Implementacija softvera za filtriranje neželjenih poruka upotrebom klasifikacionih algoritama

Stefan Tubić, Miloš Cvetanović, Zaharije Radivojević, Saša Stojanović

***Apstrakt* – Algoritmi mašinskog učenja i pretrage podataka često se primenjuju u rešavanju problema u različitim oblastima. U ovom radu prikazano je nekoliko algoritama za pretprocesiranje i klasifikaciju teksta u mejl porukama sa ciljem izbora najboljeg modela i njegove primene unutar postojećeg *Postfix* mejl servera. U radu su obrazloženi algoritmi koji su korišćeni, pojašnjena njihova implementacija, prikazan postupak uporedne analize, koji je rezultovao najboljim modelom za klasfikaciju i prikazana pozicija modela kao komponente unutar mejl servera.**

***Ključne reči* – mašinsko učenje; pretraživanje podataka; klasifikacija; email; filtriranje neželjenih poruka.**

I. UVOD

Algoritmi mašinskog učenja su danas jedna od najpopularnijih tema u oblasti kompjuterske nauke. Jedna od definicija mašinskog učenja je: „Oblast studiranja koja kompjuterima daje sposobnost da uče bez potrebe da budu eksplicitno programirani“ [1].

Mašinsko učenje i algoritmi pretrage podataka počinju primetno da imaju uticaj u mnogim aplikacijama koje korisnici svakodnevno koriste. Primena ovakvih algoritama je zastupljena u raznim oblastima, mnogim softverskim i hardverskim rešenjima. Algoritmi mašinskog učenja i pretrage podataka rade po principu učenja nad podacima i stvaranja predikcija nad novim podacima.

Stvaranje inteligentnog softvera, aplikacije koja sama ume da odlučuje i da uči nad podacima je upravo bila motivacija za izradu ovog master rada za primenu nekih od algoritama mašinskog učenja na konkretan problem.

Predmet ovog rada je upotreba algoritama za pretragu podataka radi klasifikacije mejl poruka koje pristižu na mejl server. Realizovan sistem je filter komponenta koja se ugrađuje u mejl server.

Stefan Tubić, dipl. master inž. elektrotehnike i računarstva – Elektrotehnički fakultet, Univerzitet u Beogradu, Bulevar kralja Aleksandra 73, 11000 Beograd, Srbija (e-mail: stefan.tubic@etf.bg.ac.rs).

Miloš Cvetanović, doktor elektrotehnike i računarstva – Elektrotehnički fakultet, Univerzitet u Beogradu, Bulevar kralja Aleksandra 73, 11000 Beograd, Srbija (e-mail: cmilos@etf.bg.ac.rs).

Zaharije Radivojević, doktor elektrotehnike i računarstva – Elektrotehnički fakultet, Univerzitet u Beogradu, Bulevar kralja Aleksandra 73, 11000 Beograd, Srbija (e-mail: zaki@etf.bg.ac.rs).

Saša Stojanović, doktor elektrotehnike i računarstva – Elektrotehnički fakultet, Univerzitet u Beogradu, Bulevar kralja Aleksandra 73, 11000 Beograd, Srbija (e-mail: stojsasa@etf.bg.ac.rs).

II. POSTOJEĆI RADOVI I SOFTVER U OBLASTI FILTRIRANJA POŠTE

Algoritmi za filtriranje mejl poruka pripadaju različitim kategorijama u zavisnosti od pristupa koji koriste, kao što su IP blokiranje, crne liste, bele liste, filtriranje na osnovu zaglavlja, filtriranje na osnovu sadržaja poruke.

U poređenju sa drugim metodama filtriranje poruka na osnovu zaglavlja, IP blokiranje, crne liste i bele liste su generalno implementaciono i komputaciono jednostavnije metode koje lice koje šalje spam poruke može lako zaobići na primer promenom svoje IP adrese, email adrese i na druge načine.

Sledeći radovi opisuju načine i analiziraju određene pristupe za filtriranje pošte.

S. Dhanaraj i Dr. V. Karthikeyani [2] su napravili pregled mogućih rešenja u borbi protiv neželjenih mejl poruka čiji sadržaj nije tekst već jedna ili više slika. Metode su podeljene u više kategorija. Metode iz prve kategorije pokušavaju da otklone neželjene poruke pre nego što su poslate. Rešenja su crne, bele liste kreirane po IP adresi, korisniku, kompaniji ili domenu. Druga kategorija metoda omogućava otklanjanje nakon što su poruke poslate. To je omogućeno programima za blokiranje koji onemogućavaju da poruka dospe u poštansko sanduče. Oni mogu raditi korišćenjem baze podataka koja sadrži povratne informacije o neželjenim porukama iz zajednice korisnika. Drugi mehanizam je korišćenje fajervola koji onemogućava neautorizovanim izvorima da šalju poruke u određenu mrežu. Treća kategorija metoda predstavlja zaštitu od poruka koje dospeju u poštansko sanduče. Za ovu zaštitu koriste se mnogi softverski programi koji filtriraju poštu i onemogućavaju klijentskoj aplikaciji da poruke preuzme sa mejl servera. Ovakav softver najčešće vrši klasifikaciju sadržaja poruke korišćenjem nekog klasifikacionog algoritma, među kojima se za klasifikaciju poruka sa slikama dosta dobro pokazao algoritam Naïve Bayes.

Usarat, Aranya i Somsak [3] su opisali primenu genetičkog algoritma na filtriranje mejl poruka. Poruke koje su bile neželjene nazvane su korpus. Implementacija rešenja sastoji se iz dva procesa. Prvi vrši generisanje ključnih reči iz korisnog sadržaja poruke. Drugi proces koristi dobijene ključne reči, generiše hromozom nad kojim vrši genetičke operacije ukrštanje, mutaciju. Nakon što se one izvrše vrši se evaluacija hromozoma fitnes funkcijom. Nakon što se opisani procesi izvrše nad spam porukama vrši se selekcija hromozoma rulet tehnikom, pa će neki od hromozoma učestvovati u filtriranju novih poruka. Filtriranje počinje tako što se novodobijena poruka transformiše u hromozom i uporedi sa selektovanim hromozomima. Ukoliko je preklapanje hromozoma na 3 ili više gena, tada se prototip

hromozomu dodeli jedan poen. Ukoliko je procenat dobijenih poena veći od 30% smatra se da je poruka neželjena.

Pingchuan i Teng-Sheng [4] predložili su rešenje za filtriranje poruka klasifikacijom korisnog sadržaja poruke. Princip je takav da program iz poruka prvo uzme koristan sadržaj, zatim ukloni HTML tagove, time dobije kolekciju reči i na kraju ukloni manje značajne reči koje prebroji. Nakon pretprocesiranja vršena je klasifikacija poruka upotrebom agloritma Naïve Bayes.

Dalje sledi opis softvera napravljen za potrebe filtriranja poruka.

Unutar *Postfix* mejl servera postoji više dodataka [5] koji služe za filtriranje neželjene pošte kao i dodataka koji rutiraju poruke do softvera za filtriranje. Za neželjenu poštu bi se smatrale poruke koje pristižu od određenih adresa, sa određenim neželjenim tekstualnim sadržajem ili poruke koje u sebi sadrže virus ili neki drugi maliciozni softver. U nastavku su navedeni dodaci koji služe u te svrhe.

A. Spam assassin

Koristi robustan frejmvork za bodovanje i dodatke da integriše širok spektar naprednih heuristika i statističkih testova za analizu nad zaglavljima poruka i tekstualnom telu uključujući analizu teksta, Bajesovo filtriranje, DNS liste za blokiranje.

B. Spampd - Spam Proxy Daemon

Spampd je program koji se koristi unutar *e-mail* sistema za dostavu radi skeniranja poruka radi nalaženja mogućih nepoželjnih komercijalnih poruka (odnosno *spam* poruka). On koristi SpamAssassin (*SA*) koji u stvari radi skeniranje poruka. *Spampd* se ponaša kao transparentan *SMTP/LMTP* proksi između dva mejl servera i u toku transakcije prosleđuje mejl poruku ka *SA*, pa je potom prima nazad.

C. Avcheck

Avcheck je jednostavan program koji nije ni *MTA(Mail Transfer Agent)* niti je antivirus softver. On se nalazi između ta dva. Njegova namena je da primi poruku, prosledi je antivirus programu, od njegao dobije informaciju da li poruka sadrži virus, a na osnovu toga odluči koju akciju će sprovesti nad porukom.

III. KORIŠĆENI ALGORITMI ZA KLASIFIKACIJU

U ovom poglavlju biće navedeni i objašnjeni algoritmi za klasifikaciju podataka. Algoritmi za klasifikaciju uče nad određenim skupom podataka, u ovom radu nad skupom podataka dobijenim iz mejl poruka, i kreiraju model za klasifikaciju novih poruka.

A. Stablo odlučivanja - C4.5 algoritam

Problem konstrukcije stabla odlučivanja [10] može se izraziti rekurzijom. Prvo selektovati atribut koji treba smestiti u koren stabla, i napraviti po jednu granu za svaku vrednost tog atributa. Na ovaj način će se početni skup instanci podeliti u onoliko novih skupova koliko ima grana početnog čvora. Nadalje u svakoj novom čvoru potomku treba izabrati novi atribut nad kojim treba izvršiti grananje na isti način. Izgradnja stabla se zaustavlja ukoliko su iskorišćeni svi atributi ili ukoliko su svi listovi čisti, što znači da su u svakom listu instance samo jedne klase.

Potrebno je odrediti koji atribut je najpogodniji za grananje. Najbolje je izabrati atribut čiji su potomci najčistiji, odnosno potomci u kojima su instance samo jedne klase. Dakle, atribut najpogodniji za grananje je onaj koji pruža najveću informacionu dobit, čija definicija sledi u daljem tekstu. Mera čistoće ili infromaciona vrednost je količina informacija potrebna da se nova instanca klasifikuje u jednu od klasa i računa se funkcijom entropije (1):

$$entropy(p_1, p_2, \dots, p_n) = -\sum_{i=1}^{i=n} p_i \cdot \log(p_i). \quad (1)$$

Promenljiva p_i predstavlja verovatnoću da nova instanca pripada klasi *i*. Informaciona dobit se dobija oduzimanjem entropije nakon grananja od entropije pre grananja. Entropija nakon grananja je prosečna informaciona vrednost potomaka uzimajući u obzir broj instanci koje se nalaze u svakom potomku.

B. Učenje bazirano na instancama (K najbližih suseda)

Za razliku od stabla odlučivanja za čije je stvaranje modela potrebno određeno vreme, učenje bazirano na instacama [11] ima kreiran model onog trenutka kada je skup trening instanci definisan. U ovom algoritmu dosta vremena se troši na klasifikaciju novih instanci.

Za početak ćemo uzeti u obzir da klasifikacija instance zavisi samo od jednog suseda, odnosno kada je vrednost parametra k jednaka 1. Da bi se najbliži sused odredio potrebno je odrediti funkciju distance. Za numeričke atribute može se koristiti jednostavno Euklidovo rastojanje (2):

$$\sqrt{\sum (a_i^{(1)} - a_i^{(2)})^2}. \quad (2)$$

Ovde $a_i^{(k)}$ predstavlja *i*-ti atribut za k-tu instancu. U nekim slučajevima Euklidova distanca nije pogodna pa se mogu koristiti druge kao što je Menhetn distanca.

Pre upotrebe algoritma potrebno je numeričke atribute normalizovati na istu skalu, obično od 0 do 1, korišćenjem sledeće formule (3):

$$a_i = \frac{v_i - \min(v_i)}{\max(v_i) - \min(v_i)}. \quad (3)$$

gde je a_i normalizovana vrednost, a v_i nenormalizovana vrednost atributa.

Ukoliko je atribut nominalan tada se distanca određuje funkcijom različitosti koja za dve instance čije vrednosti atributa nisu jednake vraća distancu 1, u suprotnom distancu 0.

U slučaju uticaja više od jednog suseda koristi se princip većine ili princip glasanja okolnih suseda.

C. Naivan Bayes

Algoritam Naivan Bayes (Naïve Bayes) [12] zasniva se na primeni Bajesove teoreme (4):

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}. \quad (4)$$

gde je H hipoteza, E dokaz, P(X) verovatnoća događaja X, P(X,Y) verovatnoća da se dogodi događaj X ukoliko se dogodio događaj Y.

Na ovaj način moguće je izračunati verovatnoću da je instanca određene klase. Nova instanca pripadaće onoj klasi za koju verovatnoća po ovoj teoremi bude najveća.

IV. KORIŠĆENI ALGORITMI ZA PRETPROCESIRANJE

Uspešna pretraga podataka, tj. uspešnost algoritama koji pretražuju podatke, zavisi dosta od kvaliteta samih podataka. U ovom poglavlju biće opisani algoritmi koji vrše pripremu podataka za klasifikaciju.

A. Selekcija atributa

Prilikom određivanja najboljeg skupa atributa korišćena su dva pristupa. Prvi se bazirao na rangiranju atributa, dok se drugi bazirao na evaluiranju kombinacija/skupova atributa.

U prvom pristupu evaluacija atributa je vršena računanjem informacione dobiti u procesu deljenja početnog skupa instanci na onoliko skupova koliko taj atribut ima mogućih vrednosti. Evaluirane atributi se sortiraju po informacionoj dobiti i bira se prvih r atributa.

Drugi pristup daje bolji skup atributa. Ova prednost se plaća sporijom pretragom. Ideja je naći najbolji skup atributa pretraživanjem prostora skupova atributa koji je pojašnjen u sledećem pasusu.

Skup svih mogućih kombinacija atributa inicijalnog skupa podataka jeste prostor skupova atributa. Broj kombinacija se povećava eksponencijalno sa povećanjem broja atributa pa je dosta neefikasno pretraživati čitav prostor. Da bi se problem donekle rešio pribeglo se upotrebi nekoliko algoritama za pretraživanje, a to su selekcija unapred (forward selection), eliminacija unazad (backward elimination), prvo najbolji (best first).

Selekcija unapred ima prazan inicijalan skup atributa. U prvoj iteraciji pravi četiri nova skupa od po jednog atributa dodavanjem po jednog različitog atributa svaki put. Zatim ih evaluira i bira najbolji od njih, tako da ima početni skup za sledeću iteraciju. Dakle problem je rekurzivan. Svaki put na početni skup se dodaje po jedan atribut koji do sada nije bio u skupu i bira se najbolji od „otvorenih“ skupova atributa u toj iteraciji. Ukoliko je taj novoizabrani skup bolji od prethodnog algoritam ide u sledeću iteraciju, ukoliko nije prestaje sa radom i vraća prethodno izabrani najbolji skup atributa. Pretraga prestaje i u slučaju da se dostigne skup sa svim atributima.

Eliminacija unazad radi analogno kao i selekcija unapred s tim što je sada inicjalni skup atributa skup sa svim atributima i prazni se do skupa sa jednim elementom. Uslov kraja je ukoliko u trenutnoj iteraciji nijedan od novih „otvorenih“ skupova nije bolji od početnog skupa za tu iteraciju, ili ukoliko se stigne do skupa sa jednim elementom.

Mana ova dva algoritma je ta što ne garantuju najbolje rešenje jer ne pretražuju čitav prostor odnosno nalaze samo lokalno najbolje rešenje. Sa druge strane pretraga celog prostora je neprihvatljiva jer oduzima mnogo vremena. Rešenje je koristiti „prvo najbolji“ algoritam.

Prvo najbolji može ima dva moda rada. Može pretraživati prostor po principu selekcije unapred ili po principu eliminacije unazad. U tom modu rada je dok ne završi svoje izvršavanje. Razlika u odnosu na prethodna dva algoritma je što ne završava svoj rad kada u trenutnoj itraciji ne može da pronađe bolji skup. Da bi nastavio sa radom on održava sortiranu listu do sada izabranih skupova atributa i u takvoj situaciji bira do sada najbolji skup atributa iz liste koji nije obiđen. Na dalje nastavlja po istom principu pretrage. Algoritam poseduje parametar koji predstavlja broj dozvoljenih uzimanja elementa iz liste koju održava. Ukoliko bi ovaj parametar imao beskonačnu vrednost algoritam bi pretražio čitav prostor. Dakle algoritam „prvo najbolji“ neće naći samo lokalno najbolje rešenje.

Drugo pitanje je evaluacija jednog takvog skupa. Ona se bazira na simetričnoj nesigurnosti (5):

$$U(A,B) = 2\frac{H(A)+H(B)-H(A,B)}{H(A)+H(B)}. \quad (5)$$

gde je H funkcija entropije koja je već opisana u prethodnom poglavlju, a H(A,B) zajednička entropija za atribute A i B i računa se kao entropija uzimajući u obzir zajedničke verovatnoće za sve kombinacije vrednosti atributa A i B. Simetrična nesigurnost (korelacija) uvek leži u opsegu između 0 i 1.

Izbor karakteristika zasnovan na korelaciji (Correlation-based feature selection) je mera koja se koristi za evaluaciju skupa atributa (6):

$$\sum_j U(A_j, C) \,/\, \sqrt{\sum_i \sum_j U(A_i, A_j)} \quad (6)$$

gde je C klasni atribut, A atribut, a indeksi i i j se kreću kroz svih m atributa skupa. Ukoliko je svih m atributa skupa u savršenoj korelaciji sa klasnim atributom i ukoliko su u savršenoj korelaciji jedni sa drugima tada će brojilac razlomka imati vrednost m a imenilac vrednost , dakle evaluacija bi bila 1. Najmanja vrednost evaluacije je 0. Skup atributa će biti najbolju evaluaciju ukoliko neklasni atributi nisu u korelaciji jedni sa drugima i ukoliko su u korelaciji sa klasnim atributom. Na ovaj način će se izbeći i redundantni atributi.

B. Diskretizacija atributa

Određeni algoritmi zahtevaju da atributi budu nominalni, pa su zato korišćene sledeće dve diskretizacione metode:

- diskretizacija po principu jednake širine opsega (equal-width binning)
- diskretizacija sa opsezima sa jednakim brojem instanci (equal-frequency binning)

Prva metoda za svaki atribut pronađe njegovu minimalnu i maksimalnu vrednost i takav opseg vrednosti podeli na određeni broj intervala jednake širine. Na ovaj način može doći do postojanja intervala sa vrlo velikim i vrlo malim brojem instanci.

Druga metoda pronađeni opseg deli tako da u svakom, ukoliko je to moguće, bude podjednak broj instanci, pa je problem koji je postojao u prvoj metodi rešen.

Pored ove dve koje ne koriste vrednost klasnog atributa, postoje i naprednije metode koje to upravo rade. Jedna od njih je diskretizacija zasnovana na entropiji.

V. IMPLEMENTACIJA SOFTVERA

Prilikom realizacije sistema korišćeni su alat NetBeans IDE 8.1 i Weka 3.8.0 [6] – Alat Weka je korišćen kao pomagalo radi testiranja određenih algoritama. Aplikacija je urađena u programskom jeziku Java.

Softver je implementiran sa dva cilja. Prvi je uporedna analiza klasifikacionih algoritama i metoda za pretprocesiranje podataka sa aspekta njihove primenljivosti za potrebe filtriranja neželjenih mejl poruka. Drugi cilj rada je implementacija softverske komponente koja će, u okviru postojećeg *Postfix* mejl servera, obavljati klasifikaciju poruka algoritmom koji se u postupku uporedne analize bude pokazao kao najefikasniji.

U ovom poglavlju biće opisana arhitektura i implementacija softvera. Čitav proces implementacije i dobijanja rezultata sastoji se iz sledećih faza:

1. Pronalaženje i prikupljanje mejlova
2. Kreiranje inicijalnog skupa podataka
3. Pretprocesiranje inicijalnog skupa podataka
4. Uporedna analiza algoritama za pretprocesiranje i algoritama za klasifikaciju

A. *Pronalaženje i prikupljanje mejlova*

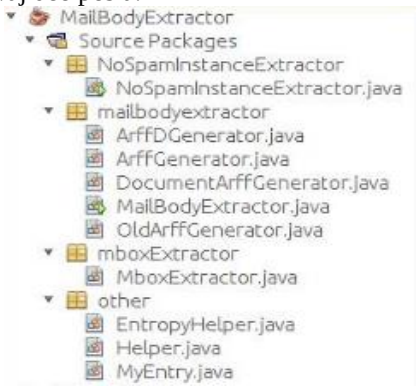
Spam poruke su preuzete sa internet sajta http://untroubled.org/spam/. Donator ovih poruka je Bruce Guenter koji je prikupljao spam poruke od 1998. i prikupljao ih je u toku pisanja ovog rada. Arhiva sadrži više od 3 miliona poruka, što je više nego dovoljno za potrebe rada.

Standardne poruke, odnosno poruke koje nisu spam, su preuzete sa internet sajta https://www.cs.cmu.edu/~./enron/. Donatori ovih poruka su 150 zaposlenih, većinom senior menadžera, u američkoj Enron korporaciji. Mejl poruke ovih korisnika prikupljene su od strane CALO(A Cognitive Assistant that Learns and Organizes) projekta i organizovane po folderima za svakog korisnika posebno. Ova arhiva sadrži oko pola miliona poruka što je bilo dovoljno za potrebe rada.

B. *Kreiranje inicijalnog skupa podataka*

Rezultat ove faze je dobijanje početnog skupa podataka iz prikupljenih poruka koji bi se koristio u sledećim fazama.

Projekat MailBodyExtractor čija je struktura prikazana na slici 1 radi ovaj deo posla.



Sl. 1. Projekat MailBodyExtractor

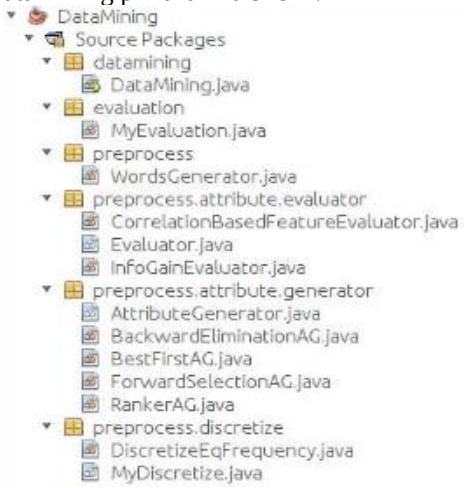
Klasa MailBodyExtractor vrši izvlačenje korisnog sadržaja iz mejl poruka metodom getTextMailContent() koja prvo pozivom metode getMail() dohvata mejl iz fajla, a potom odbacuje sva zaglavlja koja u poruci postoje. Za ove potrebe korišćena je biblioteka *javax.mail* [9]. Mejl poruka je takve strukture da njeno telo može sadržati koristan sadržaj u više formata. Formati na koje se naišlo su html i txt. Ukoliko je poruka sadržala oba formata uziman je html iz koga je potom izvučen sadržaj izostavljajući sve html tagove. Dakle vršeno je parsiranje html koda. Rezultat ove faze je skup fajlova, gde svaki predstavlja koristan sadržaj za jednu mejl poruku.

Klasa DocumentArffGenerator vrši generisanje inicijalnog skupa podataka unutar .arff fajla. To obavlja metodom generateArff() koja prvo generiše atribute pozivom metode generateArffAttributes(), potom i podatke metodom generateArffData(). Metoda generateArffAttributes() generiše samo dva atributa. Prvi predstavlja prethodno dobijen koristan sadržaj iz poruka, a drugi je klasa te poruke (0 – poruka nije spam, 1 – poruka je spam).

Rezultat ove faze su dva .arff fajla. Prvi predstavlja skup podataka za treniranje, a drugi skup podataka za testiranje. Obezbeđeno je da broj instanci jedne i druge klase bude jednak u skupu podataka za treniranje kako bi kasnije generisani modeli za klasifikaciju dali bolje rezultate. Otprilike 2/3 poruka korišćeno za treniranje, a ostatak za testiranje, što se u praksi pokazalo kao dobar odnos. Na ovaj način dobijen je inicijalni skup podataka koji će se u sledećoj fazi koristiti za pretprocesiranje.

C. *Pretprocesiranje inicijalnog skupa podataka*

U inicijalnim .arff fajlovima postoji samo koristan sadržaj poruka i njihova klasa. U ovoj fazi je potrebno definisati atribute koji najbolje karakterišu poruke i omogućavaju njihovu preciznu klasifikaciju. Ova faza se sastoji iz dva prolaza nad podacima. Za ove potrebe implementiran je projekat DataMining prikazan na slici 2.



Sl. 2. Projekat DataMining

U prvom prolazu klasa WordsGenerator vrši generisanje skupa reči koji će se pojaviti kao atributi u inicijalnom skupu atributa. Metoda writeWordsToFile upisuje dobijene reči u fajl prerthodno pozivajući metodu getWords koja prolazi kroz sve fajlove sa korisnim sadržajem i za svaki sadržaj fajla poziva metodu getWordsFromString koja u mapu words unosi pronađene reči i povećava broj pojavljivanja već postojeće reči. U fajl će biti upisane samo one reči čiji je broj pojavljivanja veći ili jedan izrazu (7):

$$\frac{numOfFiles \cdot percentThreshold}{100.00}. \quad (7)$$

Parametar numOfFiles je ukupan broj pređenih fajlova, a percentThreshold ceo broj definisan u klasi Helper. Dodatni uslov da bi reč bila upisana u fajl je i da ona nije nebitna reč. Nebitne reči definisane su u posebnom fajlu i dobijene su metodom getStopWords(). Neke od nebitnih reči u engleskom jeziku su: the, does, but, and, any, as, a, not, of, or, out i druge. Klasa ArffGenerator vrši generisanje .arff fajla koji će u sebi sadržati sledeće atribute:

- reči koje se često pojavljuju (146 atributa)
- znakove iz fiksno definisanog skupa znakova (6 atributa)
- tri specijalna atributa (3 atributa)

Prva grupa atributa dobijena je na već opisan način klasom WordGenerator. Vrednosti atributa u prvoj grupi su brojevi pojavljivanja određene reči. Druga grupa atributa je skup sledećih predefinisanih znakova za koje se smatralo da će poboljšati klasifikaciju: {';', '(', '[', '!', '$', '#'}. Vrednosti atributa u drugoj grupi su brojevi pojavljivanja određenih znakova. Treća grupa atributa su atributi capitalLetterLengthAverage, capitalLetterLengthLongest, capitalLetterNumber. Atribut capitalLetterLengthAverage predstavlja prosečnu dužinu neprekidnog niza velikih slova. Atribut capitalLetterLengthLongest predstavlja najveću dužinu neprekidnog niza velikih slova. Atribut capitalLetterNumber predstavlja ukupan broj velikih slova.

Fajl u .arff formatu se kreira pozivom metode generateArff() koja generiše gore navedene atribute pozivom generateArffAttributes() i podatke pozivom metode generateData(). Da bi se generisali podaci potrebno je proći kroz inicijalni skup podataka (inicijalni .arff fajl) i dobiti broj pojavljivanja definisanih reči, znakova, i vrednosti poslednja tri specijalna atributa. Prebrojavanja vrši metoda getInfoAboutData() koja kao argument prima korisni sadržaj mejl poruke i postavlja statičke atribute klase ArffGenerator. Kako se ovaj posao ponavlja za svaku instancu potrebno je restartovati brojače prilikom svake iteracije metodom restartCounters(). Nakon dobijanja svih potrebnih vrednosti za atribute, novodobijena instanca biva upisana u .arff fajl metodom writeInstanceToArff().

Rezultat prvog prolaza su .arff fajlovi sa trening i test instancama sa 155 numeričkih atributa i jednim klasnim nominalnim atributom.

Drugi prolaz podrazumeva korišćenje .arff fajlova dobijenih prilikom prvog prolaza. Sada je potrebno od inicijalnog skupa atributa selektovati podskup koji se pokaže kao najbolji. S obzirom da su za ove potrebe korišćeni algoritmi za klasifikaciju koji zahtevaju da atributi budu nominalni, svi numerički atributi moraju prvo biti diskretizovani. Sprovodi *equal frequency binning* diskretizacija.

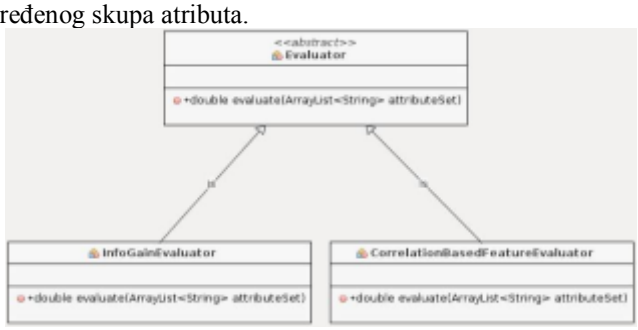
Diskretizaciju sprovodi apstraktna klasa MyDiscretize i njena konkretna podklasa DiscretizeEqFrequency. Klasa MyDiscretize poseduje atribut numOfAttrValues koja definiše broj željenih mogućih vrednosti za svaki od nominalnih atributa koji će biti kreirani. Ova vrednost takođe predstavlja i željeni broj intervala na koji će biti podeljen opseg vrednosti određenog numeričkog atributa. Klasa DiscretizeEqFrequency poseduje metodu discretize() kojom kreira nove vrednosti za atribute u novom skupu podataka. Metoda radi tako što u memoriju učita ceo .arff fajl koji obrađuje. Razlog za to je brže izvršavanje, dok je mana ograničena veličina .arff fajlova koji se mogu koristiti.

Kreiranje novih arff fajlova vrši klasa ArffDGenerator. Metoda generateDArff() generiše .arff fajl sa diskretizovanim atributima pozivajući metode generateDArffAttributes() i generateDArffData(). Metoda generateDArffAttributes() kreira isti broj atributa koji je dobio na ulazu s tim što umesto tipa numeric sada vrši nabrajanje nominalnih vrednosti atributa. Za te vrednosti uzeti su generički nazivi v0, v1 …, vn. Metoda generateDArffData() generiše instance za novi diskretizovani skup podataka. Transformaciju numeričkih vrednosti atributa u nominalne vrši metoda getNominalValues() kojoj se kao parametri prosleđuju instanca, lista atributa, kao i skup graničnih vrednosti za svaki atribut. Povratna vrednost metode getNominalValues() je transformisana instanca koja posle biva upisana u fajl metodom writeInstanceToArff().

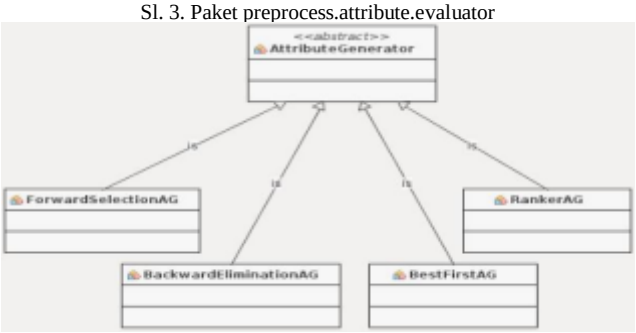
Rezultat diskretizacije je novi par trening i test, sada diskretizovanih skupova podataka predstvljenih u .arff formatu.

Novodobijeni fajlovi još uvek sadrže sve atribute iz inicijalnog skupa. Mnogi od ovih atributa verovatno ne daju valjanu informaciju o tome da li je neka instanca u jednoj ili drugoj klasi, neki od njih su u korelaciji sa drugim neklasnim atributima pa su time redundantni i treba ih izbaciti.

Paketi odgovorni za selekciju atributa su *preprocess.attribute.evaluator* čiji je UML dijagram prikazan na slici 3 i *preprocess.attribute.generator* čiji je UML dijagram prikazan na slici 4. Generatori atributa koriste jedan od 5 algoritama za generisanje najboljeg skupa atributa, dok evaluatore koriste ovih 5 algoritama radi evaluiranja određenog skupa atributa.



Sl. 3. Paket preprocess.attribute.evaluator



Sl. 4. Paket preprocess.attribute.generator

Apstraktna klasa AttributeGenerator prikazana na slici 4 prilikom svake svoje iteracije generiše jedan skup atributa koji apstraktna klasa Evaluator ocenjuje/evaluira. Kroz konstruktor se svakom generatoru atributa prosleđuje i objekat klase koja

će evaluirati skupove atributa. Metoda chooseAttributes() vrši izabir atributa pozivom metode generateAttributes(). Svaki konkretan algoritam ima svoj način određivanja najboljeg skupa atributa pa je metoda generateAttributes() apstraktna.

Klasa ForwardSelectionAG vrši selekciju atributa polazeći od praznog skupa atributa i iterativno povećava broj atributa. U implementaciji metode generateAttributes() se održavaju lista obiđenih i neobiđenih atributa. Sve dok nije zadovoljen uslov kraja algoritam pokušava dodavanjem da dobije bolji skup od dosadašnjeg najboljeg. U unutrašnjoj petlji prolazeći kroz listu neobiđenih atributa program dodaje po jedan atribut iz te liste i evaluatorom procenjuje trenutni skup atributa. Pri svakoj iteraciji spoljašnje petlje bira najbolji skup pronađen u unutrašnjoj petlji. Kada je uslov kraja zadovoljen metoda vraća najbolji skup atributa.

Klasa BackwardEliminationAG vrši selekciju atributa analogno kao klasa ForwardSelectionAG, ali tako što polazi od punog skupa atributa (skup sa svim mogućim atributima) i iterativno smanjuje broj atributa.

Klasa BestFirstAG je implementacija algoritma BestFirst koji je već opisan. Kao što je već pomenuto algoritam može raditi u dva režima rada za koje su zadužene metode generateAttributesForward() i generateAttributesBackward(). U prvom radi po principu algoritma implementiranog klasom ForwardSelectionAG, dok u drugom radi po principu algoritma implementiranog klasom BackwardEliminationAG. Dodatni parametar koji se definiše za algoritam „prvo najbolji“ jeste maksimalni broj pokušaja koji algoritam prilikom pretrage ima.

Poslednja klasa iz ove grupe, klasa RankerAG, vrši jednostavno rangiranje atributa. Dakle metoda generateAttribute() prolazi kroz sve atribute i prepušta ih evaluatoru. Nakon što su atributi dobili ocene, sortira ih i zadržava onoliko prvih atributa koliko ih je definisano poljem numOfAttrToKeep. Na kraju metoda vraća skup zadržanih atributa.

Klasa InfoGainEvaluator vrši evaluaciju atributa tako što izračuna informacioni dobitak za taj atribut. Ova evaluacija može raditi samo u kombinaciji sa generatorom RankerAG.

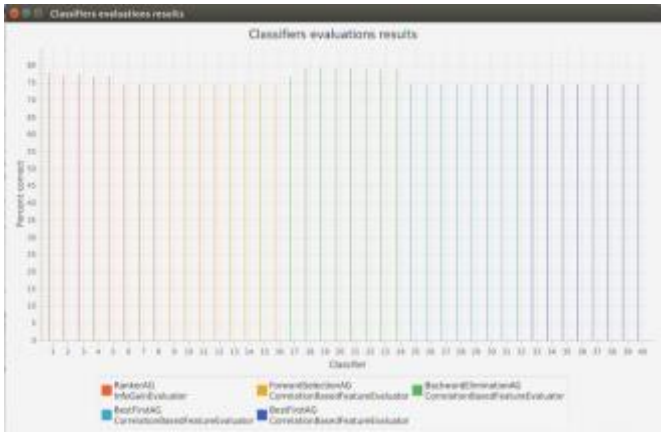
Klasa CorrelationBasedFeatureEvaluator vrši evaluaciju skupa atributa koristeći Correlation-based feature selection metodu. Metoda evaluate() prednost daje skupu atributa, za koje važi da su oni u što manjoj korelaciji jedni sa drugima, a u što većoj korelaciji sa klasnim atributom.

D. Uporedna analiza algoritama za pretprocesiranje i algoritama za klasifikaciju

Opisani algoritmi za pretprocesiranje poruka koriste se u sprezi sa algoritmima za klasifikaciju kako bi se dobio najbolji model za klasifikaciju pristiglih poruka. Korišćeni algoritmi za klasifikaciju implementirani su u okviru paketa weka.jar. To su J48 koji je implementacija algoritma C4.5, IBk koji je implementacija algoritma zasnovanog na instancama (K najbližih suseda) i NaiveBayes koji je implementacija algoritma naivni Bayes. Klasa MyEvaluation kao glavnu funkciju ima izvršavanje uporedne analize. Njene osnovne funkcionalnosti i metode će biti navedene u nastavku.

Prva i osnovna funkcionalnost je uporedna analiza algoritama za pretprocesiranje i algoritama za klasifikaciju. Tu funkciju obavlja metodom getAlgorithmsEvaluations(). Ova metoda u sebi sadrži dve ugneždene petlje. Prva, spoljašnja petlja prolazi kroz skup metoda za pretprocesiranje i za svaku metodu u unutrašnjoj petlji prolazi kroz skup algoritama za klasifikaciju. Na taj način biće dobijene sve moguće kombinacije elemenata skupa za pretprocesiranje i elemenata skupa za klasifikaciju. Za generisanje ovih evaluacija potrebno je dosta vremena zbog velikog broja instanci. Iz tog razloga klasa MyEvaluation omogućava čuvanje trenutne evaluacije u objektni fajl metodom saveAlgorithmsEvaluations(). Ove evaluacije mogu se iz fajla pročitati metodom getAlgorithmsEvaluations(). Druga funkcionalnost je ispis rezultata. Dakle, moguće je ispisati karakteristike najbolje evaluacije metodom printBestAGAndClassifier(). Moguć je ispis karakteristika svih evaluacija redosledom koje su one kreirane ili u sortiranom poretku metodama printEvaluationResults() i printEvaluationResultsSorted() respektivno. Sortiranje evaluacija obavlja se na osnovu procenta tačnih klasifikacija. Treća funkcionalnost jeste izbor najbolje kombinacije metode pretprocesiranja i klasifikatora, odnosno izbor najboljeg klasifikacionog modela koji će se kasnije koristiti na mejl serveru u okviru filter komponente. Ovo je omogućeno metodom chooseBestAGAndClassifier(). Za izbor najboljeg klasifikacionog modela, prilikom upoređivanja takođe je korišćen procenat tačno klasifikovanih instanci. Najbolji izabrani model se može čuvati u posebnom fajlu metodom saveModel().

Prvi rezultat je uporedna analiza algoritama za pretprocesiranje i algoritama za klasifikaciju i dobijanje najboljeg modela, a na slici 5 su prikazani rezultati ove analize. Na x osi su ispisani redni brojevi određenog klasifikacionog modela (redni broj kombinacije algoritma za pretprocesiranje i algoritma za klasifikaciju), a na y osi je prikazan procenat tačnih klasifikacija tog klasifikacionog modela. Gledajući s leva na desno postoji 5 grupa modela i one su redom vezane za algoritme pretprocesiranja: Ranker, Forward Selection, Backward Elimination, BestFirst koji kreće od praznog skupa atributa, BestFirst koji kreće od punog skupa atributa. Svaka grupa sastoji se od 8 algoritama za pretprocesiranje i njihov redosled je već opisan na kraju prethodnog poglavlja. Dakle u svakoj grupi nalazi se sledeći klasifikacioni algoritmi: naivan Bayes, stablo odlučivanja koje nije orezano i nije korišćeno uzdizanje podstabala, stablo odlučivanja koje nije orezano i jeste korišćeno uzdizanje podstabala, stablo odlučivanja koje jeste orezano i nije korišćeno uzdizanje podstabala, stablo odlučivanja koje jeste orezano i jeste korišćeno uzdizanje podstabala, K najbližih suseda za K=1, K najbližih suseda za K=2, K najbližih suseda za K=3.

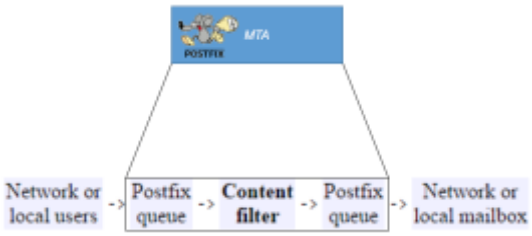


Sl. 5. Rezultati uporedne analize

Sa slike 3 se može videti da se najbolje pokazala kombinacija sa rednim brojem 19 i to je kombinacija eliminacije unazad (BackwardElimination) kao algoritma za pretprocesiranje i stabla odlučivanja koje nije orezano i kod kojeg jeste korišćeno uzdizanje podstabala.

Najbolji model tačno je klasifikovao 21711 poruka, a pogrešno klasifikovao 5559 poruka. Odnosno procenat instanci koji je tačno klasifikovan je 79.615%. Parametri matrice konfuzije su sledeći TP = 7887, TN = 13824, FP = 3249, FN = 2310, gde je TP(true positive) broj instanci koji je pozitivan i klasifikovan kao pozitivan, TN(true negative) broj instanci koji je negativan i klasifikovan kao negativan, FP(false positive) broj instanci koji je negativan i klasifikovan kao pozitivan, FN(false negative) broj instanci koji je pozitivan i klasifikovan kao negativan.

Drugi rezultat ovog rada jeste implementirana filter komponenta za mejl server. Dobijeni najbolji model je korišćen kao klasifikator mejlova koji su pristizali na server. Na slici 6 se može videti da je filter komponenta (*Content filter*) komponenta MTA (Mail Transfer Agent) komponente koja predstavlja deo Postfix mejl servera na Linux operativnom sistemu.



Sl. 6. Filter komponenta [7]

VI. ZAKLJUČAK

Kao što se može videti iz rezultata, ovaj rad predstavlja jedan od načina na koji se algoritmi za pretragu podataka, ujedno i algoritmi mašinskog učenja mogu primeniti unutar jedne aplikacije za klasifikaciju poruka. Aplikacija primenjena nad mejl porukama može se koristiti za klasifikaciju i analizu nad bilo kakvom skupu dokumenata, jer se fokusirala na koristan sadržaj koji se nalazio u mejl porukama.

Izrađena aplikacija, kao dodatak (plugin) za Postfix mejl server, predstavlja gotovu komponentu koja je spremna za upotrebu. Predstavlja jednu modularnu aplikaciju tako da se može lako i brzo proširiti novim funkcionalnostima i zahtevima.

ZAHVALNICA

Ovaj rad je napisan u okviru projekata III44009 i TR32047 kod Ministarstva prosvete, nauke i tehnološkog razvoja Republike Srbije.

LITERATURA

[1] *Machine Learning* [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning (30.05.2017)
[2] S. Dhanaraj and V. Karthikeyani, "A study on e-mail image spam filtering techniques," 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering, Salem, 2013, pp. 49-55.
[3] U. Sanpakdee, A. Walairacht and S. Walairacht, "Adaptive Spam Mail Filtering Using Genetic Algorithm", 2006 8th International Conference Advanced Communication Technology, Phoenix Park, 2006, pp. 441-445.
[4] P. Liu and T. S. Moh, "Content Based Spam E-mail Filtering", 2016 International Conference on Collaboration Technologies and Systems (CTS), Orlando, FL, 2016, pp. 218-224.
[5] *Postfix Add-on Software* [Online]. Available: http://www.postfix.org/addon.html (30.05.2017)
[6] *Weka 3.8.0* [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/ (30.05.2017)
[7] *The Postfix Documentation* [Online]. Available: http://www.postfix.org/documentation.html (30.05.2017)
[8] Ian H.Witten, Eibe Frank, Mark A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, Elsevier, 2011
[9] *JavaMail API* [Online]. Available: http://www.oracle.com/technetwork/java/javamail/index.html (30.05.2017)
[10] *Decision tree* [Online]. Available: https://en.wikipedia.org/wiki/Decision_tree_learning (30.05.2017)
[11] *K-nearest neighbors algorithm* [Online]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm (30.05.2017)
[12] *Naive Bayes* [Online]. Available: https://en.wikipedia.org/wiki/Naive_Bayes_classifier (30.05.2017)

ABSTRACT

Machine learning and data mining algorithms have been frequently used to solve problems in different areas. In this paper it is shown a few algorithms for preprocessing and classification of text in email messages with goal of selecting the best model and its usage within existing Postfix email server. The paper explained algorithms that are used, explained their implementation, shown a method of comparative analysis, which resulted the best model for classification and shown model position as component in mail server.

The implementation of software for filtering unwanted messages using classification algorithms

Stefan Tubić, Miloš Cvetanović, Zaharije Radivojević, Saša Stojanović