris nėra lygus 0, t.y., kiekviename dokumente tėra tik absoliuti mažuma visų galimų požymių. J. Kivinen su kolegomis [25] pateikė teorinius ir empirinius įrodymus, kad SVM savybėmis pasižymintys algoritmai yra tinkami spręsti problemas, pasižyminančias dideliais išretintais duomenų vektoriais.

Galiausiai, dauguma tekstinio klasifikavimo problemų yra tiesiškai atskiriamos, t.y. daugumos kategorijų dokumentus galima tiesiškai atskirti vienus nuo kitų [21], o atraminių vektorių klasifikavimo algoritmo tikslas – surasti tokius kategorijas atskiriančius atraminius vektorius.

Dėl šių priežasčių galima teigti, kad SVM algoritmai turėtų puikiai susidoroti su tekstinio turinio klasifikavimo problemomis.

2.3. Klasifikatoriaus įvertinimas

Galima išskirti kelis pagrindinius klasifikatoriaus įvertinimo kriterijus – tai klasifikatoriaus apmokymo naudingumo koeficientas (vidutinis laikas, reikalingas iš turimo duomenų rinkinio D sukonstruoti klasifikavimo taisyklę f), klasifikavimo naudingumo koeficientas (vidutinis laikas, reikalingas suklasifikuoti dokumentų aibę D pasitelkiant klasifikavimo taisyklę f) bei klasifikavimo efektyvumas (klasifikavimo taisyklės f gebėjimas padaryti teisingą sprendimą dokumentų atpažinime). Pats svarbiausias ir labiausiai reikšmingas parametras iš šių trijų – tai klasifikavimo efektyvumas.

Teksto klasifikavimo efektyvumo įvertinimo parametrų yra nemažai, tačiau dažniausiai naudojami yra tikslumas (angl. precision) ir atkūrimas (angl. recall) [41], kuriuos toliau patogumo dėlei žymesime atitinkamai π ir ρ. Klasifikatoriaus c kategorijos tikslumas – tai santykis teisingai suklasifikuotų dokumentų iš klasės c su visais dokumentais, kuriems klasifikatorius priskyrė klasę c. Tuo tarpu atkūrimas skaičiuoja teisingai suklasifikuotų dokumentų iš klasės c santykį su visais dokumentais, kurie turėjo būti suklasifikuoti kaip priklausantys klasei c. Tikslumo ir atkūrimo parametrams skaičiuoti pasitelksime tokius dažnai naudojamus matavimo vienetus, kaip TP_c, teisingo atpažinimo (angl. true positive), t.y. dokumentų kiekis teisingai priskirtas kategorijai c; FP_c, klaidingo atpažinimo (angl. false positive) – dokumentų kiekis klaidingai priskirtas kategorijai c; TN_c, teisingo atmetimo (angl. true negative) – dokumentų kiekis teisingai nepriskirtas kategorijai c; TN_c, klaidingo atmetimo (angl. true negative) – dokumentų kiekis klaidingai nepriskirtas kategorijai c. Šiuos įverčius galime patogiai apskaičiuoti pasinaudojus 2 lentele.

		Tikroji kategorija	
		Taip	Ne
Apskaičiuota kategorija	Taip	TP	FP
	Ne	FN	TN

2 lentelė: Efektyvumo matavimo vienetų nustatymo lentelė

Konkrečiai kategorijai c tikslumo ir atkūrimo įverčius galime apskaičiuoti pasitelkę tokias formules:

$$\pi_c = \frac{TP_c}{TP_c + FP_c}, \rho_c = \frac{TP_c}{TP_c + FN_c} \tag{14}$$

Kai efektyvumas skaičiuojamas kelioms kategorijoms, tam pasitelkiami lokalūs π ir ρ įverčiai ir apskaičiuojami globalūs įverčiai visai kategorijų aibei. Norint tai pasiekti yra du galimi būdai: mikrovidurkis (angl. microaverage) arba makrovidurkis (angl. macroaverage) [41]. Mikrovidurkis suteikia kiekvienam dokumentui vienodą svorį, tuo tarpu makrovidurkis – vienodą svorį kiekvienai klasei. Mikro (15 formulė) ir makro (16 formulė) vidurkius galime apskaičiuoti pagal tokias formules:

$$\pi_{mi} = \frac{\sum_{c \in C} TP_c}{\sum_{c \in C}(TP_c + FP_c)}, \rho_{mi} = \frac{\sum_{c \in C} TP_c}{\sum_{c \in C}(TP_c + FN_c)} \tag{15}$$

$$\pi_{ma} = \frac{\sum_{c \in C} \frac{TP_c}{TP_c + FP_c}}{|C|}, \rho_{ma} = \frac{\sum_{c \in C} \frac{TP_c}{TP_c + FN_c}}{|C|} \tag{16}$$

Vis dėlto, nei tikslumo, nei atkūrimo įverčiai negali būti tikslūs efektyvumo indikatoriai, jei yra naudojami atskirai. Paprastai aukštas tikslumo įvertis gali būti pasiektas žemo atkūrimo įverčio kaina [41]. Vadinasi, yra būtina klasifikatoriaus efektyvumą apskaičiuoti juos kombinuojant. Pats populiariausias tokio kombinavimo metodas – tai $F - funkcija$ (angl. $F - measure$) [19], kurią galime interpretuoti kaip svorinį atkūrimo ir tikslumo įverčių vidurkį ir kuri apibrėžiama kaip:

$$F_\beta = \frac{(\beta^2 + 1) \cdot \pi \cdot \rho}{\beta^2 \cdot \pi + \rho}, 0 \leq \beta \leq +\infty \tag{17}$$

Parametras β pritaiko tikslumo ir atkūrimo įverčiams svorį. Dažniausiai naudojamos β reikšmės yra 0.5, 1.0 ir 2.0, o $F - funkcijos$ atitinkamai vadinamos $F_{0.5}, F_1$ ir F_2. Svoriui β artėjant link ∞, $F - funkcijos$ reikšmė artėja link atkūrimo įverčio ρ.

3. Internetinio turinio analizė

3.1. Tinklalapių struktūra

Tinklalapiai – tai daugialypės terpės dokumentai, kuriuose pateikiamas tekstinis, grafinis, vaizdinis, garsinis ar bet koks kito pobūdžio turinys. Kiekvieną *HTML* dokumentą grafiškai galime pavaizduoti medžio pavidalu (priedas nr. 2), kurio pagrindinės ir dažniausiai vartojamos sudedamosios dalys būtų puslapio metaduomenys (pavadinimas, raktiniai žodžiai, trumpas puslapio aprašymas) ir puslapio turinys, sudarytas iš vieno ar daugiau prasminių blokų, kurių kiekvienas dar gali turėti savyje tokių pačių blokų arba lapinių elementų, tokių kaip tekstiniai elementai, nuorodos, nuotraukos, formų elementai, *JAVA* programėlės, *FLASH* animacija, vaizdo ar garso failai ir kt. Kiekvienas blokas ar medžio lapinis elementas gali turėti papildomą tekstinę informaciją (dažnai vartojami *alt* ir *title* atributai), skirtą pateikti vartotojui daugiau informacijos apie konkretų puslapio elementą.

Vienas iš svarbiausių *HTML* dokumento pateikimo principų yra tas, kad dokumento elementai pagal savo semantinę prasmę yra jungiami blokais, t.y. dažniausiai viename bendrame bloke saugoma susijusi informacija, pvz. straipsnio tekstas, jo iliustracija bei pavadinimas. Tačiau tą patį, vizualiai identišką puslapį, galimą aprašyti ne vienu būdu į pagalbą pasitelkiant skirtingas *HTML* kalbos žymas bei esant reikalui ir *CSS* stilių lenteles. Tai daugiausiai priklauso nuo to kaip pageidaujama vizualiai pateikti turinį vartotojui bei nuo kiekvieno programuotojo įgūdžių ir programavimo stiliaus. Be viso to ir pačios žymos yra skirtos tik nusakyti, kaip reikia vizualiai pateikti turinį, bet nepateikia jokios informacijos apie tai kokio pobūdžio turinys saugomas jose. Dėl to nors vartotojui vizualiai nesunku atskirti reikšminius puslapio blokus (priedas nr. 3), tačiau programiškai be galo sudėtinga nustatyti, kur prasideda vienas turinio blokas, o kur prasideda kitas bei kurie blokai yra vienas su kitu susiję, todėl šią *HTML* dokumentų savybę išnaudoti yra begalo sudėtinga.

3.2. Tinklalapių indeksavimas

Pats paprasčiausias būdas atvaizduoti tinklalapį – tai išgauti visą tekstinį turinį saugomą *BODY* elemente. Toks metodas neleidžia išnaudoti tinklalapio savybių, t.y. *HTML* dokumento struktūros ir sąryšių su kitais dokumentais. Siekiant išnaudoti HTML dokumento struktūrą [51], galima atsižvelgti į tai, kokiame elemente saugomi žodžiai. Pavyzdžiui, galima puslapį apibūdinti tik iš *TITLE* elemente aptinkamų žodžių. Siekiant gauti didesnės naudos būtina žinoti, kur galima aptikti daugiau informacijos duodančių žodžių. Pavyzdžiui,

galima teigti, kad *TITLE* elemente saugomi žodžiai yra daug labiau nusakantys dokumento esmę nei žodžiai saugomi *BODY* elemente. D. Riboni [37] ir J. M. Pierre darbuose [33] buvo išbandyti įvairūs tinklalapius apibūdinantys variantai ir nustatyta, kad geriausi klasifikavimo rezultatai buvo pasiekti naudojant tik dokumentų meta duomenis, o ne patį dokumento turinį. Iš pradinės duomenų aibės autoriai pašalino puslapius, kurie neturi meta duomenų. Dėl šios priežasties buvo nukrypta nuo realybės, kadangi internete daugybė dokumentų nenaudoja arba naudoja klaidingus ar net klaidinančius meta duomenis ir tokiais atvejais klasifikavimo rezultatai pasitelkiant tik meta duomenis būtų iškraipyti.

Kitas būdas naudojamas apibūdinti tinklalapius, paremtas išorinių tinklalapių, į kuriuos nurodo analizuojamas dokumentas, turiniu. Toks metodas buvo panaudotas S. Chakrabarti darbe [8], tačiau klasifikavimo rezultatai buvo prastesni nei įprasto tekstinio klasifikavimo. J. Furnkranz [17] pasitelkė atvirkščią metodą ir pasinaudojo išoriniais dokumentais, turinčiais nuorodų nukreiptų į analizuojamą objektą. Buvo panaudoti nuorodose pateikti žodžiai, taip pat žodžiai esantys netoli nuorodų. Toks principas pagerino klasifikavimo rezultatus, tačiau yra sunkiai įgyvendinamas, nes praktiškai neįmanoma gauti puslapių, kurie saugo nuorodas į analizuojamą dokumentą. Taip pat abu pasiūlyti metodai reikalautų atsiųsti ir išanalizuoti visus išorinius dokumentus, kas užimtų žymiai daugiau laiko ir skaičiuojamosios galios nei įprastos dokumentų atvaizdavimo metodikos.

Kitas galimas variantas, kurį savo darbe aprašė T. Joachims [22], tai šių dviejų paminėtų dokumentų atvaizdavimo būdų apjungimas. Šiame darbe buvo apjungta tekstinė ir dokumentų sąryšių informacija bei pasitelktas atraminių vektorių klasifikatorius. Nustatyta, kad kombinuojant skirtingas dokumento atvaizdavimo formas klasifikavimo tikslumas geresnis, nei naudojant vieną iš kombinuotame modelyje panaudotų formų. Taip pat kombinavimas yra neišvengiamas norint apsisaugoti nuo dažnai pasitaikančių problemų, kai puslapyje trūksta vieno ar kito tipo informacijos. Pavyzdžiui, klasifikuojant dokumentą, turintį mažai sąryšių su išoriniais dokumentais, tuo pačiu per mažai būtinų požymių, tik nuorodomis pagrįsto dokumento apibūdinimo būdo panaudojimas yra netikslus.

Pasinaudoje minėtomis įdėjomis pasiūlysime dokumento pateikimo būdą sudarant duomenų vektorių ne tik pasitelkiant įprastą tekstinį turinį, bet taip pat ir raktinius žodžius bei nuorodas, tačiau nereikalaujantį atsisiųsti su nuorodomis susietų išorinių dokumentų.

3.3. Tinklalapių klasifikavimas

Nors ir egzistuoja ne vienas straipsnis, sprendžiantis tekstinių dokumentų klasifikavimo problemas, tačiau juose dažniausiai tik esama užuominų apie internetinius puslapius bei trūksta gilesnės internetiniams puslapiams būdingų požymių analizės. F. Sebastiani savo darbe [41] pagrindinį dėmesį skyrė tradiciniam tekstiniam klasifikavimui. S. Chakrabarti [6] bei R. Kosala ir H. Blockeel [26] bendrai apžvelgė internetinio turinio analizės sprendimus ir mažiau koncentravosi į patį klasifikavimą. J. Furnkranz [18] savo darbe apžvelgė įvairius internetinio turinio analizės aspektus bei trumpai aptarė nuorodų struktūros panaudojimą siekiant pagerinti internetinio turinio klasifikavimo rezultatus. Iš labiausiai su mūsų problema susijusių darbų galėtume paminėti X. Qi and B. D. Davison [35] bei D. Riboni [37] darbus, kuriuose išimtinai kalbama tik apie internetinių puslapių klasifikavimo problemas, internetiniams puslapiams būdingų požymių išskyrimą, daugiausiai įtakos turinčių požymių išskyrimą bei klasifikavimo algoritmus.

Lyginant su įprastu tekstiniu klasifikavimu, internetinio turinio klasifikavimas iš esmės skiriasi keliais aspektais. Pirmiausia, tradicinis teksto klasifikavimas atliekamas klasifikuojant labai gerai struktūrizuotus dokumentus [5]. Internetiniai puslapiai tokia savybe nepasižymi. Antra, tinklalapiai – tai daug daugiau nei tik paprasti tekstiniai failai, tai dalinai struktūrizuoti dokumentai saugomi *HTML* formatu siekiant juos vizualiai patraukliai pateikti vartotojams. Nors kitų rūšių dokumentų rinkiniai taip pat gali savyje turėti struktūrinės ir atvaizdavimui skirtos informacijos, tokios kaip žymos, tačiau prieš klasifikavimą ši informacija vis tiek yra pašalinama ir klasifikavimo algoritmams įtakos nedaro. Galiausiai, internetiniuose dokumentuose aprašomi sąryšiai su kitais dokumentais. *HTML* dokumentai savyje saugo nuorodas į kitus dokumentus, o kiti dokumentai savo ruožtu gali saugoti atgalines nuorodas atgal į tą patį dokumentą. Nors ši savybė ir nėra unikali tik internetinio dokumentams, tačiau nėra dažnai naudojama sprendžiant įprastas tekstinio klasifikavimo problemas. Nors daug darbo yra nuveikta panaudojant nuorodas internetinio turinio klasifikavime, į struktūrines tinklalapių savybes atsižvelgiama retai. Šiame darbe pasistengsime pateikti paprastą ir efektyvų būdą, skirtą klasifikuoti internetinius puslapius pasitelkiant ne tik nuorodas, bet ir *HTML* dokumentų struktūrinius blokus.

4. Internetinio turinio klasifikavimas

Kaip ir bet kurio tipo dokumentų klasifikavimas, taip ir internetinio turinio klasifikavimas susideda iš dviejų standartinių etapų:

1. Klasifikatoriaus apmokymas, klasifikavimo taisyklės f sudarymas.
2. Naujų dokumentų klasifikavimas pasitelkus 1 etape gauta klasifikavimo taisykle f.

4.1. Internetinio turinio požymių išskyrimas ir apmokymas

Kiekvieną bet kokio tipo dokumento klasifikavimo procesą galime suskirstyti bent į kelis dažniausiai vartojamus etapus:

1. Požymių iš apmokymo duomenų išskyrimas.
2. Požymių atranka, mažiausiai reikšmingų požymių pašalinimas.
3. Apmokymo aibės dokumentų indeksavimas, t.y. požymių vektorių sudarymas naudojant po 2 etapo likusius, daugiau įtakos turinčius požymius.
4. Klasifikatoriaus apmokymas pasitelkiant apmokymo aibės dokumentų požymių vektorius ir galutinio klasifikavimo žodyno D paruošimas.

Požymių išskyrimas. Šio skyriaus 4.1.1 - 4.1.4 dalyse aprašysime algoritmo, skirto iš *HTML* dokumento medžio ar bet kurio jo pomedžio išgauti klasifikavimui reikalingus požymius. Be šių žingsnių neapsieina ne tik apmokymo, bet ir pačio klasifikavimo algoritmas, kadangi jo metu išanalizuojamas *HTML* dokumento pomedis, išgaunamas ir apdorojamas tekstinis turinys bei sudaromi raktinių bei neraktinių termų dažnumų vektoriai, kuriuos vėliau galima panaudoti kaip įvesties duomenis požymių atrankos bei indeksavimo procesams įgyvendinti.

Požymių atranka. Šiame darbe siūlomas algoritmas nereikalauja naudoti būtent vienokio ar kitokio požymių atrankos algoritmo, todėl galima naudoti bet kurį, svarbiausia, kad jo rezultatai tenkintų keliamus reikalavimus ir pateisintų lūkesčius. Šiame darbe požymių atrankai pasitelksime 2.1.3 skyriuje aprašytą *CDW* metodą, kadangi šis parodė visai neblogus atrankos rezultatus lyginant su tokiu žinomu ir pripažintu algoritmu kaip χ^2. Be to

šis metodas yra linkęs tokiems mažai informacijos teikiantiems žodžiams kaip jungtukai, artikeliai ir pan. suteikti mažiausią svorį, todėl leidžia išvengti būtinybės sudaryti ir palaikyti *skiriamųjų žodžių* duomenų bazės kiekvienai žinomai kalbai.

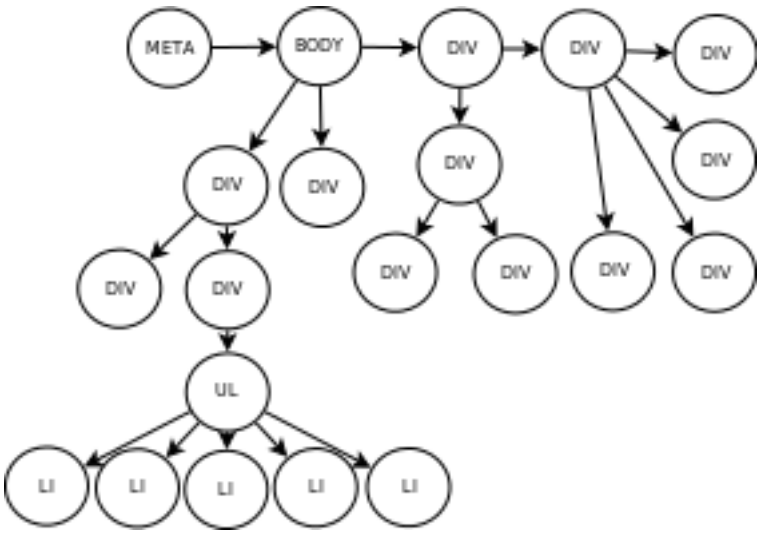
Dokumentų indeksavimas. Žinant kaip atliekamas požymių išskyrimas ir jų atranka belieka pateikti siūlomus dokumento požymio vektoriaus (indekso) sudarymo principus (4.1.5 poskyris). Šio etapo metu dokumentai paverčiami į vienetinius vektorius, kurie leidžia juos tarpusavyje palyginti ir tuo pačiu apmokyti klasifikatorių.

Klasifikatoriaus apmokymas. Tai paskutinis klasifikatoriaus apmokymo etapas (4.1.6 poskyris), kurio metu iš apmokymo duomenų aibės apmokomas klasifikatorius bei sudaromas klasifikavimui skirtas žodynas D, sudarytas iš termų aibės bei, mūsų naudojamo SVM klasifikatoriaus atveju, apskaičiuoto termų svorinių įverčių.

4.1.1. *HTML* dokumento medžio transformacija

Kaip anksčiau buvo minėta, šiame darbe aprašomo klasifikavimo algoritmo vienas iš pagrindinių tikslų – galimybė blokuoti tik dalį puslapio turinio, tuo pat metu paliekant likusias turinio dalis matomas vartotojui. Siekiant įgyvendinti šią užduotį algoritmo tikslas yra nustatyti kuo didesnio kiekio *HTML* dokumento elementų kategorijas, t.y. stengtis suklasifikuoti kuo daugiau medžio mazgų, o tuo pačiu ir pomedžių. Tokia užduotis reikalauja atlikti daug pakartotinių skaičiavimų, kadangi *HTML* dokumentai – tai kelis šimtus, o neretai ir tūkstančius mazgų turinčios medžio pavidalo struktūros. Todėl pirmas internetinių puslapių požymių išskyrimo etapas, kurį privalome atlikti, tai pradinio *HTML* dokumento medžio (priedas nr. 2) transformavimas į klasifikatoriui priimtiną kompaktiškesnę ir greičiau apdorojamą medžio struktūrą (2 pav.).

Pagrindinis transformavimo principas – surasti tarp medžio mazgų tuos elementus, kurie pagal savo paskirtį yra naudojami tekstinio ir grafinio turinio blokams formuoti. Tokius elementus vadinsime blokiniais (3 lentelė). Taip pat verta paminėti, kad ne kiekvieną blokinį elementą verta talpinti į naują medį. Kiekviename puslapyje galima aptikti tekstinio turinio blokų, kuriuose blokiniai elementai pasitelkiant stilių lenteles naudojami tik kaip formatavimo priemonė, o informacija juose yra glaudžiai susijusi. Todėl darome prielaidą, kad jeigu blokiniame elemente egzistuoja tik tekstinis turinys, tai tokį blokinį elementą su visais jame esančiais vaikiniais blokiniais elementais galime traktuoti kaip vieną bloką ir jo neskaldyti. Tai, pavyzdžiui, aktualu su lentelėmis, kuriose gali būti talpinama įprasta



2 pav.: *HTML* dokumento semantiniai blokai

statistinė, tarpusavyje susijusi informacija, ir nėra tikslo jos skaidyti į dalis, o geriausia traktuoti kaip vieną vienetą. Taip pat šiame etape apdorojame ir dokumento meta duomenis, kurie patogumo dėlei tampa nauju dirbtiniu šakniniu transformuoto medžio mazgu *META*, kuriame saugomi dokumento pavadinimas, raktiniai žodžiai bei puslapio aprašymas. Kaip matome 2 pav., transformuoto medžio mazgų kiekis pastebimai sumažėja, tokiu būdu gaunama kompaktiškesnė forma.

Blokiniai elementai					
BODY	*FRAME*	*FRAMESET*	*IFRAME*	*DIV*	*ARTICLE*
SECTION	*TABLE*	*TH*	*TD*	*DL*	*DD*
DT	*OL*	*UL*	*LI*		
Objektai					
APPLET	*AUDIO*	*CANVAS*	*EMBED*	*FRAME*	*IFRAME*
IMG	*OBJECT*	*VIDEO*			
Nuorodos					
A					

3 lentelė: *HTML 4* ir *HTML 5* elementų kategorijos

Nauja medžio struktūra sudaryta iš mazgų, saugančių savyje nuorodas į originalaus *HTML* dokumento elementus. Kadangi klasifikatoriui teks daug kartų klaidžioti medžio struktūra, todėl kiekvienas mazgas papildomai saugo ir nuorodas į tėvinį bei vaikinius elementus. Galutinį šiame etape naudojamo tranformavimo algoritmą galime aprašyti keliais etapais:

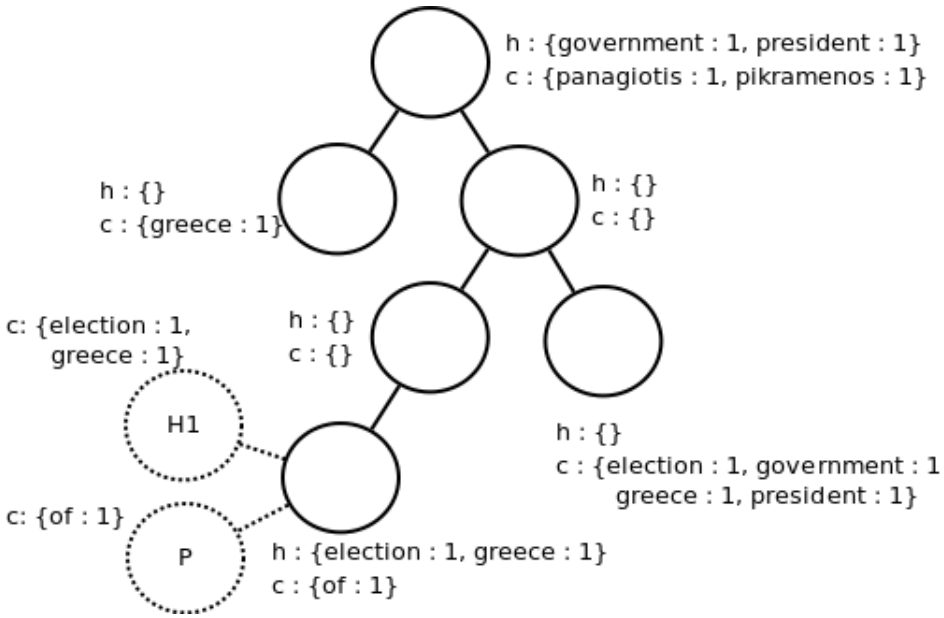
1. Iš originalaus *HTML* medžio pašalinami neblokiniai elementai (3 lentelė).

2. Iš likusių blokinių elementų pašalinami tie elementai, kuriuose neegzistuoja nė vieno

objekto ar nuorodos (3 lentelė).

3. Iš *HEAD* mazgo elementų *TITLE, META[KEYWORDS]* ir *META[DESCRIPTION]* išgaunami dokumento meta duomenys. Juos apjungus sudaromas vienas dirbtinis *META* mazgas, kuris tampa medžio šakniniu elementu.

4.1.2. Žodžių dažnumo vektorių sudarymas



3 pav.: Mazgų žodžių dažnumo vektorių h ir c sudarymas

Sekantis dokumento analizės etapas – tai kiekvieno transformuoto medžio mazgo žodžių dažnumų vektorių sudarymas (3 pav.). Kiekvienam mazgui sudaroma po du vektorius: vektorius h sudaromas iš raktinių žodžių, o vektorius c iš visų likusių žodžių. Vektoriai sudaromi tik iš konkrečiame mazge saugomų tekstinių duomenų (pvz., 3 pav. taškinėmis linijomis pažymėti *H1* ir *P* elementai) nenaudojant vaikinių blokinių mazgų turinio. Raktiniai žodžiai išgaunami iš tokių *HTML* elementų kaip *A, H1, H2* ir kt. Šie elementai (4 lentelė) paprastai naudojami straipsnių pavadinimams, nuorodoms į kitus puslapius pateikti. Juose pateikiama informacija labiau išryškinta iš bendro konteksto ir dėl to daroma prielaida, kad suteikia daugiau svorio, nustatant galutinę kategoriją.

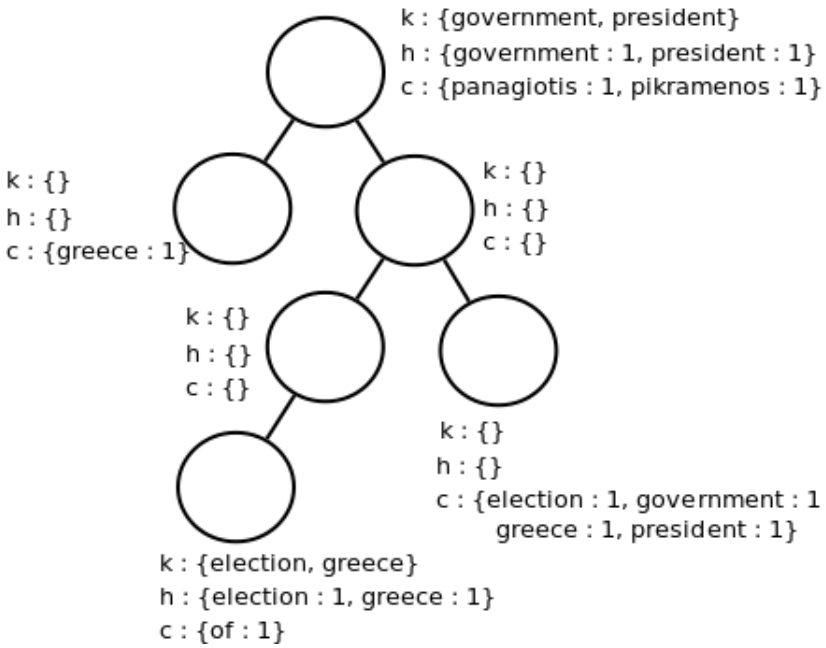
Raktiniai elementai								
A	*H1*	*H2*	*H3*	*H4*	*H5*	*H6*	*HEADER*	*FIGCAPTION*

4 lentelė: *HTML 4* ir *HTML 5* raktiniai elementai

4.1.3. Raktinių žodžių analizė

Kitas dokumento turinio interpretavimo etapas sudarytas iš trijų žingsnių: raktinių žodžių aibių sudarymo, jų veikimo sričių nustatymo bei raktinių žodžių paveldėjimo.

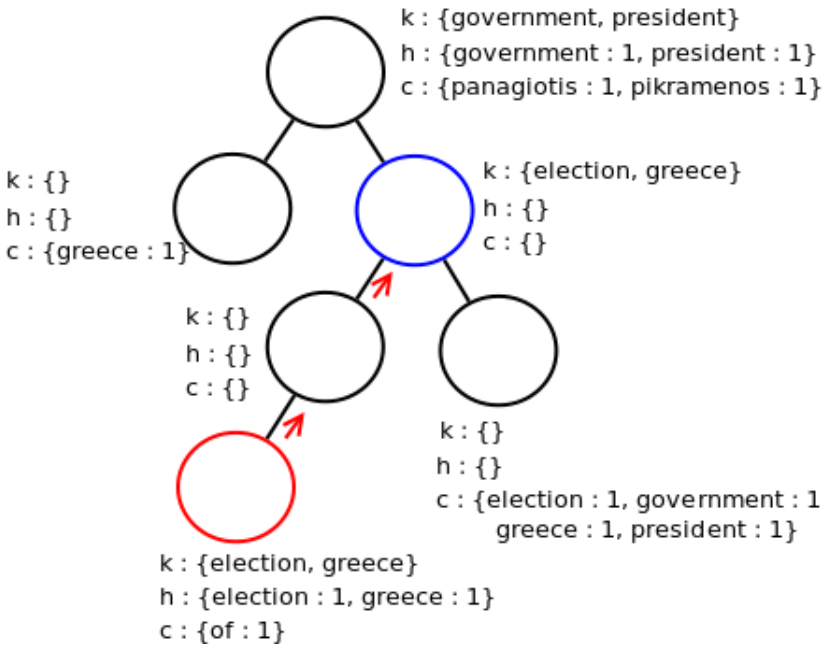
Raktinių žodžių aibių sudarymas. Šio etapo tikslas paprastas – tolimesnės analizės patogumui iš vektorių h sudaryti raktinių žodžių aibes k. Kiekviename medžio mazge aibė k sudaroma iš tame pačiame mazge esančio vektoriaus h paimant tik unikalius žodžius (4 pav.).



4 pav.: Raktinių žodžių aibių k sudarymas

Raktinių žodžių veikimo sričių nustatymas. Tai labai svarbus etapas, kurio tikslas nustatyti, kurioje medžio atšakoje konkreti raktinių žodžių aibė k turi su joje esančiais žodžiais susijusio turinio. Yra daroma prielaida, kad jei toje pačioje medžio atšakoje, kurios šakniniame mazge yra analizuojama aibė k, egzistuoja prasmingo turinio arba bent vienas objektas (3 lentelė), tai traktuojama, kad ši medžio atšaka ir priklauso raktinių žodžių aibės k veikimo sričiai. Prasmingu turiniu laikome tokius c vektorius, kuriuose egzistuoja bent vienas žodis, kuris nėra skaičius ir kurio ilgis yra ne mažiau pasirinktos vertės (pvz. 3 simboliai).

Taip pat pasitaiko atvejų, kai pradinėje aibės k pomedyje neegzistuoja jokio reikšmingo turinio. Tai labai dažnas atvejis, kai pvz. formatuojant *HTML* dokumentą straipsnio pavadinimas (vektorius k) ir straipsnio turinys (vektorius c) patenka į skirtingus pomedžius, o

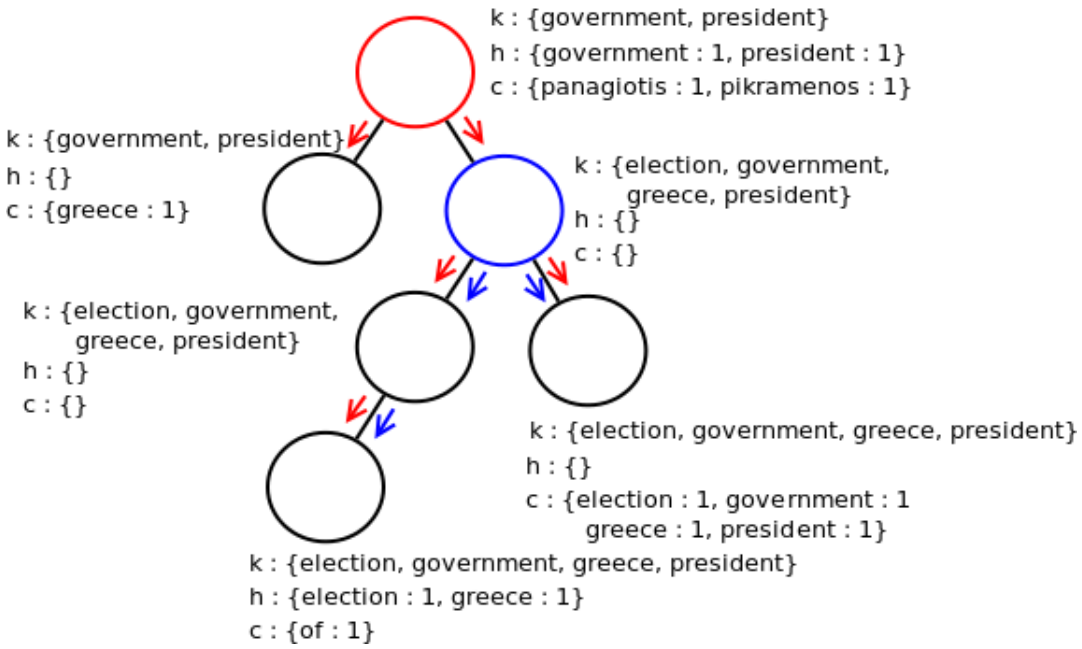


5 pav.: Raktinių žodžių veikimo sričių nustatymas

neretai ir atsiduria skirtingame medžio lygyje. Tokiais atvejais yra būtina surasti, kuris artimiausias tėvinis mazgas galimai būtų aibės k veikiamo pomedžio šakninis elementas. 5 pav. raudonai apvestame mazge matome netuščią aibę k, sudarytą iš vektoriaus h žodžių. Šiame mazge esančiame vektoriuje c egzistuoja vienas žodis *of*, tačiau jis netenkina minimalaus ilgio taisyklės, todėl negalime laikyti, kad raudonai pažymėtame mazge egzistuoja reikšmingo turinio, kurį galima būtų apibūdinti aibėje k esančiais žodžiais. Tokiu atveju tenka pakilti medžiu iki pirmo sutikto tėvinio elemento, kuriame, įskaitant ir jo vaikinius elementus, egzistuotų reikšmingo turinio. 5 pav. matome kaip algoritmas pakyla dviem lygiais aukštyn iki tėvinio mazgo, pažymėto mėlyna spalva. Algoritmas sustabdo tolimesnį kilimą medžiu aukštyn, kadangi šiuo atveju mėlyno mazgo pomedyje egzistuoja pakankamai prasmingo turinio: *election, government, greece* ir t.t. Taigi mėlynai pažymėtas tėvinis mazgas paveldi raudoname mazge esančius k aibės žodžius, o tokiu būdu galimai surandama aibės k veikimo sritis.

Raktinių žodžių paveldėjimas. Ankstesniame etape nustačius visų k aibių veikiamus pomedžius belieka k aibės turiniu pasidalinti su vaikiniais elementais. Kuriame nors mazge esanti aibė k turi prasmę ne tik tame konkrečiame mazge, bet ir to mazgo visuose vaikiniuose elementuose, nesvarbu kuriame medžio lygyje jie bebūtų.

Pavyzdžiui 6 pav. raudono mazgo raktiniai žodžiai iš aibės k turi prasmę ne tik raudoname mazge esančiame vektoriuje c, bet ir visame mazgo pomedyje: t.y. vaikiniuose elementuose, mėlyname mazge bei jo vaikiniuose mazguose ir t.t. 6 pav. spalvotomis rodyklėmis



6 pav.: Raktinių žodžių paveldėjimas

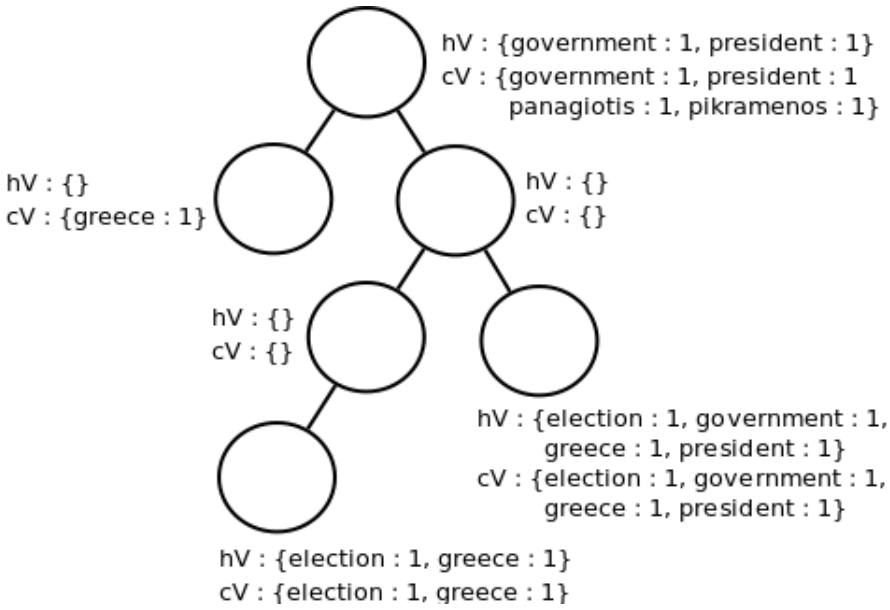
pažymėta, iš kuriuose mazguose esančių aibių k yra paveldimi raktiniai žodžiai bei kurie mazgai kokių mazgų raktažodžius paveldi. Šis raktinių žodžių analizės etapas atliekamas paprastumo ir patogumo sumetimais, kad tolimesniuose turinio apdorojimo etapuose analizuojant h ir c turinį nereikėtų pakartotinai kilti medžiu aukštyn, siekiant surinkti visų tėvinių mazgų k aibių turinį, t.y. pakartotinai nustatyti, kokie raktiniai žodžiai reikšmingi konkrečiame mazge.

4.1.4. Turinio analizė

Šis turinio analizės etapas sudarytas iš dviejų žingsnių: h ir c vektorių turinio apdorojimo ir atitinkmai naujų vektorių hV bei cV sudarymo bei šių vektorių paveldėjimo.

Turinio apdorojimas. Šiame žingsnyje atliekame h ir c vektorių žodžių analizę. Iš vektorių h ir c pašalinami skaičiai bei žodžiai, kurių ilgis maženis nei reikalaujama (pvz. 3 simboliai). Likę žodžiai papildomai apdorojami. Tam gali būti pasitelkiami įvairūs metodai pagal pageidavimą, pvz. nustatomos žodžių šaknys. Galiausiai du atskiri h ir v vektoriai apjungiami į vieną cV vektorių, sudarytame iš konkrečiame mazge esančio tekstinio turinio. cV kaip ir c vektorius yra požymių dažnumo vektorius, kuriame saugomos požymio – pasikartojimo dažnumo poros.

7 pav. matome naujai sudarytą vektorių cV, susidedantį iš h ir c vektorių elementų sumos, tačiau termai išliko identiški, nes jokie papildomi žodžių apdorojimo algoritmai (pvz. termų apjungimas pagal žodžio šaknis) šiame pavyzdyje nebuvo pritaikyti. 4 lygyje esančiame

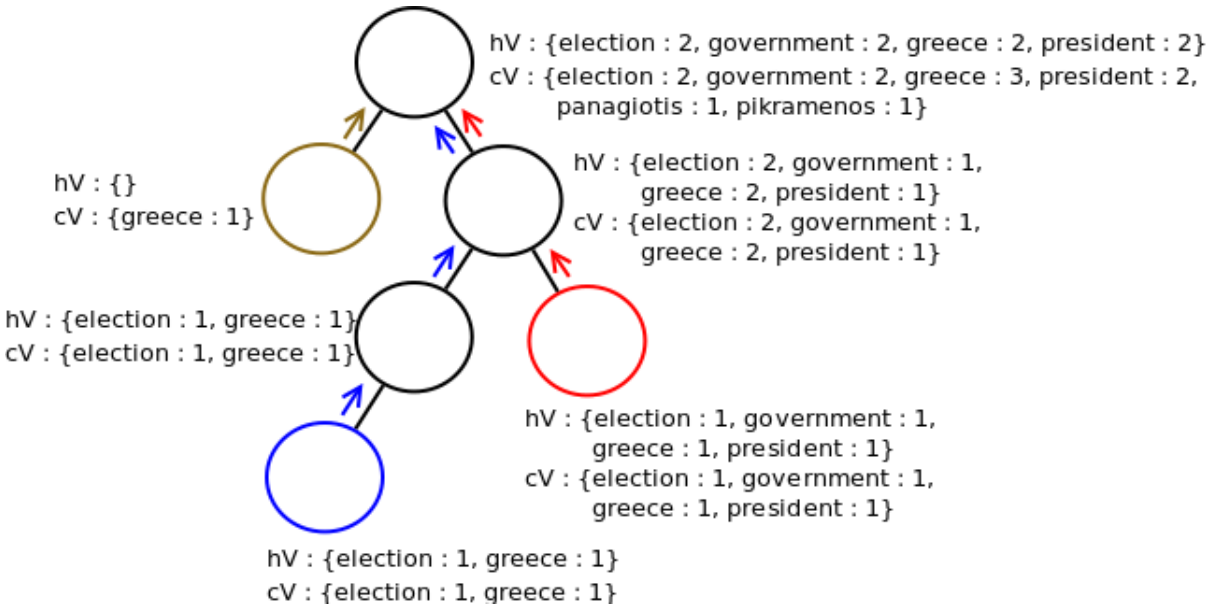


7 pav.: Turinio analizė

mazge galime pastebėti, kad apskaičiuotame cV vektoriuje nebeliko požymio *of*, kadangi jis netenkino minimalaus reikalaujamo žodžio ilgio, todėl buvo pašalintas. Be cV vektoriaus taip pat sudaromas ir hV vektorius, kuris kaip ir cV yra dažnumo vektorius, sudarytas tik iš raktinių požymių pasitelkiant ankstesniuose etapuose sudaryta raktinių požymių aibe k bei nauju vektoriumi cV. Šiame apdorojimo etape galima pastebėti, kad hV vektorius yra cV vektoriaus poaibis, o atitinkami sutampančių požymių dažnumo įverčiai yra lygūs (7 pav.).

Turinio paveldėjimas. Galime pastebėti, kad iki šiol hV ir cV požymių vektoriai yra sudaryti tik iš tuose mazguose, kuriuose jie apskaičiuojami, egzistuojančių požymių ir juose neatsispindi joks turinys saugomas vaikiniuose mazguose esančiuose hV ir cV vektoriuose. Kadangi sekančiuose etapuose teks analizuoti ne vieną medžio pomedį (medžio pomedžių skaičius yra lygus medžio mazgų skaičiui) kiekvieną atskirai, todėl yra aktualu sumažinti perteklinių skaičiavimų kiekį ir išvengi pakartotino hV ir cV požymių surinkimo iš vaikinių mazgų. Tam pasiekti pasitelkiame požymių vektorių hV ir cV paveldėjimą iš vaikinių mazgų, t.y. mazgo hV ir cV vektoriai apjungiami su vaikinių mazgų hV ir cV vektoriais ir taip gaunami suminiai hV ir cV vektoriai. To pasekoje kiekviename mazge hV ir cV vektoriai saugo požymius, surinktus iš konkretaus mazgo pomedžio.

8 pav. skirtingų spalvų rodyklėmis parodyta, kaip iš žemiausio lygio mazgų požymiai kyla medžio struktūra aukštyn iki šakninio medžio elemento ir prisijungia prie pakeliui sutiktų hV ir cV vektorių. Galiausiai matome, kad šakniniame medžio elemente esantys hV ir cV vektoriai yra sudaryti iš visame medyje randamų požymių.



8 pav.: Turinio paveldėjimas

4.1.5. Dokumentų indeksavimas

Požymių vektorių sudarymas. Iki šiol buvo apskaičiuoti du dažnių vektoriai hV ir cV, tačiau jie labiau pritaikyti skaičiuoti suminius vektorių įverčius pomedžiuose ir dar nėra paruošti galutiniam kategoriniam įverčiui apskaičiuoti, pasitelkiant bet kurį pageidaujamą klasifikatorių. Indeksavimo proceso metu vektoriai hV ir cV yra apjungiami ir sudaromas vienas vektorius v. Kadangi kol kas vektoriuje v turime tik termo-dažnio poras, tikslinga apskaičiuoti svorinius termų įverčius w_t. Konkretaus termo svoriui w_t apskaičiuoti pasinaudosime ne dažniausiai naudojamu $TF \cdot IDF$ metodu, o pasiūlysime naudoti $TF \cdot CDW$ modifikaciją (18 formulė):

$$w_t = TF_t \cdot CDW_{t,c}, \tag{18}$$

kur TF_t – tai lokaliai normalizuotas, konkrečiame analizuojamame dokumente (šiuo atveju dokumento mazge) rasto termo, logaritminis įvertis, o $CDW_{t,c}$ – termo t svoris dokumentui priklausančioje klasėje c. CDW įvertis naudojamas vietoj standartiniame $TF \cdot IDF$ metode vartojamo IDF įverčio, tik skirtingai nuo jo, CDW yra skirtas spręsti dviejų klasių problemas ir vartojamas apskaičiuoti termo priklausomybės klasei, o ne priklausomybės konkrečiam dokumentui įvertį. Logaritminis įvertis TF_t, apskaičiuojamas pagal formulę:

$$TF_t = \begin{cases} \frac{1+\log tf_t}{\sum_{t \in v} TF_t} & tf_t > 0 \\ 0 & tf_t = 0 \end{cases} \tag{19}$$

kur tf_t – termų kiekis analizuojamame dokumente ar kaip šiuo atveju – dokumento mazge. Taip pat formulėje pasitelkiamas logaritminis dažnio įvertinimas, kadangi termo svoris nėra tiesiogiai proporcingas jo kiekiui, pvz. termas pasitaikantis 10 kartų nėra 10 kartų svarbesnis už termą, pasitaikantį tik kartą. Galiausiai gautas logaritminis įvertis yra lokaliai normalizuojamas ir pasitelkiant 18 formulę apskaičiuojami galutiniai požymių svoriai w_t.

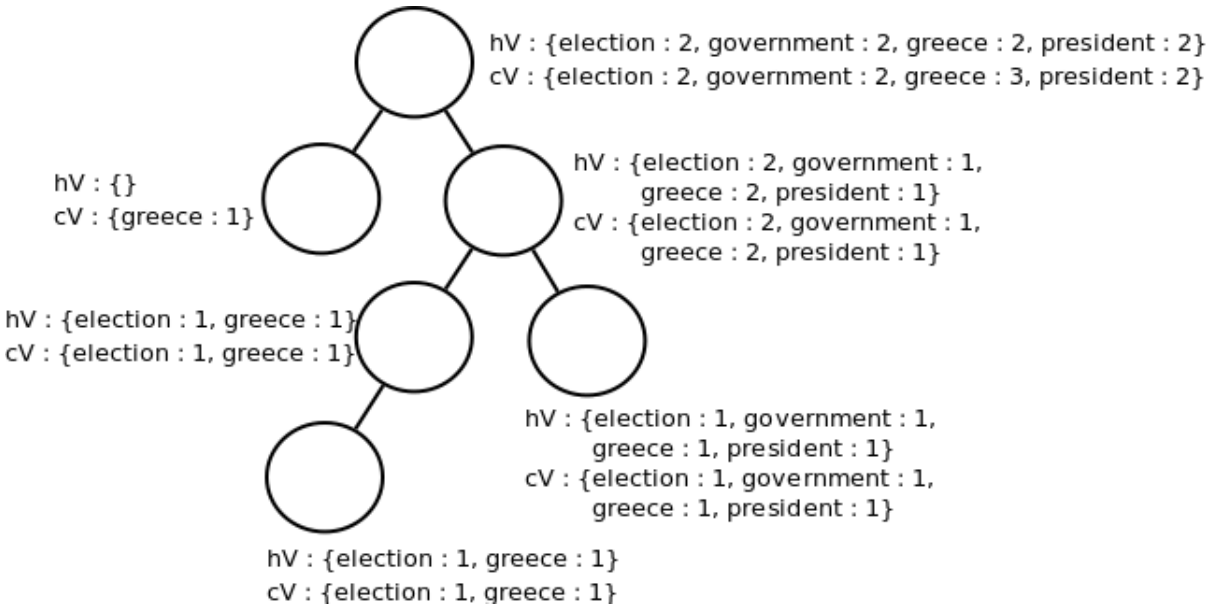
Pabaigai, siekiant, kad iš apskaičiuotų w_t įverčių sudarytas vektorius v būtų nepriklausomas nuo analizuojamo dokumento dydžio, privalome normalizuoti gautą vektorių v. Tai leistų adekvačiai palyginti ne vienodo ilgio dokumentus. Vektorius v normalizuojamas padalinant kiekvieną jo komponentę w_t iš vektoriaus ilgio. Tam mes pasitelksime kosinuso normalizavimo funkciją $1/\sqrt{\sum w_i^2}$, kuri visus vektorius nepriklausomai nuo jų ilgio pavers vienetiniais vektoriais.

4.1.6. Klasifikatoriaus apmokymas

Šiame darbe pasinaudosime atraminių vektorių klasifikatoriumi, aprašytu 2.2.2 skyriuje. Šio etapo metu pasitelkiant anksčiau aprašytus požymių apskaičiavimo ir eliminavimo metodus visi apmokymo aibės D dokumentai d_c transformuojami į indeksinius požymių vektorius $v_{d,c}$. Dokumentas d reprezentuojamas tik pagal šakninio medžio mazgo požymių vektorių v. Dokumentai gali būti priskiriami vienai iš dviejų kategorijų c: teigiamai arba neigiamai. Pasitelkiant gautus požymių vektorius $v_{d,c}$ yra apmokomas SVM klasifikatorius ir sudaromas klasifikavimo žodynas, sudarytas iš termų aibės bei kiekvienai termo priklausomybės klasei apskaičiuotų $w_{t,c}$ ir $CDW_{t,c}$ svorinių įverčių.

4.2. Internetinio turinio filtravimo algoritmas

Internetinio dokumento klasifikavimas taip pat kaip ir apmokymo algoritmas neapseina be tokių pačių požymių paieškos ir analizės žingsnių, aprašytų 4.1.1 - 4.1.4 skyriuose: *HTML* dokumento medžio transformacijos (4.1.1 skyrius), žodžių dažnumų vektorių sudarymo (4.1.2 skyrius), raktinių žodžių analizės (4.1.3 skyrius) bei turinio analizės (4.1.4 skyrius). Vienintelis papildomas žingsnis, kuris atliekamas klasifikuojant internetinį dokumentą, tai apmokymo metu sudaryto klasifikavimo žodyno D panaudojimas eliminuojant medžio mazguose esančiuose vektoriuose hV ir cV nežinomus požymius. Iš 8 pav. pavaizduoto medžio eliminavus nežinomus požymius, gauname naują medį (9 pav.), kurio šakniniame mazge galime pastebėti, kad cV vektoriuje nebeliko tokių klasifikatoriui nežinomų termų kaip *panagiotis* ir *pikramenos*.

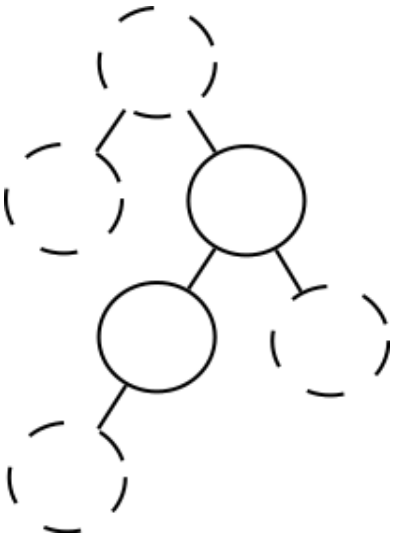


9 pav.: Nežinomų požymių pašalinimas pasitelkiant klasifikavimo žodyną D

4.2.1. Mazgų analizė

Šiame turinio analizės etape nustatome, kurie medžio mazgai yra aktualesni ir svarbesni. Svarbiais mazgais laikysime tuos medžio mazgus, kuriuose egzistuoja neigiamo pobūdžio turinio, t.y. h arba c požymių vektoriuose egzistuoja bent vienas požymis, kuriam SVM klasifikatoriaus apmokymo metu buvo suteiktas neigiamos vertės svoris. Tokiu būdu nustatome potencialius mazgus, kurių pomedžiuose gali būti saugomas neigiamo pobūdžio turinys ir atmetame įtakos neturinčius nereikšmingus mazgus. Nereikšmingų mazgų kategorija priklausys tik nuo aukščiau esančių reikšmingų mazgų kategorijų, todėl klasifikuoti šių mazgų nėra tikslo, o užtenka nustatyti tik potencialiai įtartinų medžio mazgų kategorijas. Toks mazgų suskirstymas į reikšmingus ir nereikšmingus leidžia išspręsti kontekstinį klasifikavimo uždavinį, kurio tikslas nustatyti atskirtų turinio blokų klasę ir blokuoti tik dalį turinio nepaliečiant likusio. Taip pat akivaizdu, kad nustatytų įtartinų mazgų kiekis visada maženis arba lygūs visų mazgų skaičiui, tokiu būdu algoritmas klasifikuodamas tik šiuos mazgus išvengs bereikalingo likusių mazgų klasifikavimo ir taip sutaupys laiko bei išteklių.

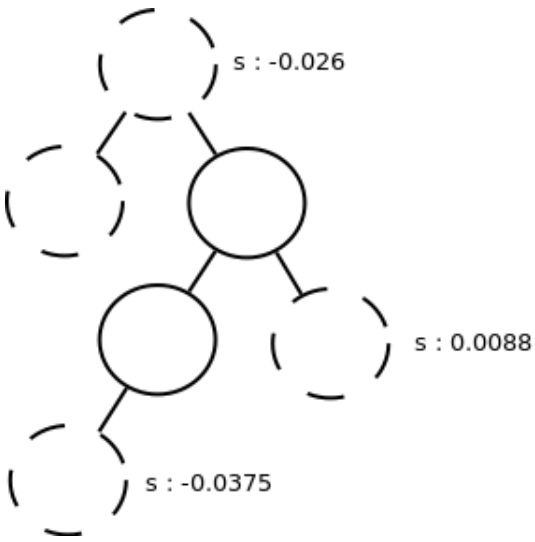
Tarkime, kad požymiai *greece* ir *government* SVM klasifikatoriaus apmokymo metu įgavo neigiamus įverčius. Tokiu atveju gauname 10 pav. medį, kuriame punktyrine linija pažymėti medžio mazgai, kurių h ir c vektoriuose buvo rasti šie požymiai. Dėl to užuot klasifikavus šešis pomedžius užtenka klasifikuoti keturis.



10 pav.: Aktualių mazgų nustatymas

4.2.2. Mazgų priklausomybės klasei įverčių skaičiavimas

Šio internetinio puslapio klasifikavimo etapo tiklas – nustatyti svarbių medžio mazgų kategorijas. Klasifikuojami ne visi ankstesniame etape išskirti mazgai. Algoritmas ignoruoja tuos mazgus, kuriuose egzistuoja per mažai turinio, kad būtų galima patikimai nustatyti mazgo kategoriją. Mazgai, kurių *HTML* turinyje neegzistuoja objektų (3 lentelė), o cV vektoriaus unikalių požymių kiekis mažesnis nei 2^1 bei požymių dažnių suma mažesnė nei 3^1 arba jei *HTML* turinyje egzistuoja bent vienas objektas, o požymių dažnių suma mažesnė nei 2^1, yra ignoruojami, o jų kategorija nėra skaičiuojama. 11 pav. galime pastebėti, kad iš keturių esminių mazgų buvo apdoroti tik trys, o vienas ignoruotas, kadangi jame buvo tik vienas žodis, iš kurio spręsti apie turinio bloko kategoriją yra netikslinga.



11 pav.: Mazgų įverčių skaičiavimas

[1]Kalibruojami parametrai

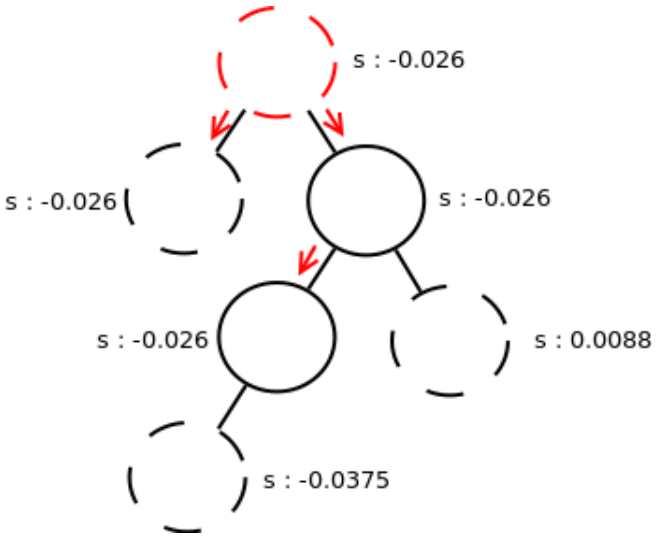
Mazgų įverčiai s apskaičiuojami dviem žingsniais: pirmiausiai iš vektorių hV ir cV sudaromas galutinis požymių vektorius ir galiausiai pasitelkiant klasifikavimo žodyną apskaičiuojamas įvertis s.

Požymių vektorių sudarymas. Detalesnius principus kaip iš termų dažnumų vektorių hV ir cV sudaromas požymių vektorius v aptarėme 4.1.5 skyriuje.

Įverčių s skaičiavimas. Galiausiai, turint galutinį konkretaus mazgo požymių vektorių v bei turint apmokymo metu sudarytą žodyną D belieka pasitelkiant SVM klasifikatoriaus sandaugos funkcija (11 formulė) apskaičiuoti galutinį mazgo įvertį s. Apskaičiavus visų svarbiausių medžio mazgų įverčius s gauname 11 pav. pavaizduotą medį.

4.2.3. Klasifikavimo įverčių paveldėjimas

Ankstesniame etape apskaičiavome trijų iš šešių mazgų kategorijas. Beliko išsiaiškinti likusių trijų mazgų kategorijas. Šiame etape darome elementarią prielaidą, kad vaikiniai mazgai, kurie neturi nustatyto kategorijos įverčio s, yra tos pačios kategorijos kaip ir pirmas tėvinis elementas, kuris ją turi, kadangi klasifikuojant tėvinį mazgą buvo panaudoti duomenys ir iš vaikinių mazgų, todėl jie natūraliai turėtų priklausyti tai pačiai kategorijai. Jei toks tėvinis elementas neegzistuoja, traktuojama, kad įvertis s lygus 0, t.y. mazgas priklauso neutraliai kategorijai. Norint įgyvendinti šią įdėją, užtenka atlikti mazgų kategorijų įverčių s paveldėjimą.



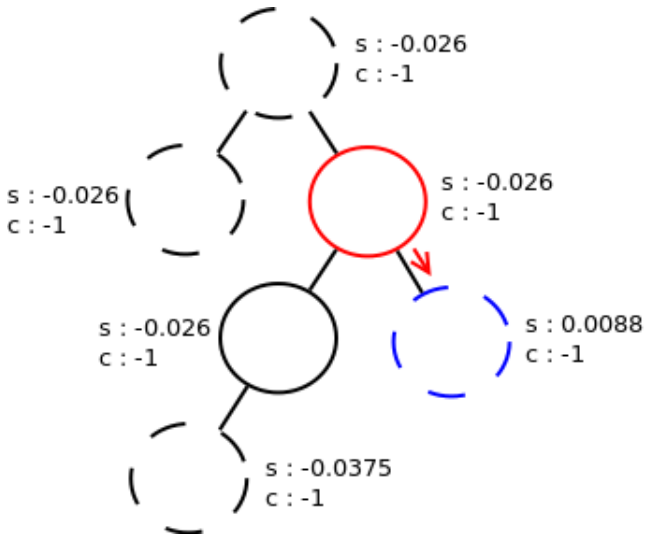
12 pav.: Įverčių s paveldėjimas

12 pav. rodyklėmis pavaizduota kaip raudona spalva pažymėto šakninio medžio elemento įvertis s paveldimas visų žemiau esančių vaikinių elementų, kurie dar neturėjo apskaičiuoto

įverčio. Paveldėjimo proceso metu sutikus įvertį s turintį vaikinį elementą, paveldėjimo procesas toje atšakoje sustoja.

4.2.4. Medžio mazgų preliminarių klasių nustatymas

Apskaičiavus mazgų kategorijų įverčius s, galime nustatyti, kuriai klasei priklauso kiekvienas mazgas. Jei apskaičiuotas kategorijos įvertis s yra neigiamas, tai traktuojame, kad mazgo kategorija c yra neigiama, priešingu atveju – teigiama. Jei gauta kategorija yra teigiama, tačiau egzistuoja toks tėvinis elementas, kurio kategorija yra neigiama, o įvertis $|s|$ yra didesnis už nagrinėjamo mazgo įvertį $|s|$, tuomet mazgas paveldi tėvinio elemento neigiamą kategoriją c.



13 pav.: Mazgų preliminarių klasių c nustatymas

Daroma prielaida, kad kuo mazgo įvertis s toliau nuo nulinės reikšmės, tuo mazgo kategorija yra labiau teigiama arba neigiama, o kadangi tėvinis mazgas yra labiau neigiamos kategorijos nei vaikinis mazgas teigiamos, tai galime teigti, kad ir vaikinis mazgas yra galimai neigiamos kategorijos. 13 pav. matome, kaip silpnai teigiamos kategorijos mėlynas mazgas paveldi stipriau neigiamos kategorijos raudono mazgo neigiamą klasę c.

4.2.5. *HTML* elementų užimamo ploto analizė

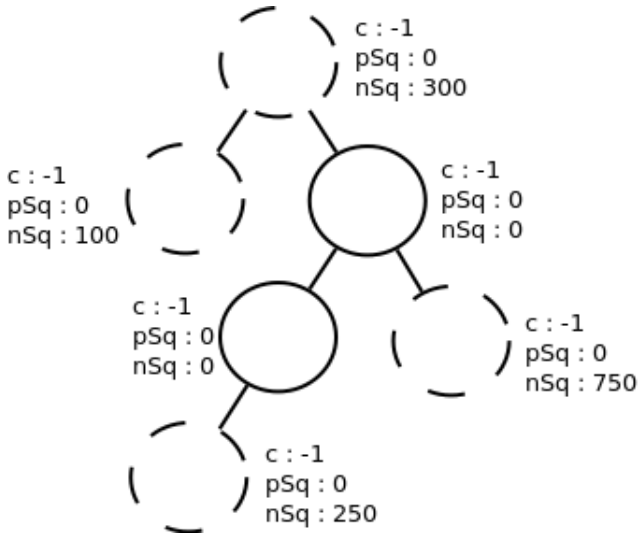
Sekantis dokumento klasifikavimo etapas – tai *HTML* dokumento elementų užimamo ploto analizė, sudaryta iš dviejų žingsnių: kiekvieno medžio mazgo elementų užimamo ploto skaičiavimo ir užimamo ploto paveldėjimo etapų.

Plotų analizė. Šiame etape suskaičiuojame, kokį plotą užima kiekviename medžio mazge esantys esminiai *HTML* elementai (5 lentelė).

Formų elementai						
BUTTON	*DATALIST*	*INPUT*	*SELECT*			
Objektai						
APPLET	*AUDIO*	*CANVAS*	*EMBED*	*IMG*	*OBJECT*	*VIDEO*
Kiti elementai						
TEXT_NODE						

5 lentelė: *HTML 4* ir *HTML 5* elementai, užimantys reikšminį plotą

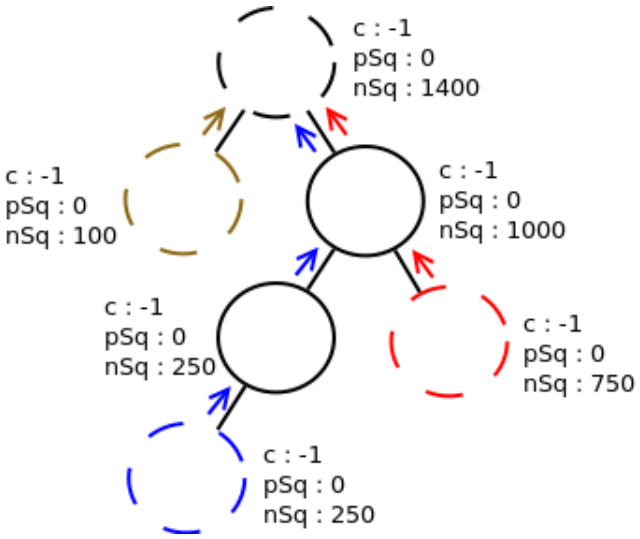
Priklausomai nuo kategorijos susumuojamas teigiamos ir neigiamos kategorijų elementų užimamas plotas ir žymimas atitinkamai pSq ir nSq (14 pav.). Nepriklausomai nuo objekto geometrinės formos plotas skaičiuojamas pasitelkiant gaubiančiuosius stačiakampius. Iš analizės buvo pašalinti tokie elementai kaip *TEXTAREA, DIV* ir pan., kadangi juose informacijos gali ir nebūti, o pasitelkus gaubiančiuosius stačiakampius galima apskaičiuoti nenulinės reikšmės plotą, kuris galėtų iškreipti galutinius klasifikavimo rezultatus.



14 pav.: Teigiamos ir neigiamos kategorijų elementų užimamo ploto analizė

Plotų paveldėjimas Šio žingsnio tikslas – apskaičiuoti kokį plotą kokioje medžio atšakoje užima teigiamas ir neigiamas turinys. Kaip ir turinio paveldėjimo atveju, taip ir šiuo atveju pageidaujamą rezultatą galime pasiekti pasitelkus pSq ir nSq įverčių paveldėjimą iš vaikinių mazgų, t.y. mazgo pSq ir nSq įverčiai sudedami su vaikinių mazgų pSq ir nSq įverčiais, taip gaunami suminiai pSq ir nSq įverčiai. To pasekoje kiekviename mazge pSq ir nSq įverčiai saugo galutinį plotą, kurį užima teigiamas ir neigiamas turinys konkrečiame mazgo pomedyje.

15 pav. skirtingų spalvų rodyklėmis parodyta, kaip iš žemiausio lygio mazgų pSq ir nSq įverčiai kyla medžio struktūra aukštyn iki šakninio medžio elemento ir yra pridedami prie pakeliui sutiktų tėvinių mazgų pSq ir nSq įverčių. Galiausiai matome, kad šakniniame medžio elemente esantys pSq ir nSq įverčiai parodo, kokį plotą užimą vienokio ir kitokio pobūdžio turinys visame dokumente.



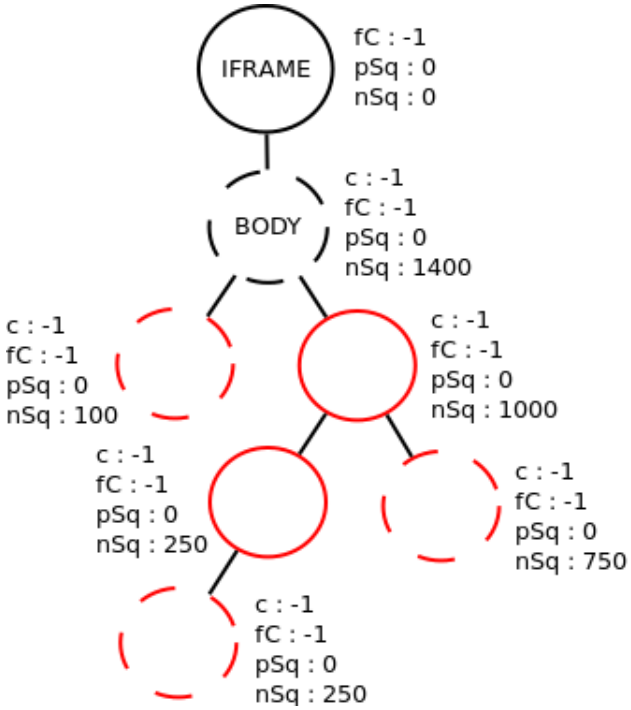
15 pav.: Pomedžiuose esančių teigiamos ir neigiamos kategorijų elementų užimamo ploto analizė

4.2.6. Galutinių kategorijų nustatymas

Šį analizės etapą galima suskirstyti į dvi lygiagrečiai vykdomas dalis: kritinių mazgų bei pomedžių kategorijų analizės.

Kritinių mazgų plotų analizė. Šiame analizės etape pasitelkiant *HTML* turinio užimamą plotą pSq ir nSq siekiama nustatyti, kiek kokio turinio egzistuoja ir kokiai kategorijai galima priskirti *BODY* mazgus, atitinkančius *BODY* elementus *HTML* medžio struktūroje. *BODY* mazgai – tai kaip šliuzai į visą pagrindinio puslapio arba įterptinių, naudojančių *IFRAME* žymas, išorinių dokumentų turinį. Kokiai kategorijai galima priskirti *BODY* elementą priklauso nuo to, kiek ir kokio pobūdžio turinio algoritmas aptiko konkretaus elemento pomedyje. Kategorija gali būti viena iš trijų: teigiama, neigiama arba neutrali. Jei pomedyje aptinkamo neigiamo turinio užimamas plotas nSq užima santykinai labai mažą dalį nuo viso turinio ploto ($pSq+nSq$), galima teigti, kad visas pomedis greičiausiai yra teigiamo pobūdžio, ir visi tame pomedyje esantys mazgai irgi yra teigiami nepriklausomai nuo 4.2.4 skyriuje aprašytame etape nustatytų preliminarių kategorijų c. Taip daroma dėl tos, kad

pomedyje gali pasitaikyti smulkių turinio blokų, kurie klasifikavimo proceso metu buvo identifikuoti kaip neigiami, o jų įverčiai taip pat gali būti tik silpnai neigiami. Tokių nedidelių turinio blokų turinys nors ir įgavęs neigiamą įvertį iš tiesų neretai būna neutralaus pobūdžio, todėl pasitelkus tam tikrą slenkstį BP ir jo neviršijant galime eliminuoti klasifikavimo metu padarytas klaidas.



16 pav.: Galutinių klasių nustatymas kritiniuose mazguose

BP įvertis – tai procentinė neigiamo turinio elementų užimamo ploto dalis nuo viso dokumento turinio užimamo ploto. Be BP slenksčio taip pat naudojame ir kitą slenkstį BN, kurio paskirtis yra priešinga pirmajam. BN slenkstis nusako minimalų neigiamo turinio užimamo ploto procentą, kurį viršijus teigiama, kad pomedyje visas esantis turinys yra neigiamo pobūdžio ir nėra tikslo rodyti daug neigiamo turinio turinčio turinio bloko, nors ir jame egzistuoja silpnai teigiamo turinio turinčių sričių, kurių užimamas plotas yra nereikšmingai mažas. Tokiu būdu daug neigiamo turinio turintys blokai yra blokuojami ir nėra tikslo rodyti nereikšmingas smulkias jų iškarpas, o tikslinga jas taip pat pašalinti. BP ir BN slenksčiai tarpusavyje susiję ir gali būti abu lygūs arba $BP < BN$, kiti variantai negalimi. Slenksčiai gali būti atvaizduojami vienoje atkarpoje tarp 0 ir 100 ir dalinti atkarpą į du-tris rėžius. Beliko nusakyti, kas įvyksta, kai rasto neigiamo turinio plotas patenka tarp BP ir BN. Tokiu atveju, pomedyje esantis turinys traktuojamas kaip mišraus pobūdžio ($fC = 0$), ir visų jame esančių mazgų klasė fC priklausys nuo ankstesniame etape apskaičiuoto įverčio c. BP ir BN slenksčiai gali būti kalibruojami pagal vartotojo tolerancijos laispnį ir tik nuo jo prik-

lauso kokie $[0; BP]$, $(BP; BN)$ ir $[BN; 100]$ rėžiai bus naudojami. Galime pastebėti, kad jei $BP = BN$, vidurinysis mišraus turinio rėžis gali būti eliminuotas, o visi dokumente esantys mazgai klasifikuojami tik kaip absoliučiai teigiami arba neigiami, o kontekstinis klasifikavimas, kurio metu, pavyzdžiui, tarp daug teigiamo turinio pasitaikytų pašalintų neigiamo turinio turinčių sričių, būtų išjungtas.

16 pav. matome, kad $BODY$ elementas įgavo neigiamą galutinę kategoriją fC, nes visas dokumente esantis turinys buvo neigiamo pobūdžio ($pSq = 0$). Taip pat jei $BODY$ mazgo tėvinis mazgas yra *META* arba *IFRAME*, tai ir jis paveldi galutinę tiesioginio vaikinio $BODY$ elemento klasę fC pagal tokias taisykles (20 formulė):

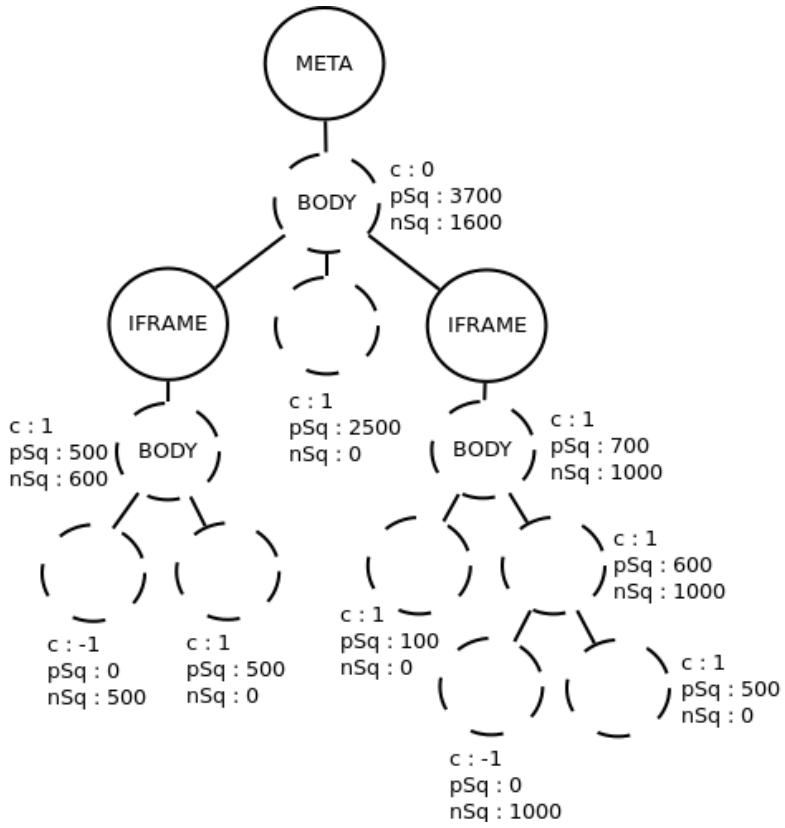
$$fC_{e \in (META,\ IFRAME)} = \begin{cases} 1 & nSqP \leq BP \\ -1 & nSqP \geq BN \\ 0 & kitu\ atveju \end{cases} \tag{20}$$

kur $nSqP$ yra neigiamo turinio užimamo ploto santykis su viso turinio užimamu plotu ir apskaičiuojamas pagal 21 formulę:

$$nSqP = \frac{nSq}{nSq + pSq} \tag{21}$$

Pomedžių plotų analizė. Sekančiame žingsnyje algoritmas bando nustatyti likusių (ne *META*, ne *IFRAME* ir ne $BODY$) dokumento medžio mazgų galutines klases. Tam tikslui taip pat yra analizuojami plotų suminiai įverčiai pSq ir nSq bei pasitelkiamas trečias slenkstis PN, kuris nusako, koks minimalus neigiamo turinio procentas turi būti analizuojamame pomedyje, kad visas pomedis bei jo mazgai būtų traktuojami kaip neigiami ir įgautų neigiamą galutinę klasę fC. 16 pav. matome kaip raudonai pažymėti medžio mazgai dėl juose esančio neigiamo turinio įgavo neigiamas kategorijas fC. Šio žingsnio esminė idėja yra eliminuoti daug neigiamo turinio turinčius turinio blokus, kartu apimant ir šiektiek jų aplinkos. Pašalinto aplinkinio galimai teigiamo turinio sąskaita atsiranda mažesnė tikimybė, kad pomedžio artimoje aplinkoje esantis neigiamas turinys išliks matomas.

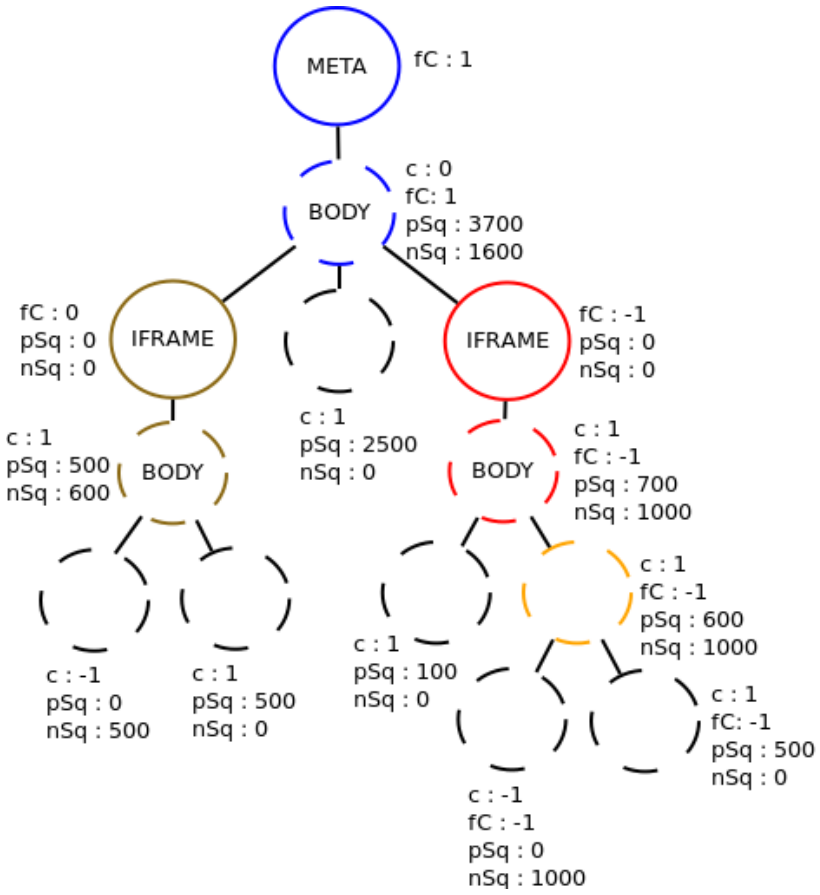
16 pav. matome, kad absoliučiai visi medžio mazgai įgavo galutines klases fC, dėl to sekantis žingsnis, kurio metu nustatoma mazgų, neįgavusių galutinės fC įverčio kategorijos, net nebūtinas, nes nebeliko tokių mazgų su neįvertinta fC reikšme. Dėl šios priežasties pateiksime didesnio ir įvairesnio medžio pavyzdį, kuriame pamatysime visas šiame etape galinčias įvykti situacijas. Tarkime, kad turime toki medį (17 pav.), kuriame turime apskaičiuotus įvairiausius c, pSq ir nSq įverčius. Belieka pagal šiame etape aprašytus princi-



17 pav.: Galutinių klasių nustatymas kritiniuose mazguose

pus mazgams nustatyti galutinius kategorijų įverčius fC. Tarkime, kad vartotojas pasirinko BP ir BN įverčius atitinkamai 35% ir 55%. Tokiu atveju analizuodami $BODY$ elementus (18 pav.) gauname, kad raudonas $BODY$ elementas dėl per didelio neigiamo turinio užimamo ploto įgyja neigiamą kategoriją ($fC = -1$), mėlynas – dėl didelio teigiamo ploto įgyja teigiamą kategoriją ($fC = 1$), o rudas elementas neįgauna jokios kategorijos, kadangi šiame mazge neigiamo ir teigiamo turinio procentas yra apylygis ir patenka į $(BP; BN)$ rėžį. Nustačius, kokioms galutinėms kategorijoms priklauso $BODY$ elementai, belieka pagal 20 formulę nustatyti kategorijas į jų tėviniams elementams $IFRAME$ bei $META$. Pasitelkus minėtą taisyklę gauname, kad $META$ elementas įgyja teigiamą mėlyno $BODY$ elemento kategoriją, raudonas $IFRAME$ elementas – neigiamą, o likęs rudas $IFRAME$ elementas – neutralią kategoriją.

Taip pat, pasitelkę PN įvertį pagal antrą šio etapo žingsnį nustatome neigiamo turinio turintiems pomedžiams galutines kategorijas fC. Tarkime, kad vartotojo pasirinktas slenkstis PN yra lygus 60%. Tuomet galime pastebėti, kad egzistuoja vienintelis toks daugiau nei 60% neigiamo turinio turintis mazgas, pažymėtas oranžine spalva, todėl šis mazgas bei visi jo pomedyje esantys vaikiniai elementai įgauną neigiamą fC kategoriją. Galiausiai matome, kad turimame medyje beliko tik penki mazgai, kurie dar neįgavo galutinės kategorijos fC



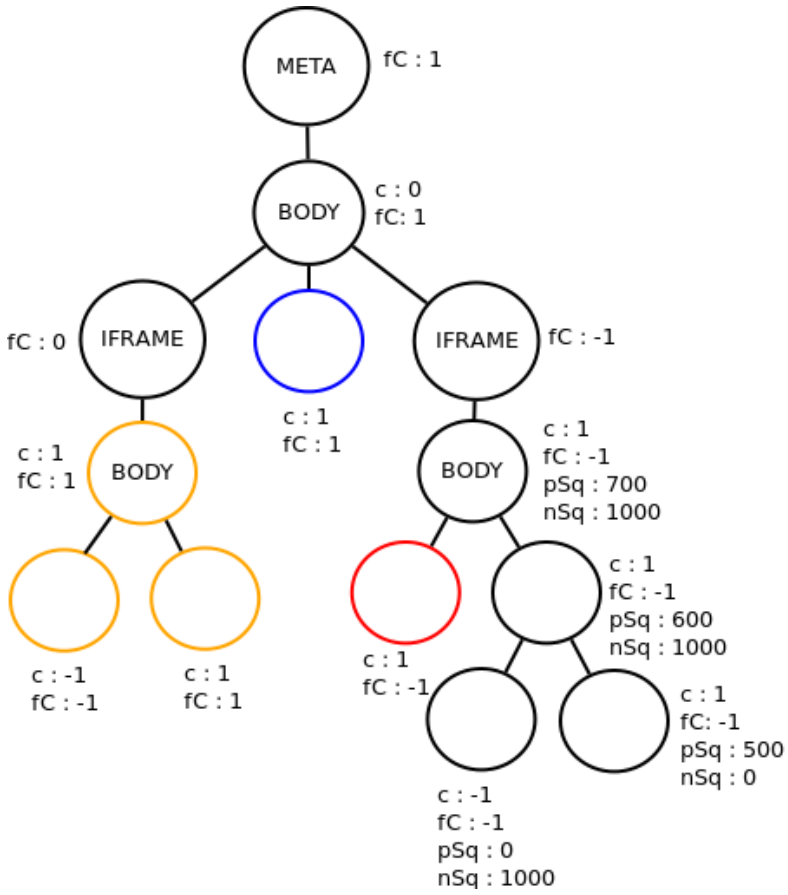
18 pav.: Galutinių klasių nustatymas kritiniuose mazguose

įverčio.

4.2.7. Galutinių mazgų klasių paveldėjimas

Tai paskutinis *HTML* dokumento analizės etapas, kurio metu likusiems mazgams, kurie ankstesniame etape neįgavo galutinio kategorijos įverčio fC, yra nustatoma galutinė kategorija priklausomai nuo arčiausiai esančių tėvinių *BODY, IFRAME* ir *META* mazgų. fC įverčio neturintys mazgai paveldi pirmo sutikto tėvinio elemento, įgavusio galutinę klasę fC, kategoriją, bet tik jei ji buvo neigiama ($fC = -1$) arba teigiama ($fC = 1$). Jeigu tėvinio elemento kategorija buvo neutrali ($fC = 0$), tai vaikinių elementų kategorija priklauso nuo ankstesniuose etapuose įgautų preliminarių kategorijų c.

19 pav. matome, kad mėlynas mazgas įgavo galutinę teigiamą kategoriją, kadangi arčiausiai jo esančio tėvinio elemento kategorija fC buvo teigiama. Kadangi šio mazgo kaimyniniai mazgai *IFRAME* jau turėjo galutines kategorijas fC todėl jie *BODY* elemento kategorijos nepaveldėjo, tuo pačiu kategorijos nepaveldėjo ir šių kaimynų pomedžiuose žemiau esantys elementai. Raudonas mazgas paveldėjo tėvinio elemento neigiamą kategoriją fC nepaisant to, kad paties mazgo preliminari kategorija buvo c. Likę oranžiniai mazgai nepaveldėjo jokios



19 pav.: Galutinė mazgų plotų analizė

kategorijos, kadangi arčiausiai esančio tėvinio *IFRAME* elemento, turinčio apskaičiuotą fC įvertį, galutinė kategorija yra neutrali, ko pasekoje visų jos vaikinių elementų galutinės kategorijos sutampa su jų apskaičiuotomis preliminariomis kategorijomis c.

4.2.8. Neigiamo turinio pašalinimas

Nustačius galutines visų dokumento medžio mazgų kategorijas fC paskutinis turinio analizės etapas yra rasto neigiamo turinio turinčių pomedžių pašalinimas iš *HTML* dokumento. Šalinant elementus yra stengiamąsi neišdarkyti likusio turinio išdėstymo. Tam pasitelkiamas *CSS* stiliaus parametras *visibility:hidden*, kuris paslepia elementą, tačiau išlaiko jo užimamą plotą toki patį koks ir buvo. Šalinant elementus naudojamąsi tokiais principais:

- Jei šakninio *META* mazgo kategorija fC yra neigiama, tai visas puslapio turinys yra slepiamas ir rodomas perspėjamasis puslapis, pranešantis, kad vartotojo pageidaujamame puslapyje yra neigiamo pobūdžio informacija, o norint jį pamatyti būtina pateikti administratoriaus slaptažodį. Priešingu atveju atliekami žemiau esančiuose punktuose pateikti apdorojimo metodai.
- Iš visų elementų, esančių neigiamą įvertį fC turinčiuose mazguose, pašalinami *alt*, *title*,

href ir *src* atributai.

- Paslepiami neigiamą turinį turintys tekstiniai *HTML* elementai *TEXT_NODE*. Kadangi tekstinis turinys yra paprasčiausiai paslepiamas, jį vistiek galima pamatyti pasitelkiant naršyklės įrankius, todėl tekstinis turinys yra papildomai apdorojamas. Tekste esantys simboliai surūšiuojami abėcėlės tvarka išlaikant tarpo simbolių vietą ir kiekį, pvz. tokia simbolių eilutė *„Lorem ipsum dolor sit amet.“* yra keičiama nauja *„.adee iiLlm mmooo prr ssttu“*.

- Objektai (*APPLET, AUDIO, CANVAS, EMBED, IFRAME, IMG, OBJECT, VIDEO*) ir formų elementai (*BUTTON, DATALIST, INPUT, SELECT, TEXTAREA*) yra pakeičiami identiškų matmenų bei turinčių identiškus *CSS* stilius tuščiais *CANVAS* elementais.

- Nuorodos *A* keičiamos *CF* žymomis. Visas kitas žymos viduje esantis turinys šalinamas pagal aukščiau aprašytus principus.

- Jei iš konkretaus pomedžio pašalinti visi neigiamą turinį turintys elementai ir nebelieka nė vieno matomo elemento, visas pomedis taip pat yra slepiamas. Tokiu būdu pašalinami visi likę nereikalingi stiliaus elementai, pvz. elementus juosiantys rėmeliai ir panašūs pagražinimai, aktualūs tik esant matomam turiniui.

5. cFilter įskiepis *Mozilla Firefox* naršyklei

Siekiant praktiškai parodyti, kaip veikia pasiūlytas klasifikavimo algoritmas, *Javascript* programavimo kalba buvo realizuota *Mozilla Firefox* naršyklės įskiepio *cFilter beta* versija. Įskiepis taip pat gali būti naudojamas ne tik kaip puslapių klasifikatorius, bet ir kaip papildomas filtravimo ir apsaugos paslaugas teikiantis naršyklės priedas. Be to, *PHP* programavimo kalba buvo sukurta serverinė sistemos dalis, atsakinga už apmokymo duomenų priėmimą, saugojimą, apdorojimą, *SVM* klasifikatoriaus apmokymą bei atnaujinimų paruošimą.

5.1. Pagrindinės įskiepio savybės

Galėtume išskirti tokias pagrindines *cFilter* realizuotas ir planuojamas realizuoti ateityje įskiepio savybes.

Saugumo nuostatos. *cFilter* įskiepyje realizuotas ne vienas apsaugos metodas, skirtas apriboti (priedas nr. 8) nepageidaujamiems asmenis priėjimą ne tik prie *cFilter* parametrų, bet ir prie sisteminių puslapių bei nustatymų, tokiu būdu apribojant kuo daugiau galimybių paveikti *cFilter* darbą. Įskiepyje realizuoti tokie saugumo metodai:

- Slaptažodžiu apsaugotas priėjimas prie *cFilter* parametrų (priedas nr. 9).
- Galimybė slaptažodžiu apsaugoti priėjimą prie *Firefox* nustatymų (priedas nr. 10).
- Slaptažodžiu apsaugotas priėjimas prie *Firefox* įskiepių diegimo vedlio.
- Slaptažodžiu apsaugotas priėjimas prie kitų kritinių sisteminių *Firefox* puslapiu (priedas nr. 11):
  - leidimų tvarkyklė (angl. *Permissions Manager*)
  - įskiepių tvarkyklė (angl. *Add-ons Manager*)
  - papildomi nustatymai (*about:config*)
  - kiti su svarbiais *Firefox* nustatymais susiję puslapiai
- Galimybė atkurti pamirštą slaptažodį.
- Slaptažodžio šifravimas *SHA512* algoritmu, dėl ko jo išgavimo tikimybė yra minimali.

Patikimų ir nepatikimų puslapių sąrašai. Galutinėje *cFilter* versijoje pasitelkiant nuorodų šablonus[1] planuojama suteikti vartotojams galimybę blokuoti priėjimą prie pasirinktų puslapių, nuotraukų, išorinių *Javascript* failų, iterptinių puslapių (*IFRAME*), kitų nepageidaujamų informacijos šaltinių.

Laikinas patikimų puslapių sąrašas. *cFilter* įskiepiui užblokavus nepageidaujamo pobūdžio turinio puslapį, išlieka galimybė įvedus prisijungimo slaptažodį laikinai pamatyti įtartiną puslapį (priedas nr. 14). Tokiu atveju puslapis įdedamas į laikinai patikimų puslapių sąrašą, kuris būna aktyvus iki kito *Firefox* naršyklės paleidimo momento, nes pirminio *cFilter* įskiepio užsikrovimo metu sąrašas saugumo sumetimais yra išvalomas.

Licencijavimas ir bendri nustatymai. Kad *cFilter* įskiepis pradėtų veikti, kiekvieną įskiepio kopiją būtina užregistruoti *cFilter* sistemoje (priedas nr. 12). Registracijos metu užtenka pateikti tik savo elektroninio pašto adresą. Sugeneruotas licencijos numeris atsiunčiamas vartotojui elektroniniu paštu. Licencijos numeris yra pririšamas prie vartotojo el. pašto adreso, todėl su vienu el. pašto adresu galima gauti tik vieną licenciją. Licencijos numeris toliau naudojamas bendravimui su *cFilter* serveriu asmeniniams bei bendriems atnaujinimams gauti, klasifikatoriaus apmokymo duomenims pateikti ir pašalinti, dalintis bendrais *cFilter* nustatymais ir kitiems veiksmams atlikti, kuriems reikia identifikuoti įskiepio kopiją.

Taip pat planuojama[2] saugoti *cFilter* nuostatas nutolusiame serveryje ir susieti jas su licencijos numeriu. Tokiu būdu kelios skirtingos įskiepio kopijos, besidalinančios tuo pačiu licencijos numeriu, nesvarbu kur jos ir kokiu atstumu viena nuo kitos būtų įdiegtos, naudotųsi tomis pačiomis saugumo, filtravimo nuostatomis bei identiškai klasifikuotų internetinius puslapius. Taip galima būtų lengviau administruoti ne vieną įskiepio kopiją, pavyzdžiui, įdiegtą į daugybę įstaigoje esančių kompiuterių.

Klasifikavimas. *cFilter* internetinius puslapius klasifikuoja pasitelkdamas su atnaujinimais gautus klasifikavimo žodynus. Įskiepis pasižymi tokiomis savybėmis kaip gebėjimas klasifikuoti dar nežinomus puslapius bei blokuoti dalį (priedas nr. 13), o esant reikalui ir visą (priedas nr. 14) internetinį dokumentą. Taip pat galimybe klasifikuoti įvairiomis kalbomis pateiktus internetinius puslapius[2]. Papildomai palikta galimybė vartotojui pagal savo tolerancijos laipsnį kontroliuoti tris pagrindinius klasifikavimo metu naudojamus parametrus: *BP*, *BN* ir *PN* (priedas nr. 15). Taip pat egzistuoja galimybė pasirinkti (priedas nr. 16),

[1] *Beta* versijoje ši savybė nerealizuota
[2] *Beta* versijoje ši savybė pilnai nerealizuota

kokius klasifikavimo žodynus vartotojas pageidautų naudoti: *cFilter* siūlomus standartinius ar pagal vartotojo apmokymo duomenis sudarytus žodynus[2].

Apmokymo režimas. Vartotojui pageidaujant paliekama galimybė apmokyti klasifikatorių pagal savo poreikius ir sudaryti savas teigiamų ir neigiamų puslapių kategorijas. Tam tikslui *cFilter* įskiepyje realizuotas apmokymo režimas, kurį žinant administravimo slaptažodį galima laikinai įjungti (priedas nr. 17) iki sekančio *Firefox* naršyklės perkrovimo. Apmokymas atliekamas rankiniu būdu pažymint pageidaujamą internetinio puslapio bloką (priedas nr. 18) ir iš pateikto kategorijų sąrašo (priedas nr. 19) pasirinkus pageidaujamą kategoriją: teigiamą, neigiamą arba neutralią. Pasirinkto puslapio bloko turinys nusiunčiamas į nutolusį serverį ir saugomas tolimesniam apdorojimui.

Reguliarūs atnaujinimai. *cFilter* serveris reguliariai atlieka įvairioms kalboms pritaikytų *SVM* klasifikatorių apmokymo procesą ir paruošia atnaujinimus, kurie kiekvieno *Firefox* pirminio užsikrovimo metu užsikrovus *cFilter* įskiepiui yra atsiunčiami ir naudojami puslapių klasifikavimo tikslais. Atnaujinimo metu atsiunčiami vienas arba daugiau dažniausiai naudojamų kalbų klasifikavimo žodynų[3]. Kokios kalbos dažniausiai naudojamos yra sužinoma iš vartotojo paskutinio mėnesio naršymo statistikos[4]. Taip pat pasitaikius atvejui, kad užkrovus pageidaujamą puslapį pasirodytų, kad įskiepis dar nebuvo atsisiuntęs puslapyje vartojamos kalbos palaikymo, jis tą padaro automatiškai atskiru pageidavimu kreipiantis į nutolusį *cFilter* serverį[5].

Dvikalbė vartotojo sąsaja. *cFilter* įskiepis automatiškai prisitaiko prie naršyklėje nustatytos vartotojo sąsajos kalbos ir priklausomai nuo jos pasirenka kokią kalbą naudoti: lietuvių ar anglų (vartojama kaip standartinė kalba visoms vertimų neturinčioms lokalėms).

5.2. Įskiepio veikimo principai

Galima išskirti keturis pagrindinius *cFilter* įskiepio veikimo scenarijus: apmokymo duomenų surinkimas, klasifikatoriaus apmokymas, turinio filtro atnaujinimas bei internetinio turinio filtravimas.

[3] *Beta* versijoje naudojamas tik anglų kalbos žodynas
[4] Ši savybė *beta* versijoje išjungta
[5] *Beta* versijoje ši savybė nerealizuota

5.2.1. Apmokymų duomenų aibės sudarymas ir papildymas

Pirmas dalykas, su kuriuo susiduriama turinio klasifikavimo procese, tai apmokymo duomenų aibės sudarymas. Taip pat ir mūsų kuriamo įskiepio atveju neapseinama be šio etapo. Vienintelis atvejis, kai eilinis vartotojas gali jo išvengti, tai naudoti *cFilter* siūlomus standartinius atnaujinimus ir nebandyti pritaikyti klasifikavimo proceso prie savo poreikių. Tiek vienu, tiek kitu atveju duomenų aibės sudaromos pagal tą patį principą – naudojant *cFilter* įskiepio siūlomą apmokymo režimą.

Apmokymo metu įskiepio vartotojas atsidaręs bet kurį internetinį puslapį gali pasirinkti jam patinkantį ar nepatinkantį dokumento bloką ir iš kategorijų sąrašo pasirinkti pageidaujamą. Yra galimi trys kategorijų variantai: teigiamas, neutralus ir neigiamas. Pirmą kartą tam pačiam turinio blokui pasirinkus vieną iš kategorijų *cFilter* įskiepis pagal 4 skyriuje aprašytus metodus sudaro transformuotą dokumento medį, apdoroja bloke saugomą *HTML* turinį ir sudaro raktinių ir visų žodžių dažnumų vektorius hV ir cV (priedas nr. 4). Toliau iš gauto vektoriaus cV išgaunama iki 10 atsitiktinių žodžių, kurie nusiunčiami *Google* vertimų servisui dokumento bloko turinio kalbai nustatyti. Tuomet turint duomenų vektorių, turinio kategoriją bei žinant turinyje vartojama kalbą siunčiama užklausa *cFilter* servisui, kuris gautus apmokymo duomenis pririša prie įskiepio licencijos numerio ir išsaugo duomenų bazėje. Galiausiai *cFilter* servisas įskiepiui grąžina apmokymo duomenų įrašo *id* numerį bei papildomai sugeneruotą unikalų 40 simbolių žetoną (*token*). *cFilter* įskiepis priskiria gautus identifikacinius numerius prie konkretaus turinio bloko, tuo tikslu, jei vartotojas persigalvotų ir pageidautų pakeisti turinio bloko kategoriją į kitą. Identifikaciniai numeriai *id* ir *token* yra aktyvūs tik vieną valandą nuo jų sūkurimo, taip apsisaugoma nuo netyčinio ar nuo piktybinio kategorijos pakeitimo iš šalies. Taip pat papildomas numeris *token* yra naudojamas saugumo sumetimais, kad niekas net ir atspėjęs *id* numerį praktiškai niekada negalėtų atspėti dar ir *token* kontrolinio numerio. Tokiu būdu duomenys apsaugomi nuo piktybinio pakeitimo iš išorės.

Vartotojas anksčiau nustatęs dokumento turinio bloko kategoriją ir panorėjes ją pakeisti, vėl pasirenka tą patį turinio bloką ir pažymi iš sąrašo kitą kategoriją. Šį kartą duomenų vektoriai bei kalba nėra analizuojami, o pasitelkiami tik identifikaciniai numeriai. Tuomet *cFilter* įskiepis jau žinodamas bloko *id* ir *token* numerius siunčia užklausą *cFilter* servisui ir nurodo pakeisti duomenų vektorių kategoriją. Kaip buvo minėta, tam tikslui vartotojui yra skiriama tik viena valanda nuo pirmojo konkretaus bloko kategorijos pasirinkimo.

5.2.2. Klasifikatoriaus apmokymas

Sekantis *cFilter* sistemos veikimo etapas – tai klasifikatoriaus apmokymas. Apmokymo metu yra sudaromas atnaujinimo paketas, skirtas *cFilter* įskiepiams ir yra saugomas duomenų bazėje (priedas nr. 5). Klasifikatorius apmokomas su kiekvienos licencijos-kalbos poros duomenų aibe, t.y. sudaromi individualūs atnaujinimai kiekvienam vartotojui ar jų grupei pagal jų pageidaujamus apmokymo duomenis bei atskirai apmokymo duomenų kalbai.

5.2.3. Turinio filtro atnaujinimas

Turint paruoštus klasifikavimui reikalingus atnaujinimus, belieka jais pasinaudoti. Kiekvieno naršyklės užsikrovimo metu užkrovus *cFilter* įskiepį įskiepis kreipiasi į *cFilter* serverį su užklausa atsiųsti su licencijos numeriu susietus vartotojo dažniausiai naudojamų kalbų žodynų atnaujinimus. Taip pat žodynai gali būti atsiunčiami ir pagal poreikį *cFilter* įskiepio darbo metu pritrūkus reikiamos kalbos žodyno (priedas nr. 6).

5.2.4. Turinio filtravimas

Kiekviena užregistruota ir įjungta *cFilter* įskiepio kopija jau yra paruošta kategorizuoti internetinius dokumentus. Kiekvieno internetinio puslapio užklausimo metu dar nepradėjus siųsti internetinio dokumento naršyklė siunčia puslapio užklausos signalą *cFilter* įskiepiui, kuris gavęs šį signalą liepia naršyklei paslėpti visa turinį konkrečioje kortelėje, kurioje planuojama užkrauti dokumentą (priedas nr. 7). Tol kol nebus apdorotas internetinis dokumentas, nebus matoma nepageidaujama informacija.

Toliau yra nustatoma, koks dokumentas buvo užklaustas. Jei tai vienas iš *cFilter* puslapių, tuomet puslapis yra iškart parodomas, o tolimesnis filtravimo procesas nevykdomas. Jeigu tai buvo vienas iš kritinių sisteminių puslapių, prie kurių priėjimas saugumo sumetimais privalo būti apribotas, įskiepis praneša apie uždraustą priėjimą ir pasiūlo galimybę pasiekti dokumentą pateikus teisingą administratoriaus slaptažodį. Jeigu buvo užklaustas bet kuris kitas sisteminis puslapis, kurį galima matyti bet kuriam vartotojui, filtravimo procesas nevykdomas, o puslapis pateikiamas vartotojui. Visais kitais atvejais, jei *cFilter* įskiepis yra įjungtas, yra patikrinama, ar užklaustas puslapis patenka į patikimų puslapių sąrašą. Jei puslapis patikimas jį leidžiama pamatyti vartotojui. Jei jo patikimų puslapių sąraše nėra, tai patikrinama, ar puslapio adreso nėra juodajame saraše. Jei taip, tai puslapis yra blokuojamas bei pasiūloma įvedus administratoriaus slaptažodį laikinai prieiti prie puslapio. Laikinas priėjimas saugumo sumetimais įmanomas tik iki sekančio naršyklės perkrovimo.

Jei puslapis netenkino jokių išankstinių sąlygų, tuomet yra palaukiama kol puslapis bus pilnai užkrautas. Turint tvarkingai užkrautą dokumentą, įskiepis kreipiasi į *Google* vertimų servisą su pageidavimu nustatyti puslapio kalbą. Žinant kokia kalba pateiktas dokumentas, belieka parinkti reikiamą klasifikavimo žodyną (esant reikalui atsisiųsti) bei apdoroti dokumento turinį. Galiausiai apdorotas turinys priklausomai nuo rezultato yra blokuojamas arba pateikiamas vartotojui.

6. Eksperimentai ir rezultatai

Siekiant įgyvendinti šiame darbe aprašytą algoritmą buvo realizuota dviejų dalių paskirstyta sistema. *PHP* programavimo kalba buvo sukurta centralizuota sistemos dalis, veikianti nutolusiame serveryje ir atsakinga už apmokymo duomenų kaupimą, jų apdorojimą, klasifikatoriaus apmokymą bei atnaujinimų paruošimą. Taip pat atsakinga už tokius su vartotojais susijusius uždavinius kaip vartotojų registracija, licencijavimas ir t.t. Sistemoje naudojamas statistinio mokymosi algoritmas *SVMlin* [44], kuris siekiant portabilumo buvo perprogramuotas *PHP* programavimo kalba. Kita realizuotos sistemos dalis – tai vartotojų *Firefox* naršyklėse veikiantys *Javascript* programavimo kalba realizuoti įskiepiai, sąveikaujantys su centrine sistemos dalimi ir naudojami internetiniams dokumentams filtruoti bei apmokymo duomenų aibėms sudaryti.

Siekiant pademonstruoti sistemos veikimo principus įskiepio pagalba rankiniu būdu iš anglakalbių įvairaus tematinio pobūdžio naujienų ir nepadoraus turinio tinklapių buvo išgautas nedidelis kiekis, t.y. 155, įvairaus dydžio iškarpų (pomedžių), iš kurių 85 priklauso teigiamai kategorijai, o 70 neigiamai. Iš dokumentų buvo pašalinti skaičiai bei žodžiai, sudaryti iš mažiau nei 3 simbolių. Iš 15067 rastų unikalių žodžių pasitelkiant *CDW* požymių atrankos metodą buvo atrinktį 14722 skirtingi požymiai. Iš apmokymo duomenų pasitelkiant tik atrinktus požymius buvo sudaryti 155 požymių vektoriai, apmokytas *SVM* klasifikatorius ir paruoštas atnaujinimo paketas, sudarytas iš 14722 unikalių termų, jų apskaičiuotų svorinių įverčių bei nuo klasės priklausomų *CDW* įverčių. Pasitelkiant apmokytu klasifikatoriumi apmokymo duomenų aibė dėl savo nedidelės imties buvo suklasifikuota 100% tikslumu.

Siekiant sudarytą klasifikatorių išbandyti su apmokymo duomenyse nepanaudotais internetiniais puslapiais buvo pasinaudota dvejomis nedidelėmis internetinių portalų nuorodų duomenų bazėmis: *Alex* [1] pateikiamas 500 populiariausių pasaulyje tinklapių sąrašas bei *TBLOP* siūlomas išimtinai nepadorių tinklalapių archyvas [45]. Iš daugiau nei 600 puslapių buvo atrinkti tik tie puslapiai, kuriuose vartojama anglų kalba, taip pat buvo pašalinti pasikartojantys puslapiai. Po šio apdorojimo proceso iš viso eksperimentams atlikti buvo panaudoti 394 internetiniai puslapiai: 275 teigiamo pobūdžio ir 119 neigiamo. Atliekant eksperimentus buvo pasitelktas *cFilter* įskiepis su standartiniais parametrais $BP = 15\%$, $BN = 50\%$ ir $PN = 66\%$.

6 lentelėje matome, kad absoliučiai visi teigiamo pobūdžio internetiniai puslapiai buvo suklasifikuoti teisingai. Su nepadoraus turinio puslapiais sekėsi kiek prasčiau. Iš 119 nepadoraus turinio puslapių 9 (7.56%) buvo klaidingai priskirti teigiamai kategorijai. Nepaisant

		Tikroji kategorija	
		Teigiama	Neigiama
Apskaičiuota kategorija	Teigiama	275	0
	Neigiama	9	110
	Precision	*Recall*	$F_{0.5}$
	96.83%	100%	97.45%

6 lentelė: *cFilter* įskiepio klasifikavimo rezultatai

to, kad nors ir apmokymo imties aibė buvo daugiau nei dukart mažesnė nei testuojamų dokumentų aibė, tai nesutrukdė klasifikatoriui sėkmingai klasifikuoti jam visiškai nežinomus internetinius puslapius ir pasiekti puikų *Recall* (100%) bei tikrai labai gerą *Precision* (96.83%) įverčius bei galutinį kompleksinį įvertį $F_{0.5}$ (97.45%).

Prastesnio neigiamo turinio klasifikavimo pasekmė dažniausiai buvo per mažas tekstinio turinio ir labai didelis neanalizuojamo vaizdinio turinio kiekis, dėl ko klasifikatoriui paprasčiausiai pritrūko būtinos informacijos, kad tinkamai sukategorizuoti tokius tinklalapius. Iš to galime daryti išvadą, kad norint tinkamai suklasifikuoti daugialypės terpės dokumentus nepakanka naudotis tik vienu iš daugelio galimų elementų tipų, galinčių pasitaikyti dokumente, ir kuo labiau išnaudoti tokius labai plačiai vartojamus ir daug informacijos teikiančius elementus kaip paveiksliukai. Taip pat esant būtinybei, kai klasifikatorius nebepadeda, verta pasinaudoti ir tokiu puslapių filtravimo metodu kaip filtravimas pagal nuorodas.

Išvados

Šiame darbe pasiūlėme bei praktiškai realizavome algoritmą, pagrįstą daugialypės terpės failų klasifikavimu ir skirtą klasifikuoti *HTML* formato dokumentus į dvi kategorijas – teigiamą ir neigiamą. Realizuotas algoritmas taip pat geba aptikti ir neigiamo pobūdžio dokumento blokus neišbrokuojant viso dokumento turinio. Algoritmas pasižymi tuo, kad nereikalauja nuolat rankiniu būdu atnaujinti blokuotinų puslapių duomenų bazės, todėl sugeba klasifikuoti ir dar jam nežinomus dokumentus. Taip pat pasiekta, kad algoritmas būtų nesudėtingas, efektyvus ir paprastas, kad būtų galimybė jį realizuoti tokio riboto funkcionalumo aplinkoje kaip *Firefox* naršyklės įskiepis. Klasifikavimo taisyklei sudaryti buvo pasitelktas tiesinis atraminių vektorių algoritmas, realizuotas *PHP* programavimo kalba ir veikiantis atskirai nuo įskiepio. Pagal pageidavimą pasinaudojus įskiepio galimybėmis galima sukurti ir savo klasifikavimo taisykles. Tokiu būdu sudarant galimybę klasifikuoti puslapius ne tik į padoraus ir nepadoraus turinio kaip buvo pasiūlyta šiame darbe, bet ir į kitokias pageidaujamas kategorijas. Vienintelis apribojimas – tai galimybė dokumentus suskirstyti tik į dvi skirtingas kategorijas.

Pasiūlytas klasifikavimo algoritmas puikiai klasifikavo teigiamo turinio puslapius, bet pasiekė silpnesnį, tačiau pakankamai gerą rezultatą klasifikuodamas neigiamo turinio portalus. Esminė to priežastis ta, kad klasifikavimui naudojamas tik puslapio tekstinis turinys ir neatsižvelgiama į tokį vizualinį turinį kaip nuotraukos ar vaizdiniai failai. Todėl ateityje būtina papildyti algoritmo funkcionalumą galimybėmis klasifikuoti *HTML* dokumentuose pateikiamas nuotraukas, vaizdinius failus bei kitus grafinius elementus pasitelkiant ne tik tekstinį, bet ir grafinį klasifikatorių.

Literatūros sąrašas

[1] Alexa, „Alexa Top 500 Global Sites“, http://www.alexa.com/topsites

[2] R. Bekkerman and J. Allan. Using bigrams in text categorization. CIIR (the Center for Intelligent Information Retrieval, University of Massachusetts), Technical Report IR-408 (2004).

[3] B. E. Boser, I. M.Guyon, V. N. Vapnik. A training algorithm for optimal margin classifiers. In: Proc. of the 5th Annual Workshop on Computational Learning Theory, pp. 144–152 (1992).

[4] M. Caropreso, S. Matwin, and F. Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. Text Databases and Document Management: Theory and Practice. Hershey, US: Idea Group Publishing (2001).

[5] C. Chekuri, M. Goldwasser, P. Raghavan, and E. Upfal. Web search using automated classification. In Proceedings of the Sixth (1997).

[6] S. Chakrabarti. Data mining for hypertext: a tutorial survey. SIGKDD Explorations Newsletter 1 (2), pp. 1–11 (2000).

[7] L.-S. Chen and C.-W. Chang. A New Term Weighting Method by Introducing Class Information for Sentiment Classification of Textual Data. Proceedings of the International MultiConference of Engineers and Computer Scientists Vol 1 (2011).

[8] S. Chakrabarti, B. Dom, P. Indyk. Enhanced hypertext categorization using hyperlinks. Proceedings of SIGMOD-98, ACM International Conference on Management of Data, ACM Press, New York, US, pp. 307-318 (1998).

[9] T.Cover, P.Hart. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), pp. 21–27 (1967).

[10] C. Cortes, V. Vapnik. Support-vector networks. Machine Learning, 20(3), pp. 273–297 (1995).

[11] L. Denoyer, P. Gallinari, J.-N. Vittaut, S. Brunesseaux and S. Brunesseaux. Structured Multimedia Document Classification (2003).

[12] P. A. Devijver and J. Kittler. Pattern Recognition Theory and Applications. Berlin: Springer-Verlag (1987).

[13] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management, pp. 148-155 (1998).

[14] J. H. Friedman. Flexible metric nearest neighbor classification. Technical report, Dept. of Statistics, Stanford University (1994).

[15] N. Fuhr. A probabilistic model of dictionary-based automatic indexing. Proceedings of RIAO-85, 1st International Conference, Recherche d'Information Assistee par Ordinateur, pp. 207-216 (1985).

[16] J. Furnkranz, T. Mitchell, and E. Riloff. A case study in using linguistic phrases for text categorization on the WWW. Proceedings of the AAAI/ICML Workshop on Learning for Text Categorization, pp. 5-12 (1998).

[17] J. Furnkranz. Exploiting structural information for text classification on the WWW. Proceedings of the 3rd Symposium on Intelligent Data Analysis (IDA-99), SpringerVerlag, Amsterdam, Netherlands (1999).

[18] J. Furnkranz. Web mining. In: O. Maimon and L. Rokach (Eds.), The Data Mining and Knowledge Discovery Handbook, pp. 899–920 (2005).

[19] Wikipedia, „F1 score“, `http://en.wikipedia.org/wiki/F1_score`.

[20] T. Hofmann. Introduction to Machine Learning. Draft Version 1.1.5, November 10 (2003).

[21] T. Joachims. Text categorization with Support Vector Machines: Learning with many relevant features. Proceedings of ECML-98, 10th European Conference on Machine Learning, pp. 137-142 (1998).

[22] T. Joachims, N. Cristianini, J. Shawe-Taylor. Composite kernels for hypertext categorisation. Proceedings of ICML-01, 18th International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, US, pp. 250–257 (2001).

[23] S. Kiritchenko, S. Matwin, and F. Famili. Functional annotation of genes using hierarchical text categorization. Proceedings of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology (2005).

[24] S. Kiritchenko, S.Matwin, and F. Famili. Hierarchical text categorization as a tool of associating genes with Gene Ontology codes. The 2nd European Workshop on Data Mining and Text Mining for Bioinformatics held in conjunction with 15th European Conference on Machine Learning (ECML), pp. 26-30 (2004).

[25] J. Kivinen, M. Warmuth and P. Auer. The perceptron algorithm vs. winnow: Linear vs. logarithmic mistake b ounds when few input variables are relevant. In: Conference on Computational Learning Theory (1995).

[26] R. Kosala and H. Blockeel. Web mining research: A survey. SIGKDD Explorations Newsletter 2 (1), pp. 1–15 (2000).

[27] D. Lewis and W. Croft. Term clustering of syntactic phrases. Proceedings of the 13th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR-90), pp. 385-404 (1990).

[28] D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. Proceedings of the 15th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR-92), pp. 37-50 (1992).

[29] C. Liao, S. Alpha, and P. Dixon. Feature preparation in text categorization. ADM03 workshop (2003).

[30] Y. H. Li, A. K. Jain. Classification of text documents. The Computer Journal, 41(8), pp. 537–546 (1998).

[31] D. Mladenic and M. Grobelnik. Word sequences as features in text learning. Proceedings of the 17th Electrotechnical and Computer Science Conference (ERK-98), pp. 145-148 (1998).

[32] H. T. Ng, W. B.Goh, K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In: Proc. of the 20th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 67–73 (1997).

[33] J. M. Pierre. Practical Issues for Automated Categorization of Web Sites (2000).

[34] Wikipedia, „Part-of-speech tagging“, `http://en.wikipedia.org/wiki/Part-of-speech_tagging`.

[35] X. Qi and B. D. Davison. Web Page Classification: Features and Algorithms. Technical Report LU-CSE-07-010, Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, 18015 (2007).

[36] S. Ray and M. Craven. Representing sentence structure in hidden Markov models for information extraction. Proceedings of the International Joint Conference on Artificial Intelligence (2001).

[37] D. Riboni. Feature Selection for Web Page Classification. In: EURASIA-ICT 2002 Proceedings of the Workshop, pp. 473-478 (2002).

[38] M. E. Ruiz, P. Srinivasan. Hierarchical neural networks for text categorization. In: Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 281–282 (1999).

[39] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Information Processing and Management. 24(5), pp. 513-523 (1988).

[40] G. Salton. Automatic text processing: The transformation, analysis, and retrieval of information by computer. Reading, MA: Addison-Wesley (1989).

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

[42] M. Simeon and R. Hilderman. Categorical Proportional Difference: A Feature Selection Method for Text Categorization. The Australasian Data Mining Conference (Aus DM), pp. 201-208 (2008).

[43] A. Sun, E. Lim, and W. Ng. Performance measurement framework for hierarchical text classification. Journal of the American Society for Information Science and Technology (JASIST), 54(11), pp. 1014-1028 (2003).

[44] V. Sindhwani, „SVMlin: Fast Linear SVM Solvers for Supervised and Semi-supervised Learning“, `http://vikas.sindhwani.org/svmlin.html`.

[45] TBLOP, „The Big List of Porn“, `http://www.tblop.com`

[46] V. N. Vapnik. The Nature of Statistical Learning Theory. New York: Springer-Verlag. ISBN 0-387-94559-8 (1995).

[47] J. Vert, K. Tsuda and B. Schölkopf. A primer on kernel methods. Kernel Methods in Computational Biology, pp. 35-70 (2004).

[48] W. A. Arentz, B. Olstad. Classifying offensive sites based on image contents. Computer Vision and Image Understanding, Vol 94, pp. 293-310 (2004).

[49] S. M. Weiss, C. Apte, F. J. Damerau, D. E. Johnson, F. J. Oles, T. Goetz, T. Hampp. Maximizing text-mining performance. IEEE Intelligent Systems, 14(4): pp. 63–69 (1999).

[50] E. D. Wiener, J. O. Pedersen, A. S. Weigend. A neural network approach to topic spotting. In: Proc. of the 4th Annual Symposium on Document Analysis and Information Retrieval, pp. 317–332 (1995).

[51] International World Wide Web Conference, Santa Clara, CA. Poster POS725. The World Wide Web Consortium (W3C), „HTML 4.01 Specification“, `http://www.w3.org/TR/html4/`.

[52] Y. Yang and J. Pederson. A comparative study on feature selection in text categorization. Proceedings of the 14th International Conference on Machine Learning (ICML), pp. 412-420 (1997).

[53] L. Zhang, J. Zhu and T. Yao. An evaluation of statistical spam filtering techniques. ACM Transactions on Asian Language Information Processing (TALIP) 3(4), pp. 243–269 (2004).

Priedas nr. 1

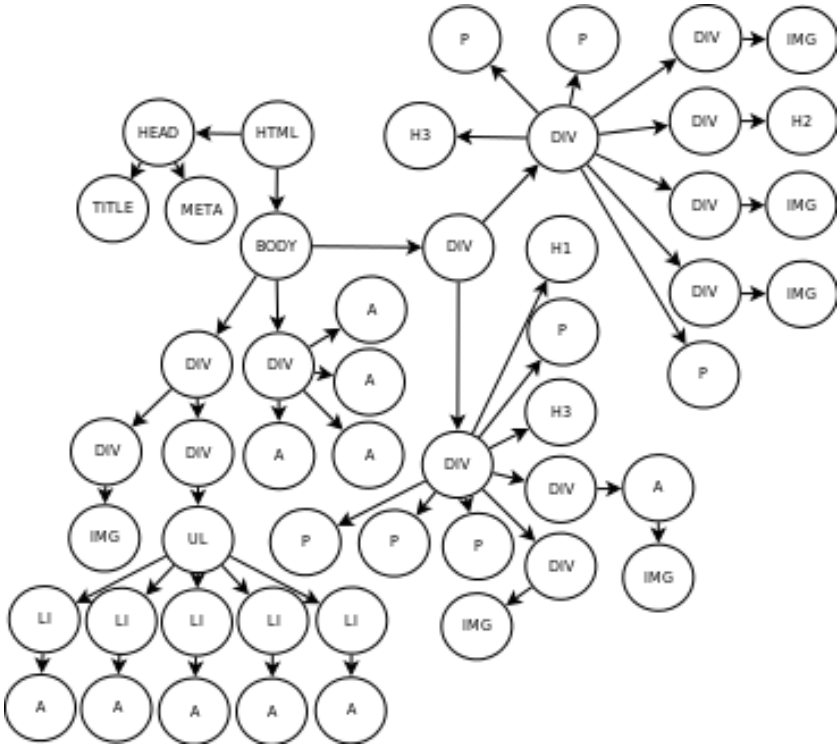
χ^2 ir CDW požymių atrankos metodų palyginamoji lentelė

Termas	χ^2	CDW	Termas	CDW	χ^2
amateur	#1	#1-13062	news	#13063	#7
anal	#2	#1-13062	blonde	#13064	#3
blonde	#3	#13064	brunette	#13065	#4
brunette	#13065	#	ass	#13066	#8
pussy	#5	#1-13062	easter	#13067	#33
blowjob	#6	#1-13062	said	#13068	#46
news	#7	#13063	fucking	#13069	#18
ass	#8	#13066	been	#13070	#64
says	#9	#1-13062	horny	#13071	#31
fuck	#10	#1-13062	end	#13072	#79
cock	#11	#1-13062	set	#13073	#101
fucked	#12	#1-13062	plan	#13074	#100
porn	#13	#13876	police	#13075	#108
tits	#14	#1-13062	teens	#13076	#52
sex	#15	#14549	friday	#13077	#107
that	#16	#14644	deal	#13078	#116
mature	#17	#13878	cute	#13079	#55
euro	#15051	#14651	our	#14706	#15052
our	#15052	#14706	you	#14707	#1575
pain	#15053	#11457	her	#14708	#251
aged	#15054	#14453	get	#14709	#15028
master	#15055	#14454	old	#14710	#15059
diamond	#15056	#14451	white	#14711	#8085
fill	#15057	#14458	com	#14712	#3907
eat	#15058	#14452	video	#14713	#1870
old	#15059	#14710	your	#14714	#5819
pounds	#15060	#14507	home	#14715	#14669
spanish	#15061	#14511	best	#14716	#14651
download	#15062	#14513	big	#14717	#221
date	#15063	#14510	all	#14718	#1536
reason	#15064	#14515	for	#14719	#202
going	#15065	#14624	the	#14720	#60
front	#15066	#14623	with	#14721	#456
action	#15067	#14622	and	#14722	#1570

7 lentelė: χ^2 ir CDW požymių atrankos algoritmų rezultatų palyginimas, kur # nurodo termo reitingo vietą

Priedas nr. 2

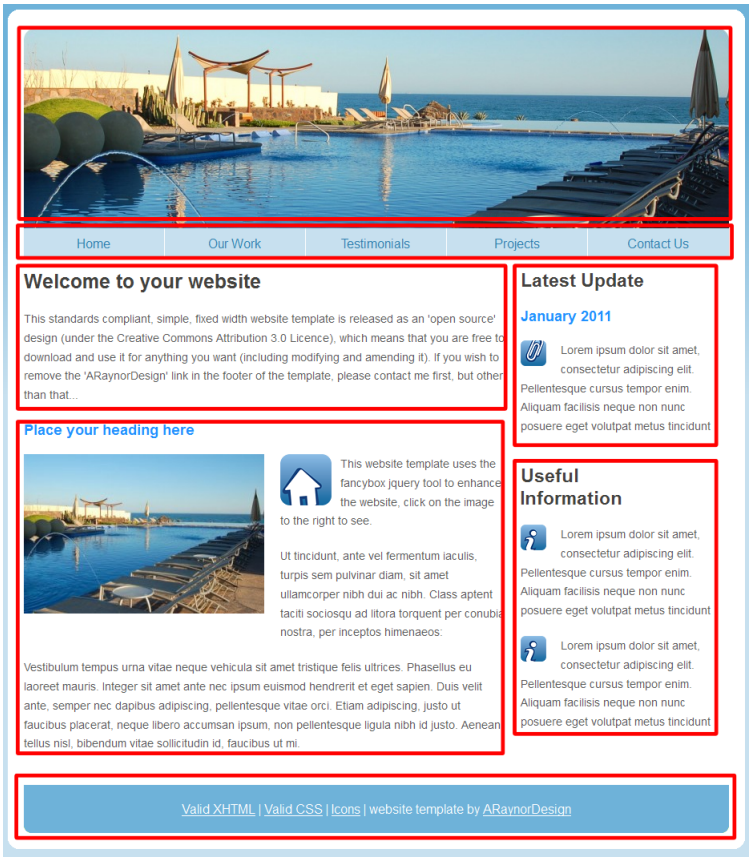
HTML dokumento žymų medis



20 pav.: $HTML$ dokumento medžio pavidalo struktūra

Priedas nr. 3

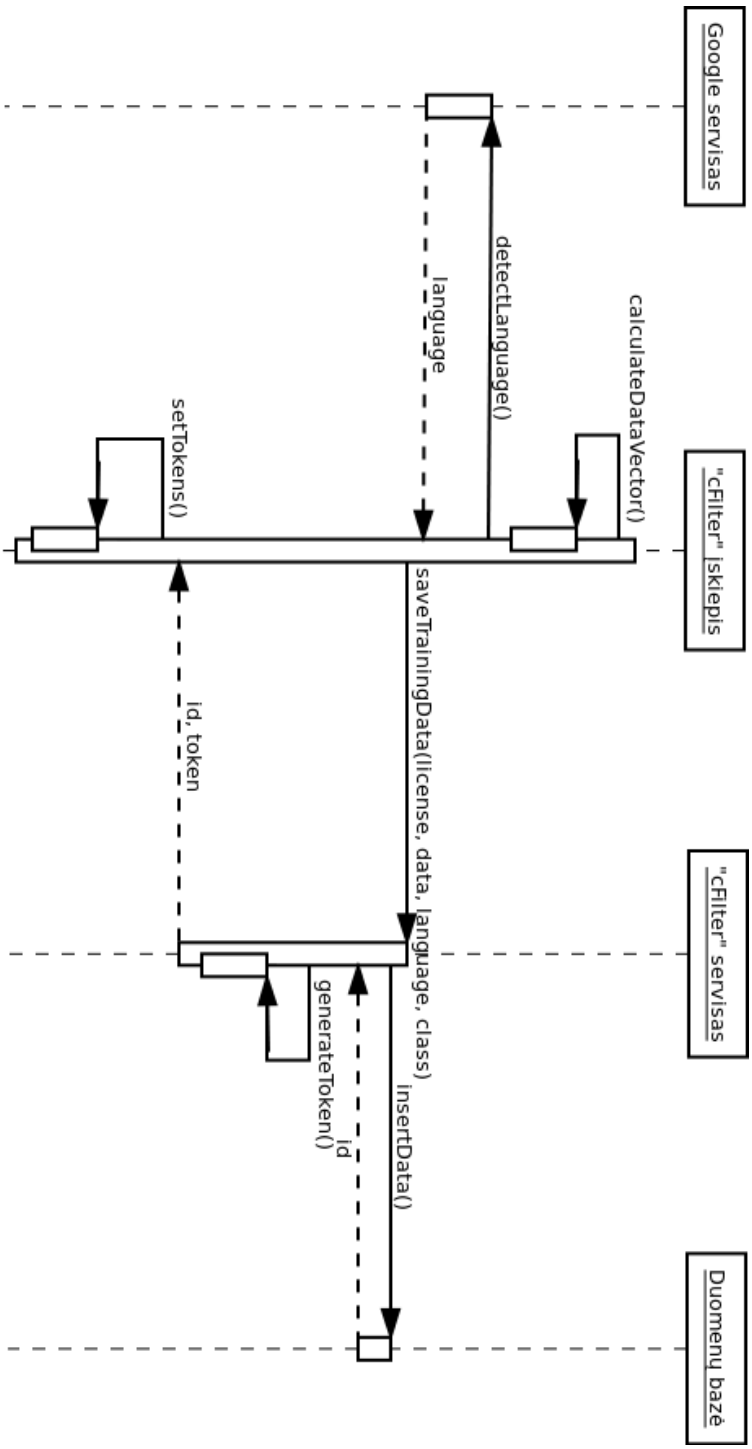
Tinklalapio semantiniai blokai



21 pav.: $HTML$ dokumento semantiniai blokai

Priedas nr. 4

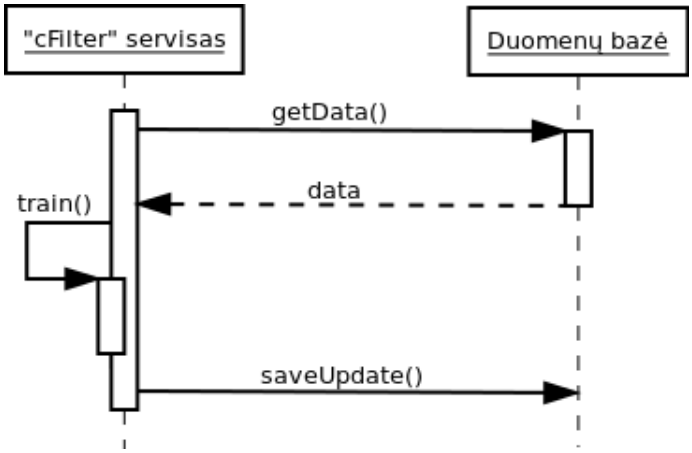
cFilter turinio filtro apmokymo schema



22 pav.: *cFilter* filtro apmokymo schema

Priedas nr. 5

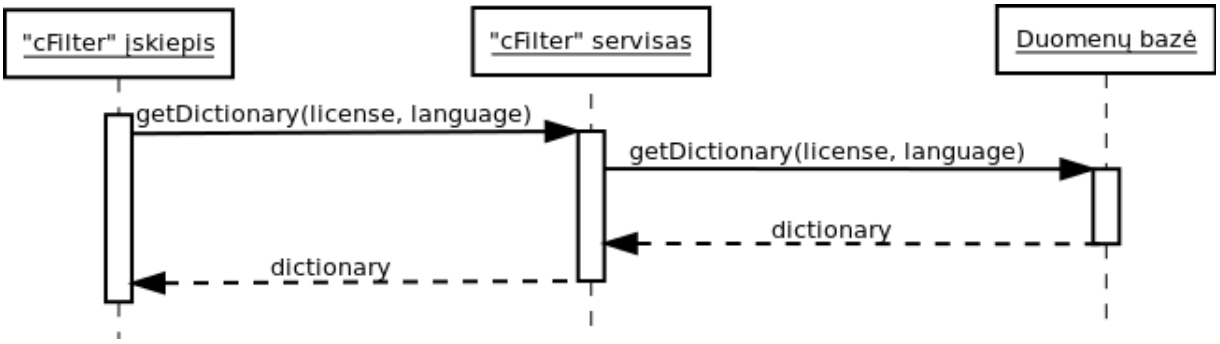
Atraminių vektorių klasifikatoriaus apmokymo schema



23 pav.: *SVM* klasifikatoriaus apmokymo schema

Priedas nr. 6

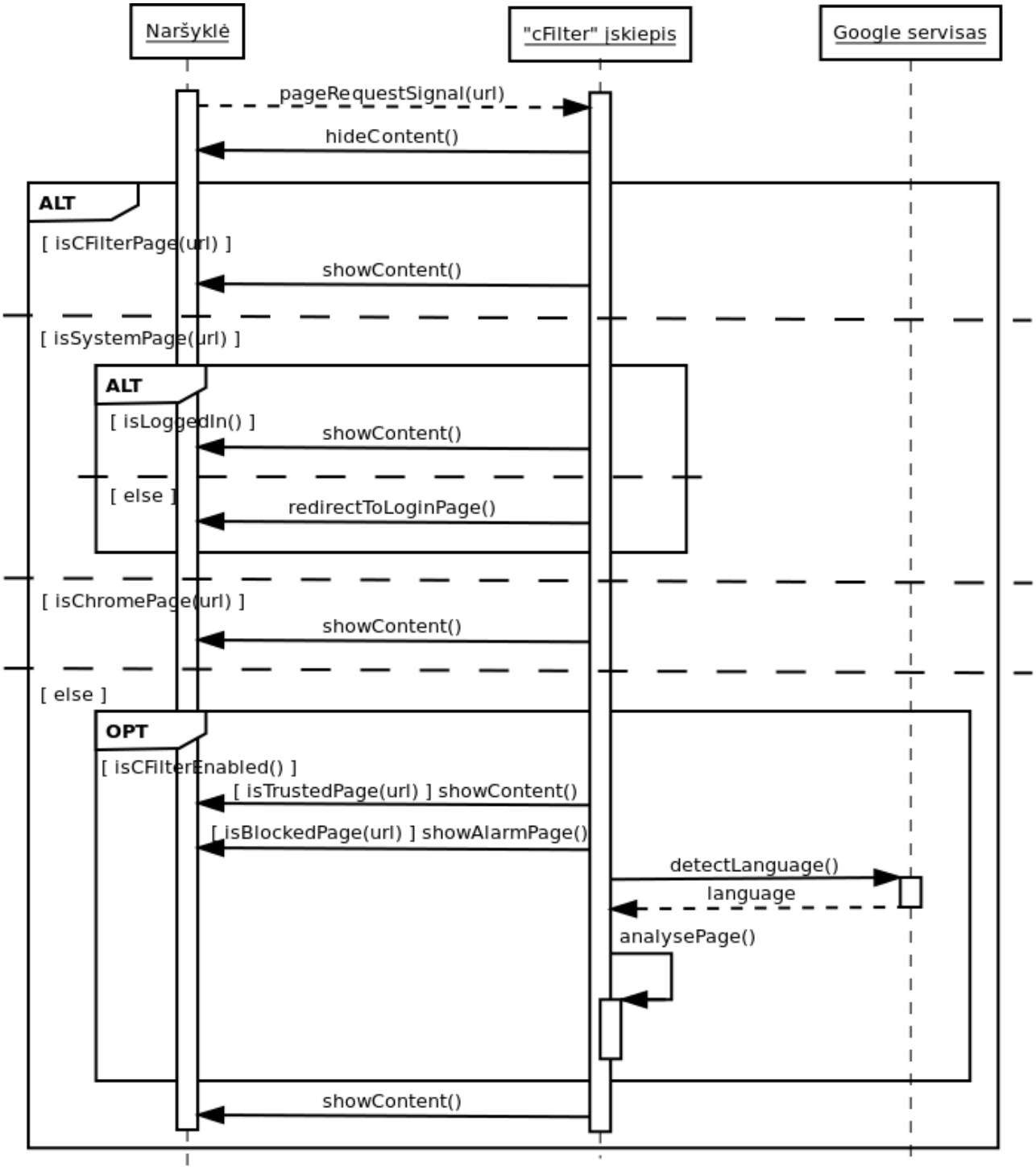
cFilter įskiepio duomenų bazės atnaujinimo schema



24 pav.: *cFilter* turinio filtro atnaujinimo schema

Priedas nr. 7

cFilter filtravimo algoritmo schema



25 pav.: *cFilter* filtravmo schema

Priedas nr. 8

cFilter saugumo nuostatų forma

26 pav.: *cFilter* įskiepio saugumo nustatymai

Priedas nr. 9

Blokuojamas priėjimas prie *cFilter* nuostatų puslapio

27 pav.: Slaptažodžiu apsaugotas priėjimas prie *cFilter* parametrų

Priedas nr. 10

Blokuojamas priėjimas prie *Firefox* nuostatų lango

28 pav.: Slaptažodžiu apsaugotas priėjimas prie *Firefox* nustatymų

Priedas nr. 11

Blokuojamas priėjimas prie *Firefox* sisteminių puslapių

29 pav.: Slaptažodžiu apsaugotas priėjimas prie sisteminių *cFilter* puslapių

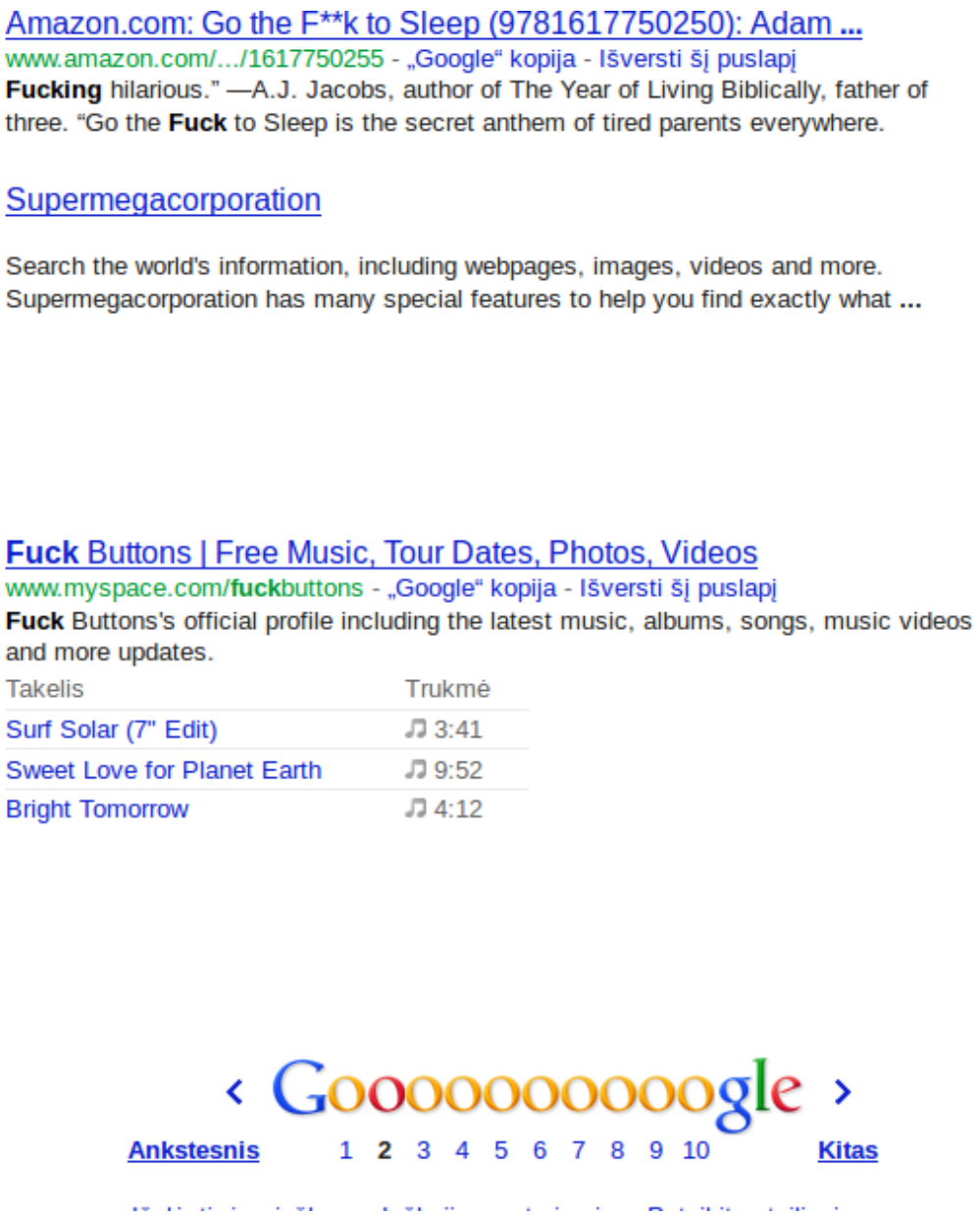
Priedas nr. 12

cFilter įskiepio kopijos registracijos forma

30 pav.: *cFilter* įskiepio registracijos forma

Priedas nr. 13

Blokuojamos įtartinos puslapio dalys



31 pav.: Dalinis puslapio filtravimas, pašalinti įtartini turinio blokai

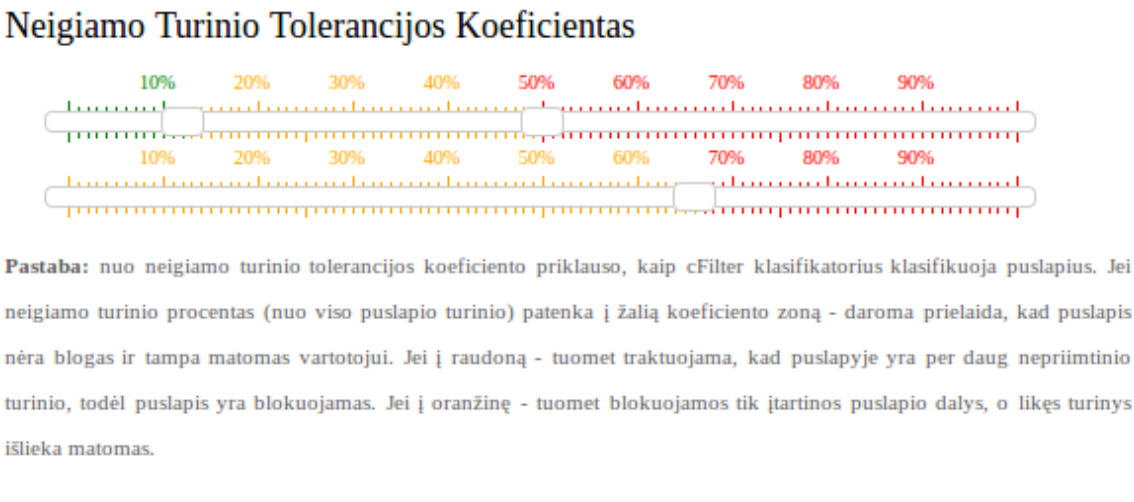
Priedas nr. 14

Blokuojamas visas įtartinas puslapis

32 pav.: Įspėjamasis pranešimas blokuojant visą puslapio turinį

Priedas nr. 15

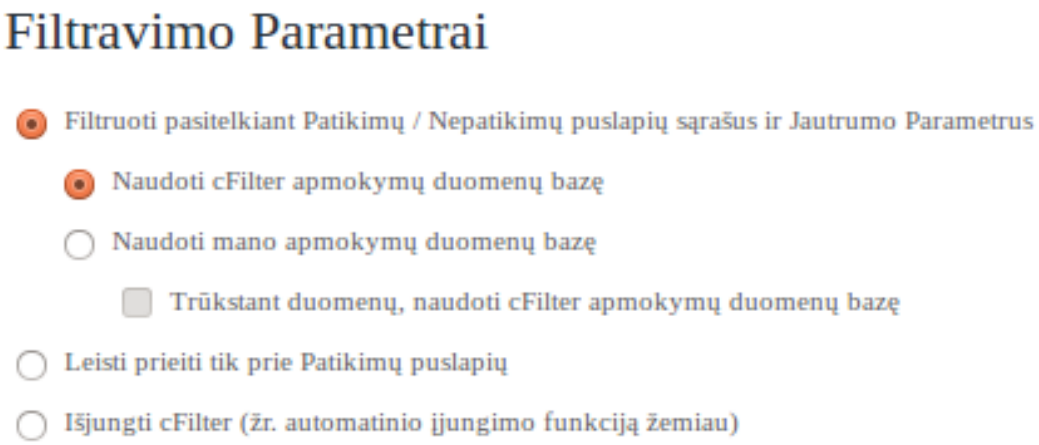
Tolerancijos koeficientų pasirinkimo forma



33 pav.: *cFilter* filtravimo parametrų BP, BN ir PN nustatymų forma

Priedas nr. 16

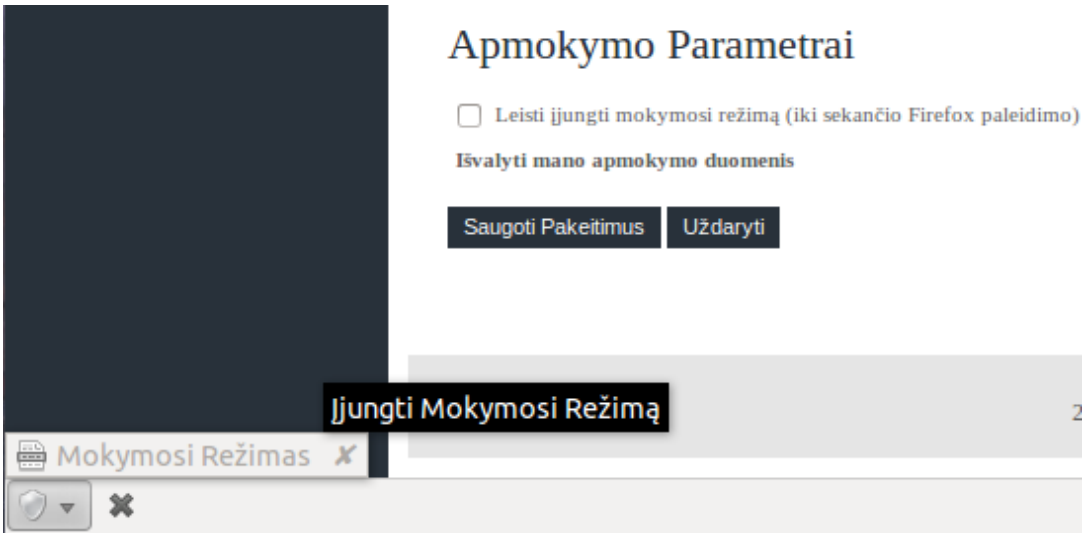
Filtravimo parametrų forma



34 pav.: *cFilter* filtravimo parametrai

Priedas nr. 17

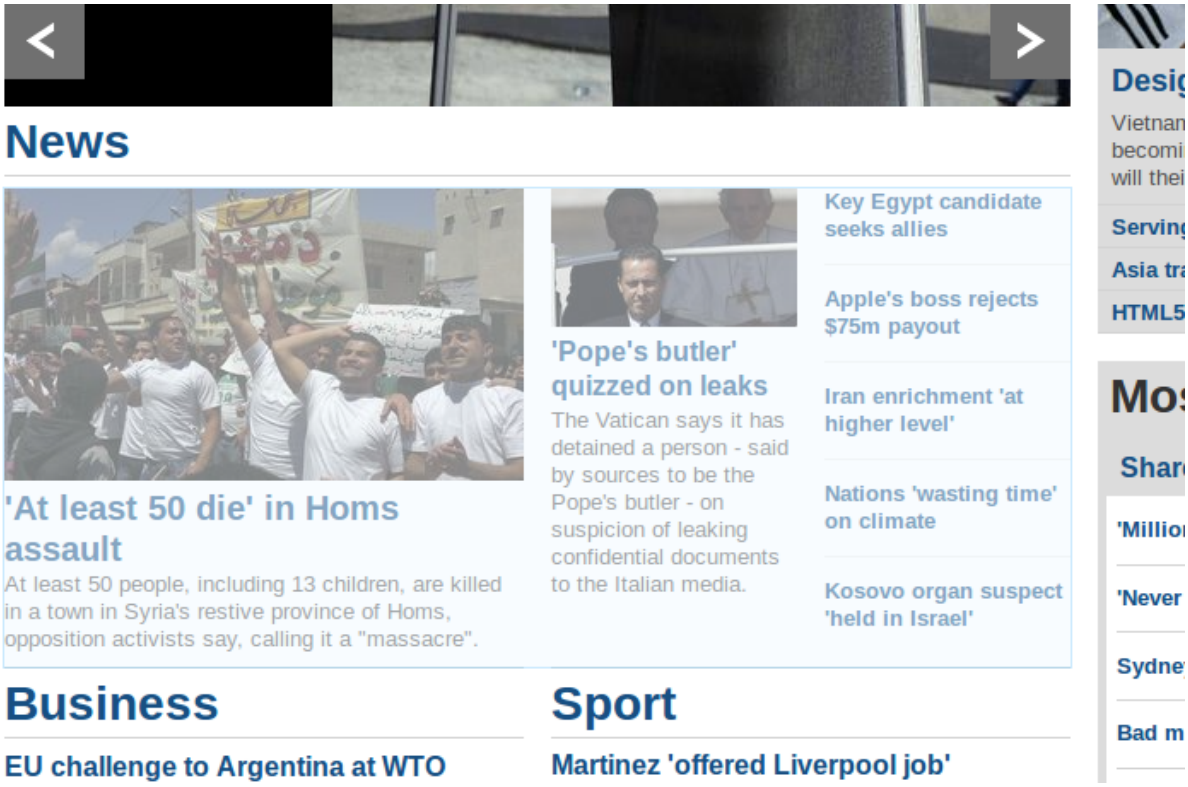
Apmokymo parametrų forma, mokymosi režimo įjungimo mygtukas



35 pav.: *cFilter* apmokymo režimo įjungimas

Priedas nr. 18

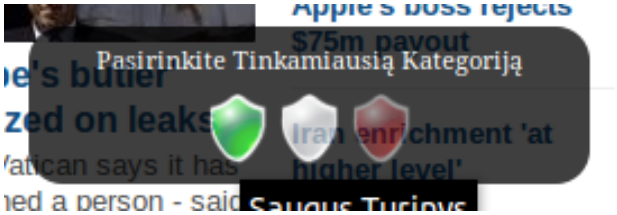
Puslapio bloko pasirinkimas apmokymo režime



36 pav.: Puslapio bloko pasirinkimas esant apmokymo režime

Priedas nr. 19

Puslapio bloko kategorijos pasirinkimas



37 pav.: Pasirinkto puslapio bloko kategorijos pasirinkimas