SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1410

POSTUPAK KLASIFIKACIJE TEKSTA TEMELJEN NA K-NN METODI I NAIVNOM BAYESOVOM KLASIFIKATORU

Renee Ahel

Zagreb, rujan 2003.

SADRŽAJ

1. UVOD

Motivacija

Tema ovog diplomskog rada jest klasifikacija teksta prema njegovom sadržaju u određeni broj unaprijed definiranih kategorija. Zašto odabrati baš tu temu? Klasifikacija teksta (danas polako ulazi u upotrebu engleski termin *text mining* – nastao od pojma *data mining*) je danas jedna od najrazvijenijih disciplina na području umjetne inteligencije. Vjerojatno najveći razlog tome je popularizacija Interneta kao medija i izvora informacija. S obzirom da Internet raste eksponencijalno, nužno je pronaći način kako tu neprestano rastuću količinu dokumenata pretraživati i pružiti korisniku čim više upotrebljivu informaciju. Relevantnost informacija koje se na zahtjev isporučuju korisniku može se povećati njihovom kategorizacijom. Prije spomenuta brzina rasta Interneta i u skladu s time i količina informacija isključuje mogućnost zapošljavanja ljudi na poslu kategorizacije, i tu na scenu stupaju metode za automatsku klasifikaciju prema sadržaju. S obzirom na način na koji sve metode rade (analiza teksta riječ po riječ) područja primjene su vrlo velika: kategorizacija web stranica, XML dokumenata, filtriranje E – mail poruka (anti – spam softver), otkrivanje neovlaštenog pristupa u računala – otprilike svaka primjena kod koje određeni skup podataka sadrži podskupove koji imaju zajedničke karakteristike. Format podataka s kojima se radi nije bitan, jer se odgovarajućim parserom svaki skup podataka može svesti na najobičniji tekst. Slijedi pregled postojećih metoda učenja za klasifikaciju teksta.

Klasične metode učenja

Temeljne ideje

Prema [1], klasične metode algoritama za klasifikaciju teksta se zasnivaju na pretpostavci da se dokument opisuje nizom reprezentativnih ključnih riječi koje se nazivaju indeksni pojmovi. Indeksni pojam je jednostavno riječ iz dokumenta čija semantika pomaže u memoriranju osnovne tematike pripadajućeg dokumenta. Zato se indeksni pojmovi koriste za indeksiranje i sažimanje sadržaja dokumenta. Indeksni pojmovi su

uglavnom imenice iz prostog razloga što imenice kao riječi imaju same po sebi značenje, a time je i njihova semantika najjednostavnija za razlučiti. Pridjevi i prilozi su mnogo manje upotrebljivi kao indeksni pojmovi jer oni uglavnom služe kao dopuna imenicama. Unatoč tome, za neke primjene (neke Internet pretraživači) se za indeksne pojmove koriste sve različite riječi u dokumentu.

Sam pogled na indeks koji sadrži indeksne pojmove nekog dokumenta ukazuje na činjenicu da svaki indeksni pojam nije jednako koristan za klasifikaciju nekog dokumenta. Također postoje indeksni pojmovi koji su jednostavno po svom značenju mnogo manje jasni od drugih. Stoga je odlučivanje koji indeksni pojam je važan za klasifikaciju, a koji nije jedan od najvažnijih problema pri klasifikaciji, a ujedno i jedan od najsloženijih. Unatoč tome, indeksni pojmovi ipak posjeduju i neke karakteristike po kojima se međusobno razlikuju, pa se one i koriste za ocjenjivanje važnosti pojedinog pojma za klasifikaciju. Npr. riječ koja se pojavljuje u svakom od 100 000 dokumenata je apsolutno nepotrebna s obzirom da ne pruža nikakvu informaciju koja je korisna za klasifikaciju, dok riječ koja se pojavljuje u samo 5 dokumenata može znatno suziti izbor pri kategorizaciji. Zato je jasno da pri opisu teme (kategorije) nekog dokumenta nisu svi indeksni pojmovi jednako važni. Ova se pojava u algoritmima za klasifikaciju prikazuje pomoću težinskih faktora.

Također, svi algoritmi za klasifikaciju koriste pretpostavku da su težine indeksnih pojmova međusobno neovisne. To je očito nužno pojednostavljenje jer je jasno da su riječi u nekom dokumentu smisleno povezane. To pojednostavljenje omogućuje značajno ubrzanje implementacije na računalu, a osim toga neka istraživanja nisu uspjela dokazati značajno poboljšanje performansi klasifikacije uzimanjem u obzir međusobnih veza među pojavama riječi.

Model temeljen na Booleovoj algebri

Model temeljen na Booleovoj algebri [1] je najjednostavniji model metoda učenja koji se temelji na teoriji skupova i Booleovoj algebri. Samim time ciljni koncepti su određeni kao izrazi Booleove algebre, pa kao takvi

imaju vrlo preciznu semantiku. Zbog svoje jednostavnosti i strogo određenog formalizma ovaj je model u prošlosti bio korišten [1] za više komercijalnih bibliografskih sustava.

Nažalost, model temeljen na Booleovoj algebri pati i od značajnih nedostataka. Prvi takav nedostatak je svojstvo da je kriterij za klasifikaciju binaran – dokument ili spada u tu kategoriju ili ne spada – bez ikakve naznake o mogućoj djelomičnoj pripadnosti dokumenta nekoj kategoriji, što u konačnici sprečava bolje rezultate u klasifikaciji. Drugi nedostatak je što nije uvijek jednostavno ciljni koncept prevesti u izraz Booleove algebre.

Vektorski model

Vektorski model [1] nadilazi nedostatak modela temeljenog na Booleovoj algebri i pretpostavlja moguću parcijalnu pripadnost dokumenta kategoriji. To se postiže pridjeljivanjem težina indeksnim pojmovima i dozvoljavanjem da težine poprime bilo koju vrijednost (nisu binarne). Težine indeksnih pojmova se koriste za izračun stupnja sličnosti između naučenih dokumenata i klasificiranog dokumenta. Sortiranjem dokumenata po stupnju sličnosti uzimaju se u obzir dokumenti koji su vrlo slični (pri vrhu) i oni koji su vrlo malo slični klasificiranom dokumentu. Glavni rezultat ovakvog pristupa je mnogo bolji rezultat u klasifikaciji od modela temeljenog na Booleovoj algebri.

Probabilistički model

Zasniva se na ideji [1] da je svaka kategorija dokumenata potpuno opisana dokumentima koji spadaju u tu kategoriju. Zato je u ovom modelu učenje svedeno na određivanje karakteristika kategorije na temelju dokumenata za koje je poznato da u tu kategoriju spadaju. Karakteristike se određuju izgradnjom probabilističke predodžbe izgleda karakteristika kategorije. Zatim se ta predodžba koristi za klasifikaciju tako što se u klasificiranom dokumentu traže karakteristike iz probabilističkog opisa. Što je broj tih karakteristika veći, veća je ukupna vjerojatnost da klasificirani dokument spada u tu kategoriju. U slučaju više kategorija, traži se kategorija sa najvećom vjerojatnosti.

Usporedba klasičnih modela učenja

Generalno gledano, model temeljen na Booleovoj algebri [1] je najlošiji klasični model učenja zbog svoje nemogućnosti prikaza parcijalne pripadnosti nekoj kategoriji, što vodi do loših rezultata klasifikacije. Za druga dva modela je vrlo neizvjesno koji pokazuje bolje rezultate. Neka su mjerenja pokazala da bi vektorski model trebao nadmašiti probabilistički pri podacima općeg sadržaja. Takvo mišljenje danas prevladava u znanstvenim krugovima i šire.

Alternativne metode učenja

Model temeljen na neizrazitoj logici

Predstavljanje dokumenata nizom ključnih riječi dovodi do opisa dokumenata koji su samo djelomično u vezi sa stvarnim semantičkim značenjem dokumenata [1]. Kao posljedica toga, uspoređivanje dokumenata na temelju indeksnih pojmova je približno točno. Ta «približnost» se može modelirati kroz definiranje svake kategorije kao neizrazitog skupa i određivanjem da svaki dokument ima stupanj pripadnosti svakoj od kategorija (taj stupanj je najčešće manji od 1). Klasifikacija dokumenta se sastoji od utvrđivanja stupnja pripadnosti klasificiranog dokumenta svakom od neizrazitih skupova.

Prošireni model temeljen na Booleovoj algebri

Ovaj model kombinira svojstva modela temeljenog na Booleovoj algebri i vektorskog modela [1]. Time se uklanjaju glavni nedostaci modela temeljenog na Booleovoj algebri – nemogućnost djelomične pripadnosti dokumenta kategoriji i nedostatak težinskih faktora. Također se dodaju i dobra svojstva vektorskog modela – jednostavnost, brzina i bolje performanse klasifikacije. Način kombiniranja je slijedeći: dokumenti se prikazuju kao vektori, a logičke operacije potrebne za Booleovu algebru su prilagođene vektorskom prostoru. Time se «razblažuju» stroge algebarske operacije u Booleovoj algebri. Model je predstavljen 1983., ali nije široko prihvaćen.

Model generaliziranog vektorskog prostora

U klasičnom modelu vektorskog prostora česta je interpretacija međusobne neovisnosti indeksnih pojmova, koja nalaže da su međusobno neovisni vektori indeksnih pojmova u vektorskom prostoru ortogonalni. U modelu generaliziranog vektorskog prostora [1] dva vektora indeksnih pojmova ne moraju biti ortogonalni. To se postiže razlaganjem vektora na manje komponente. Ovim proširenjem se zapravo omogućuje interpretacija međusobne ovisnosti indeksnih pojmova – što opet izaziva kontroverze oko pitanja da li uzimanje u obzir međusobne ovisnosti indeksnih pojmova donosi poboljšanje rezultata klasifikacije, i općenitu isplativost takvih implementacija s obzirom da su značajno zahtjevnije od klasičnih metoda koje koriste vektorski prostor.

Model latentnog semantičkog indeksiranja

Nedostaci metoda [1] koje koriste ključne riječi za reprezentaciju dokumenata u nekoj kategoriji su: loši rezultati klasifikacije kada nisu predstavljeni reprezentativni primjeri za učenje i činjenica da dokument koji spada u neku kategoriju a ne sadrži nijednu ključnu riječ kojom je opisana ta kategorija neće biti svrstan u tu kategoriju. Razlog tome je što ključne riječi samo približno opisuju semantiku nekog dokumenta (kategorije). Semantička značenja (glavne ideje) tekstova su puno bolje vezane sa konceptima koji su u tekstu opisani nego sa ključnim riječima koje se pojavljuju u njima. Zato bi se proces klasifikacije mogao temeljiti na usporedbi koncepata, a ne na usporedbi ključnih riječi. Time se izbjegavaju nedostaci ključnih riječi. Ideja algoritma je preslikati svaki dokument u prostor koncepata, koji ima manje dimenzija od klasičnog vektorskog prostora. To se postiže preslikavanjem vektora indeksnih pojmova u prostor koncepata. Pretpostavka je da je klasifikacija u prostoru koncepata po performansama superiorna klasifikaciji u klasičnom vektorskom prostoru.

Model neuronskih mreža

S obzirom da se u većini dosad opisanih metoda radi sa vektorima dokumenata koji sadrže težine pojedinih riječi, prirodno je primijeniti neuronske mreže na problem klasifikacije teksta [1]. Kod procesa učenja

neuronskoj se mreži na ulazni sloj neurona daju vektori dokumenata, a na izlaznom sloju se forsira klasifikacija u pripadajuću kategoriju. Pri klasifikaciji se na ulazni sloj neurona daje vektor klasificiranog dokumenta, a na izlazu se dobiva klasifikacija tog dokumenta.

Bayesove mreže

Bayesove mreže [1] daju mogućnost kombiniranja više različitih izvora znanja (npr. prethodne klasifikacije) u određivanju klasifikacije nekog dokumenta. Bayesova mreža je usmjereni aciklički graf u kojem čvorovi predstavljaju varijable sa slučajnim vrijednostima, a veze među čvorovima predstavljaju odnose među slučajnim varijablama. Odnosi su izraženi uvjetnim vjerojatnostima. Roditelji nekog čvora se smatraju direktnim uzrocima za događaj modeliran tim čvorom. Čvorovi koji nemaju roditelja se nazivaju čvorovi korijena.

Sadržaj rada

Metode koje će biti predstavljene u ovom radu, naivni Bayesov klasifikator i metoda k – najbližih susjeda spadaju u tradicionalne, dobro poznate metode koje su u vrhu po učinkovitosti. Međusobno će biti uspoređene na dva različita skupa podataka za testiranje: «Reuters - 21450 ApteMod» - bazi novinskih članaka iz sekcije kratkih vijesti i «Medline» - bazi kratkih članaka iz časopisa iz područja medicine (izvaci iz jedne od najvećih medicinskih baza članaka).

U nastavku slijede detaljna teoretska objašnjenja funkcioniranja korištenih metoda, zatim prikaz primjeraka testnih podataka, opis načina implementacije, prikaz i komentar dobivenih rezultata i zaključak.

Za implementaciju će biti korišten programski jezik C++ i Microsoft Foundation Classes 6.0.

2. TEORETSKA RAZMATRANJA

Naivni Bayesov klasifikator

Bayesovo učenje općenito

Bayesovo učenje daje probabilistički pristup zaključivanju. Prema [2], temelji se na pretpostavci da je pripadnost (u ovom slučaju nekog teksta) određena distribucijom vjerojatnosti (u ovom slučaju riječi od kojih se tekst sastoji) i da se optimalna klasifikacija može odrediti uzimanjem u obzir distribucije vjerojatnosti riječi od kojih se sastoji klasificirani članak i zatim svrstavanjem u toj distribuciji najsličniju kategoriju. Bayesovo učenje je bitno na području strojnog učenja i zbog svog kvantitativnog pristupa u vrednovanju različitih hipoteza. Isto tako, ovo učenje stvara temelj za algoritme učenja koji direktno manipuliraju vjerojatnostima, a mogu poslužiti i kao podloga za analizu funkcioniranja drugih algoritama koji ne manipuliraju izravno vjerojatnostima.

Najvažnije karakteristike Bayesovog učenja

Gotovo svaka znanstvena metoda ima svoje prednosti i mane.

Prednosti Bayesovog učenja [2] su:

- Svaki naučeni primjer dodatno povećava ili smanjuje vjerojatnost točnosti neke od hipoteza. To pruža znatnu dozu fleksibilnosti u odnosu na algoritme koji potpuno eliminiraju hipoteze iz prostora hipoteza ako nisu konzistentne sa čak i samo jednim primjerom za učenje.
- *A priori* znanje se može ugraditi u proces učenja na način da se prije početka procesa učenja to znanje modelira zadavanjem različitih *a priori* vjerojatnosti hipotezama. Dakle, konačno znanje se sastoji od *a priori* znanja na koje je učenjem dodana distribucija vjerojatnosti primjera za učenje.

- Bayesove metode učenja mogu baratati hipotezama koje imaju ishode ponderirane vjerojatnostima (npr. pacijent ima 96% šansi za potpuni oporavak od upale pluća).
- Klasifikacija se može vršiti kombiniranjem više hipoteza koje su težinski određene svojim vjerojatnostima.
- U slučajevima gdje se pokazuje da su Bayesove metode učenja računski prezahtjevne, svejedno mogu poslužiti kao teoretski najoptimalniji način odlučivanja s kojim se onda druge, praktično izvedive metode mogu uspoređivati.

Nedostaci Bayesovog učenja [2] su praktične prirode:

- Potrebno je poznavanje *a priori* vjerojatnosti za svaku od hipoteza prije početka procesa učenja. To može predstavljati problem jer je u većini primjena broj hipoteza značajan. Ukoliko nisu poznate *a priori* vjerojatnosti, one se mogu ili procijeniti na temelju prethodnog znanja o problemu, dostupnih podataka ili se mogu pretpostaviti. Za slučaj potpunog nedostatka informacija mogu se svim hipotezama naprosto dodijeliti jednake vjerojatnosti.
- U općem slučaju je za određivanje optimalne hipoteze potrebna značajna količina računalnog vremena – potrebno vrijeme raste linearno s povećanjem broja hipoteza. Ipak, u specijalnim slučajevima je raznim postupcima moguće potrebnu količinu računalnog vremena smanjiti.

Bayesov teorem

U većini slučajeva u strojnom učenju nastoji se pronaći hipoteza iz prostora hipoteza H koja će najbolje predstavljati primjere za učenje iz skupa primjera za učenje D. Jedan od načina definiranja najbolje hipoteze može biti hipoteza koja ima najveću vjerojatnost točnosti, uz određene *a priori* vjerojatnosti hipoteza i skup primjera za učenje D. Bayesov teorem omogućuje upravo takve vjerojatnosti. Točnije, on omogućuje izračun

vjerojatnosti hipoteze iz njene *a priori* vjerojatnosti, vjerojatnosti pojave podataka ako je hipoteza točna i predočenih podataka.

Za matematičku definiciju Bayesovog teorema [2] potrebno je uvesti nešto notacije iz područja stohastičke matematike. Sa $p(h)$ će biti označena *a priori* vjerojatnost hipoteze. $p(h)$ predstavlja bilo kakvo prethodno znanje o hipotezi koje može utjecati na vjerojatnost točnosti te hipoteze. Ako takvo znanje nije raspoloživo, može se svim hipotezama pridijeliti jednak $p(h)$. Slično tome, potrebno je uvesti $p(D)$ koji predstavlja vjerojatnost pojavljivanja podatka D u podacima (tu se pod podaci misli na sve podatke koji su predstavljeni klasifikatoru). Nužna je i veličina $p(D|h)$, vjerojatnost pojavljivanja podatka D ako je hipoteza h točna. Da bi se odredila najvjerojatnija hipoteza potrebno je utvrditi veličinu «suprotnu» posljednjoj navedenoj – $p(h|D)$ – koja predstavlja vjerojatnost da je hipoteza h točna ako je predočen podatak D. Veličina $p(h|D)$ se naziva *a posteriornom* vjerojatnošću točnosti hipoteze h, jer odražava «uvjerenost» da je hipoteza h točna nakon što je predočen podatak D. Također, $p(h|D)$ odražava utjecaj pojave podatka na vjerojatnost da je hipoteza h točna, za razliku od *a priori* vjerojatnosti $p(h)$, koja je neovisna o pojavi podatka D.

Bayesov teorem je temelj svih metoda Bayesianskog učenja jer omogućava izračunavanje *a posteriori* vjerojatnosti $p(h|D)$ iz *a priori* vjerojatnosti $p(h)$, $p(D)$ i $p(D|h)$. Formula Bayesovog teorema [12] glasi:

$$p(h \mid D) = \frac{p(D \mid h)\,p(h)}{p(D)}$$

Kao što je i vidljivo iz jednadžbe, $p(h|D)$ raste kada rastu $p(D|h)$ i $p(h)$ i pada kada raste $p(D)$ što je i logično, jer se porastom vjerojatnosti pojave podatka D neovisno o pojavi hipoteze h smanjuje vrijednost pojave podatka D kao dokaza u prilog točnosti hipoteze h.

U većini slučajeva u strojnom učenju algoritam za učenje mora iz nekog prostora hipoteza H uz pomoć skupa podataka za učenje D razlučiti najvjerojatniju hipotezu $h \in H$ (ili njih nekoliko najvjerojatnijih, ukoliko ih je

više). Takva najvjerojatnija hipoteza naziva se MAP hipoteza (od engl. *maximum a posteriori*). Korištenjem Bayesovog teorema MAP hipoteza se može odrediti računanjem *a posteriori* vjerojatnosti za svaku hipotezu u prostoru hipoteza. Veličina h_{MAP} je vjerojatnost MAP hipoteze, a računa se na slijedeći način:

$$h_{MAP} = \arg\max_{h \in H} p(h \mid D)$$

$$= \arg\max_{h \in H} \frac{p(D \mid h)\, p(h)}{p(D)}$$

$$= \arg\max_{h \in H} p(D \mid h)\, p(h)$$

U posljednjem retku je izbačen $p(D)$ iz nazivnika jer on ne ovisi o hipotezi h.

U slučajevima kada *a priori* vjerojatnosti hipoteza nisu raspoložive najčešće se mogu pretpostaviti jednake *a priori* vjerojatnosti za sve hipoteze ($p(h_i) = p(h_j)$ za sve h_i i h_j iz prostora hipoteza H). U takvim slučajevima se formula za izračun MAP hipoteze dodatno pojednostavljuje i svodi na:

$$h_{MAP} = \arg\max_{h \in H} p(D \mid h)$$

U ovakvim slučajevima MAP hipoteza se naziva i ML hipotezom (od engl. maximum likelihood) – hipotezom najveće vjerojatnosti.

Bayesov teorem se jednako dobro primjenjuje u slučajevima kada se radi o nekoliko međusobno neovisnih hipoteza i skupom za učenje pomoću kojeg tražimo hipotezu koja se najbolje slaže sa primjerima, i u slučajevima kada se radi o skupu međusobno isključivih hipoteza čija je ukupna vjerojatnost jednaka 1. Slijedi primjer [2] koji će ilustrirati primjenu Bayesovog teorema za drugi navedeni slučaj.

Primjer upotrebe Bayesovog teorema

Za ilustraciju Bayesovog teorema dobro će poslužiti problem iz medicinske dijagnostike [2] gdje postoje dvije međusobno isključive hipoteze:

1. Pacijent boluje od rijetke vrste raka,
2. Pacijent ne boluje od rijetke vrste raka.

Podaci laboratorijskih nalaza mogu biti pozitivni i negativni (pozitivan nalaz znači da pacijent ima rak, a negativan da pacijent nema rak). Također postoji podatak da u cijeloj populaciji samo 0.8% ljudi boluje od te vrste raka (dakle statistička *a priori* vjerojatnost da pacijent boluje od te vrste raka jest 0.008). Također, laboratorijski nalazi su podložni pogrešci. Nalaz je pozitivan i točan u 98% slučajeva kada pacijent stvarno ima rak, a negativan i točan u 97% slučajeva kada pacijent stvarno nema rak. Pogrešan nalaz vraća suprotan rezultat (ako pacijent ima rak a nalaz je pogrešan nalaz će biti negativan, a ako pacijent nema rak a nalaz je pogrešan nalaz će biti pozitivan). Dosad navedeni podaci se mogu izraziti pomoću vjerojatnosti na slijedeći način:

$$P(rak) = 0.008$$
$$P(\neg rak) = 0.992$$
$$P(\oplus \mid rak) = 0.98$$
$$P(\otimes \mid rak) = 0.02$$
$$P(\oplus \mid \neg rak) = 0.03$$
$$P(\otimes \mid \neg rak) = 0.97$$

Novi pacijent je bio na pretragama i laboratorijski nalaz je pozitivan. Kakva se dijagnoza može postaviti s obzirom na raspoloživo znanje? Korištenjem formule Bayesovog teorema za nalaženje MAP hipoteze slijedi:

$$P(\oplus \mid rak)P(rak) = 0.98 * 0.008 = 0.0078$$
$$P(\oplus \mid \neg rak)P(\neg rak) = 0.03 * 0.992 = 0.0298$$

Iz rezultata je vidljivo da je MAP hipoteza za ovaj slučaj hipoteza da pacijent nema rak, jer je njena *a posteriori* vjerojatnost veća. S obzirom da je za

izračun korištena pojednostavljena verzija Bayesovog teorema, a hipoteze jesu međusobno isključive, moguće je normalizirati rezultate tako da njihov zbroj bude jednak 1:

$$P(rak \mid \oplus) = \frac{0.0078}{0.0078 + 0.0298} = 0.21$$

$$P(\neg rak \mid \oplus) = \frac{0.0298}{0.0078 + 0.0298} = 0.79$$

Primjer ilustrira kako Bayesovo zaključivanje jako ovisi o *a priori* vjerojatnostima hipoteza, koje moraju biti definirane u slučaju direktne primjene Bayesovog teorema. Također je dobro pokazana dobra strana ove metode – hipoteze se nikad ne eliminiraju iz prostora hipoteza, već samo sa novim predočenim primjerima postaju više ili manje vjerojatne.

Naivni Bayesov klasifikator – teorija

Jedna od praktično primjenjivijih metoda Bayesovog učenja je naivni Bayesov algoritam za učenje, popularnije nazvan naivni Bayesov klasifikator. U nekim primjenama njegova učinkovitost [2] se može mjeriti sa neuronskim mrežama i stablom odlučivanja.

Naivni Bayesov klasifikator se primjenjuje u slučajevima gdje se primjer podatka za učenje može prikazati kao konjukcija atributa koji mogu poprimiti određeni (konačan) skup vrijednosti, a ciljna funkcija može poprimiti bilo koju vrijednost iz konačnog skupa vrijednosti V. Za određivanje ciljne funkcije klasifikator uči iz skupa primjera za učenje i nakon toga mu se predstavljaju novi primjeri. Primjeri su predstavljeni n-torkom vrijednosti atributa (a_1, a_2, ... ,a_n) – kao i primjeri iz skupa za učenje. Zadatak klasifikatora je da predvidi kategoriju novog primjera.

Pristup Bayesovog učenja klasifikaciji [2] jest pridjeljivanje najvjerojatnije kategorije, v_{MAP}, uz dane vrijednosti atributa (a_1, a_2, ... ,a_n) koje opisuju primjer koji klasificiramo.

$$v_{MAP} = \arg\max_{v_j \in V} p(v_j \mid a_1, a_2 ... a_n)$$

Pomoću Bayesovog teorema se gornji izraz može raspisati:

$$v_{MAP} = \arg\max_{v_j \in V} \frac{p(a_1, a_2 ... a_n \mid v_j) p(v_j)}{p(a_1, a_2 ... a_n)}$$

$$v_{MAP} = \arg\max_{v_j \in V} p(a_1, a_2 ... a_n \mid v_j) p(v_j)$$

Sada je moguće pokušati procijeniti veličine potrebne za izračun posljednje formule. I dok se $P(v_j)$ može odrediti bez problema (jednostavnim brojanjem koliko se puta koja kategorija pojavljuje u primjerima za učenje), $p(a_1, a_2 ... a_n|v_j)$ nije lako odrediti. Naime, da bi se odredile $p(a_1, a_2 ... a_n|v_j)$ za sve

moguće kombinacije vrijednosti atributa potrebna je ogromna količina podataka u skupu za učenje – naprosto zato što je broj različitih primjera potrebnih za učenje jednak svim mogućim kombinacijama vrijednosti atributa puta broj kategorija. Također bi za neke pouzdane podatke trebalo svaku od tih kombinacija vidjeti u više od jednog primjera, što vodi do astronomskih brojki primjera za učenje.

Zbog prethodno navedenih razloga se Bayesov naivni klasifikator temelji na pretpostavci da su pojave vrijednosti različitih atributa međusobno nezavisne. Zbog te pretpostavke (koja značajno pojednostavljuje problem) izračunavanje vjerojatnosti pojave konjukcije vrijednosti atributa uz danu kategoriju se svodi na računanje umnoška vjerojatnosti pojave vrijednosti svakog atributa posebno uz danu kategoriju:

$$p(a_1, a_2 ... a_n \mid v_j) = \prod_i p(a_i \mid v_j)$$

Ako se sada gornji izraz supstituira u formulu Bayesovog teorema dobiva se:

$$v_{NB} = \arg\max_{v_j \in V} p(v_j) \prod_i p(a_i \mid v_j)$$

I time je određena formula koja se koristi za naivni Bayesov klasifikator [2]. v_{NB} predstavlja rezultat klasifikacije Bayesovog naivnog klasifikatora. Primijenjenim pojednostavljenjem broj podataka koji se moraju ustanoviti – tu se misli na $p(a_i|v_j)$ – se svodi na broj različitih vrijednosti atributa puta broj različitih kategorija, a to čini mnogostruko manji broj u odnosu na broj kod računanja $p(a_1, a_2 ... a_n|v_j)$.

Metoda naivnog Bayesovog klasifikatora se sastoji od dvije faze: faze učenja i faze klasifikacije. Faza učenja uključuje određivanje svih relevantnih veličina $p(v_j)$ (za sve kategorije) i $P(a_i|v_j)$ (za sve vrijednosti atributa) na temelju frekvencije njihovog pojavljivanja u testnom skupu primjera. Određivanje tih vrijednosti zapravo predstavlja učenje hipoteze. U fazi učenja

se tada pomoću te hipoteze klasificiraju novi primjeri korištenjem formule za naivni Bayesov klasifikator.

Važno je primijetiti razliku [2] između naivnog Bayesovog klasifikatora i ostalih metoda strojnog učenja – naivni Bayesov klasifikator ne pretražuje eksplicitno čitav prostor hipoteza (koji je definiran svim mogućim kategorijama i svim mogućim vrijednostima atributa). Umjesto toga, ispravna hipoteza se gradi određivanjem frekvencije pojavljivanja kategorija i vrijednosti atributa.

Naivni Bayesov klasifikator – primjer

Rad klasifikatora se može pokazati na primjeru [2]: učenja koncepta «Dan za igranje tenisa». Radi se o problemu određivanja da li je pojedini dan pogodan za igranje tenisa – moguće klasifikacije su «Da» i «Ne». Svaki dan se prikazuje nizom atributa - «Vrijeme», «Temperatura», «Vlažnost» i «Vjetar». Primjeri iz skupa za učenje:

Dan	Vrijeme	Temperatura	Vlažnost	Vjetar	Dan za tenis
D1	Sunčano	Vruće	Visoka	Slab	Ne
D2	Sunčano	Vruće	Visoka	Jak	Ne
D3	Oblačno	Vruće	Visoka	Slab	Da
D4	Kišno	Ugodno	Visoka	Slab	Da
D5	Kišno	Hladno	Normalna	Slab	Da
D6	Kišno	Hladno	Normalna	Jak	Ne
D7	Oblačno	Hladno	Normalna	Jak	Da
D8	Sunčano	Ugodno	Visoka	Slab	Ne
D9	Sunčano	Hladno	Normalna	Slab	Da
D10	Kišno	Ugodno	Normalna	Slab	Da
D11	Sunčano	Ugodno	Normalna	Jak	Da
D12	Oblačno	Ugodno	Visoka	Jak	Da
D13	Oblačno	Vruće	Normalna	Slab	Da
D14	Kišno	Ugodno	Visoka	Jak	Ne

Tablica 1. Opis primjera iz skupa za učenje

Očito, vrijednosti po atributima su:

- Vrijeme: Oblačno, Kišno, Sunčano
- Temperatura: Ugodno, Vruće, Hladno
- Vlažnost: Visoka, Normalna
- Vjetar: Jak, Slab

Pretpostavka je da je Bayesov naivni klasifikator naučio skup primjera za učenje i da mu se sada za klasifikaciju predstavlja novi primjer:

(Vrijeme = sunčano, Temperatura = hladno, Vlažnost = visoka, Vjetar = jak)

Zadatak klasifikatora jest predvidjeti klasifikaciju primjera za naučeni ciljni koncept «Dan za igranje tenisa». Jednadžba naivnog Bayesovog klasifikatora za ovaj primjer:

$$v_{NB} = \arg\max_{v_j \in [Da, Ne]} p(v_j) \prod_i p(a_i \mid v_j)$$

$$= \arg\max_{v_j \in [Da, Ne]} p(v_j) P(Vrijeme = suncano \mid v_j) p(Temperatura = hladno \mid v_j)$$
$$p(Vlažnost = visoka \mid v_j) p(Vjetar = jak \mid v_j)$$

U posljednjem je retku varijabla a_i nadomještena sa konkretnim vrijednostima. Za izračun je potrebno odrediti 10 različitih vrijednosti (po 4 vjerojatnosti za navedene vrijednosti atributa za svaku kategoriju, plus *a priori* vjerojatnosti za svaku od kategorija). Prvo se određuju *a priori* vjerojatnosti po kategorijama:

$$p(DanZaTenis = Da) = 9/14 = 0.64$$
$$p(DanZaTenis = Ne) = 5/14 = 0.36$$

Na sličan način se računaju i ostale potrebne vrijednosti, npr. za vjetar:

$$p(Vjetar = jak \mid DanZaTenis = Da) = 3/9 = 0.33$$
$$p(Vjetar = jak \mid DanZaTenis = Ne) = 3/5 = 0.6$$

Nakon što su izračunate sve potrebne vrijednosti, računaju se vjerojatnosti potrebne za klasifikaciju:

$$p(da)\,p(suncano \mid da)\,p(hladno \mid da)\,p(visoka \mid da)\,p(jak \mid da) = 0.0053$$
$$p(ne)\,p(suncano \mid ne)\,p(hladno \mid ne)\,p(visoka \mid ne)\,p(jak \mid ne) = 0.0206$$

Na osnovi dobivenih vjerojatnosti, Bayesov naivni klasifikator daje odgovor «Dan za igranje tenisa» = «Ne». Kao i u primjeru za Bayesov teorem, kategorije u ovom primjeru su neovisne i međusobno isključive, pa se njihove *a posteriori* vjerojatnosti mogu normalizirati tako da njihov zbroj bude jednak jedan:

$$p(da \mid suncano, hladno, visoka, jak) = \frac{0.0053}{0.0053 + 0.0206} = 0.2$$
$$p(ne \mid suncano, hladno, visoka, jak) = \frac{0.0206}{0.0053 + 0.0206} = 0.8$$

Procjenjivanje vjerojatnosti za naivni Bayesov klasifikator

Do sada su se vjerojatnosti računale kao broj pojavljivanja (vrijednosti atributa ili kategorije) podijeljeno sa ukupnim brojem primjera za učenje. Primjerice, vjerojatnost *p(Vjetar=jak|Dan za tenis = ne)* iz prethodnog primjera se računala kao $\frac{n_c}{n}$, gdje je $n = 5$ ukupan broj primjera kod kojih je *DanZaTenis = ne*, a $n_c = 3$ broj primjera (unutar onih 5) za koje vrijedi *Vjetar = jak*.

I dok je navedeni način računanja u mnogo slučajeva sasvim zadovoljavajući, daje loše rezultate kada je n_c mali broj. Primjer: slučaj u kojem je (općenito) vjerojatnost *p(Vjetar=jak|Dan za tenis = ne)* jednaka 0.08, a u skupu primjera za učenje postoji samo 5 primjera za koje vrijedi *Dan za tenis = ne*. Tada je najvjerojatnije vrijednost n_c jednaka nuli, što je problem. Prvo,

$\frac{n_c}{n}$ daje 0, a to predstavlja pristranu procjenu (potcjenjivanje) vjerojatnosti. Drugo, ta nula se uvrštava u formulu za računanje klasifikacije, i to u umnožak, što za sobom povlači da će vrijednost atributa *Vjetar=jak* dominirati svakim slijedećim primjerom i uzrokovati klasifikaciju svakog takvog primjera *Dan za tenis = da*.

Da bi se ovakvi problemi izbjegli, potrebno je primijeniti Bayesov pristup procjenjivanju vjerojatnosti, korištenjem tzv. m - procjene vjerojatnosti [2], koja se definira kao:

$$\frac{n_c + mp}{n + m}$$

Ovdje n_c i n imaju isto značenje kao i prije, p predstavlja procjenu *a priori* vjerojatnosti, a m se naziva ekvivalentna veličina uzorka, i predstavlja težinski faktor koji određuje važnost *a priori* vjerojatnosti vrijednosti atributa u odnosu na ukupan skup primjera za učenje. Uobičajen način za određivanje p u nedostatku informacije o *a priori* vjerojatnosti jest raspodjela jednakih težina svim vrijednostima tog atributa – ako atribut ima k vrijednosti, tada sve vrijednosti tog atributa imaju težinu $p=\frac{1}{k}$. Na primjer, za procjenu *p(Vjetar=jak|Dan za tenis = ne)* poznato je da atribut *Vjetar* ima dvije vrijednosti, dakle $p = 0.5$. Treba primijetiti da se za $m = 0$ izraz za m – procjenu vjerojatnosti svodi na prethodno korišten, $\frac{n_c}{n}$. Ako su pak i m i p različiti od nule, *a priori* vjerojatnost se kombinira sa $\frac{n_c}{n}$ regulirano težinskim faktorom m. Razlog zašto se m zove ekvivalentna veličina uzorka proizlazi iz činjenice da se on može interpretirati kao pojačanje postojećih primjera za učenje sa dodatnih m virtualnih primjera za učenje distribuiranih prema p.

Primjena Bayesovog naivnog klasifikatora za klasifikaciju teksta

Razmatrat će se klasifikator [2] koji je temeljen na slijedećim pretpostavkama: postoji prostor X svih mogućih tekstualnih dokumenata (dakle, svih mogućih nizova riječi i interpunkcija). Također postoji skup

dokumenata za učenje ciljne funkcije *f(x)*, koja poprima bilo koju vrijednost iz nekog konačnog skupa V. Zadatak klasifikatora je naučiti ciljnu funkciju i predvidjeti klasifikaciju svih slijedećih primjera. Za primjer se može uzeti klasifikacija dokumenata na one koji se sviđaju nekoj osobi i na one koji joj se ne sviđaju. Zato će mogući izlaz ciljne funkcije biti «Sviđa» ili «Ne sviđa».

Dva glavna inženjerska problema pri primjeni naivnog Bayesovog klasifikatora za klasifikaciju tekstualnih dokumenata jesu kako predstaviti dokumente pomoću skupa atributa i kako odrediti vjerojatnosti potrebne za izračun klasifikacije.

Pristup koji će ovdje biti korišten je prilično jednostavan: svaka riječ u dokumentu postat će atribut, a njen sadržaj (tj. sama riječ) bit će vrijednost tog atributa. Treba primijetiti da će ovim pristupom duži dokumenti imati više atributa, a kraći manje, ali to neće predstavljati problem.

Nakon što je definiran način reprezentacije dokumenata može se upotrijebiti naivni Bayesov klasifikator. Radi konkretnosti, može se pretpostaviti da postoji 700 primjera za učenje koji su klasificirani kao oni koji se nekoj osobi sviđaju i 300 primjera za učenje koji su klasificirani kao oni koji se istoj osobi ne sviđaju. Nakon učenja se klasifikatoru predstavlja novi dokument za klasifikaciju – radi konkretnosti je moguće pretpostaviti da se radi o prethodnom odlomku. Računamo klasifikaciju kao:

$$v_{NB} = \underset{v_j \in [Svida, NeSvida]}{\arg\max} p(v_j) \prod_{i=1}^{47} p(a_i \mid v_j)$$

$$v_{NB} = \underset{v_j \in [Svida, NeSvida]}{\arg\max} p(v_j) p(a_1 = "pristup" \mid v_j) p(a_2 = "koji" \mid v_j) \ldots p(a_{47} = "problem" \mid v_j)$$

Klasifikacija naivnog Bayesovog klasifikatora maksimizira vjerojatnost nalaženja riječi koje su stvarno pronađene u nekom dokumentu iz skupa za učenje, pod pretpostavkom da su vjerojatnosti pojavljivanja riječi međusobno neovisne. Pretpostavka o neovisnosti pojavljivanja riječi u tekstu koja omogućuje pojednostavljenje

$$p(a_1, a_2 ... a_{47} \mid v_j) = \prod_{1}^{47} p(a_i \mid v_j)$$

temelji se na zaključku da su vjerojatnosti pojavljivanja riječi na nekoj poziciji u tekstu potpuno neovisne o vjerojatnostima pojavljivanja riječi na nekoj drugoj poziciji u istom tekstu, uz pripadnost teksta klasifikaciji v_j. Međutim, očito je da je ta pretpostavka netočna. Na primjer, vjerojatnost da se na nekom mjestu u tekstu pojavi riječ «učenje» je veća ako se na mjestu prije pojavila riječ «strojno» (tvrdnja dakako vrijedi samo ako se radi o tekstu koji se tiče strojnog učenja). Unatoč toj pogrešnoj pretpostavci, njeno korištenje je neizbježno, jer se u suprotnom broj vjerojatnosti s kojima se treba računati penje do astronomskih brojki. Srećom, u praktičnoj primjeni naivni Bayesov klasifikator ima izvrsne rezultate u klasifikaciji teksta, bez obzira na fundamentalnu «pogrešku» u dizajnu.

Za izračun v_{NB} korištenjem prethodno navedenih izraza potrebno je odrediti vjerojatnosti $p(vj)$ i $p(a_i=w_k|v_j)$ (gdje je w_k k-ta riječ u hrvatskom riječniku). Prva vjerojatnost se lako može izračunati iz udjela pojedine kategorije u skupu primjera za učenje: $p(sviđa)$ = 0.3, $p(Ne\ sviđa)$ = 0.7. Procjenjivanje uvjetnih vjerojatnosti za svaku riječ i svaki razred (npr. $p(«pristup» \mid Ne\ sviđa)$) je mnogo složeniji zadatak, jer je potrebno izračunati takvu vjerojatnost za svaku poziciju u tekstu, svaku riječ iz hrvatskog riječnika i svaku kategoriju. Nažalost, u hrvatskom riječniku ima preko 50000 riječi, a postoje 2 kategorije i 47 mogućih mjesta u trenutnom primjeru, pa bi količina uvjetnih vjerojatnosti koje je potrebno izračunati bila 2 * 50000 * 47 = 4700000.

Srećom, moguće je dodatno reducirati broj uvjetnih vjerojatnosti. Naime, prilično je razumno pretpostaviti da je uvjetna vjerojatnost pojavljivanja neke riječi u nekom tekstu neovisna o njenom položaju u samom tekstu. Formalno rečeno, pretpostavka implicira da su atributi međusobno neovisni i identično distribuirani uz danu kategoriju – $p(a_i = w_k \mid v_j) = p(a_m = w_k \mid v_j)$ za sve i, j, k, m. To znači da se cijeli skup uvjetnih vjerojatnosti $p(a_1 = w_k \mid v_j)$, $p(a_2 = w_k \mid v_j)$ ... zamjenjuje jednom vjerojatnošću, koja je neovisna o poziciji u tekstu. S time se broj vjerojatnosti spušta na 2 *

50000 = 100000. To je i dalje velik broj koji zahtijeva dosta računanja, ali je ipak izvediv na današnjim računalima. Posljednje pojednostavljenje se pokazuje posebno korisno kada je raspoloživa ograničena količina primjera za učenje, jer se tada primjenom te pretpostavke povećava broj primjera za računanje pojedinih vjerojatnosti, a time se i povećava pouzdanost procjene.

Ostaje još odrediti način procjene vjerojatnosti – mada je to pitanje već riješeno u odjeljku o načinima procjenjivanja vjerojatnosti, pa je zato potrebno samo prilagoditi formulu za klasifikaciju teksta:

$$p(w_k \mid v_j) = \frac{n_k + 1}{n + |Rijecnik|}$$

n predstavlja ukupan broj riječi u primjerima za učenje čija je kategorija v_j, n_k je broj pojavljivanja riječi w_k unutar broja n, a $Riječnik$ je ukupan broj različitih riječi unutar primjera za učenje.

Metoda k – najbližih susjeda

Učenje temeljeno na primjerima - općenito

Suprotno od drugih metoda na području strojnog učenja [2], koje konstruiraju općenit, generalan oblik ciljne funkcije na temelju primjera za učenje, metode učenja na temelju primjera tijekom učenja jednostavno pohranjuju predočene primjere, dok na temelju njih generaliziraju tek kod klasifikacije. Svaki novi primjer za klasifikaciju se uspoređuje po sličnosti sa pohranjenim primjerima za učenje da bi se utvrdila njegova klasifikacija. Učenje na temelju primjera ima dvije glavne metode učenja: k – nearest neighbor (k – najbliži susjed) i regresija sa lokalnim težinskim faktorima. Obje metode pretpostavljaju da se primjeri za učenje i klasifikaciju mogu predstaviti točkama u Euklidskom prostoru. Postoje i metode koje koriste složeniju, simboličku reprezentaciju primjera za učenje. Sve navedene

metode se ponekad nazivaju i «lijenim» metodama za učenje jer odgađaju izgrađivanje generalne ciljne funkcije za klasificiranje do trenutka kada treba klasificirati novi primjer. Time zapravo nikad ne izgrađuju općenitu ciljnu funkciju za cijeli prostor primjera – što je zapravo prednost, jer se umjesto jedne ciljne funkcije za cijeli prostor gradi nova, lokalna ciljna funkcija za dio prostora primjera u kojem se nalazi klasificirani primjer. To je svojstvo naročito korisno kada je ciljna funkcija prekompleksna za definiranje nad čitavim prostorom, ali se ipak može prikazati skupom lokalnih aproksimacija, u dijelovima prostora primjera.

Nedostatak učenja temeljenog na primjerima [2] je količina računalnog vremena potrebnog za klasifikaciju. S obzirom da se radi o procesu izgradnje lokalne decizijske funkcije u dijelu prostora primjera, to može biti dugotrajno. Zato je važan problem u praktičnoj implementaciji razvoj metoda za čim efikasnije indeksiranje naučenih primjera u svrhu smanjenja količine proračuna u fazi klasifikacije. Drugi nedostatak, koji pogotovo vrijedi za k – NN metodu, jest što se pri klasifikaciji kod traženja sličnih primjera uvijek uzimaju u obzir svi raspoloživi atributi. Ako pak ciljni koncept ovisi o samo nekoliko atributa, s takvim načinom traženja stvarno slični primjeri mogu ispasti jako udaljeni.

Algoritam k – najbližih susjeda (k – nearest neighbor)

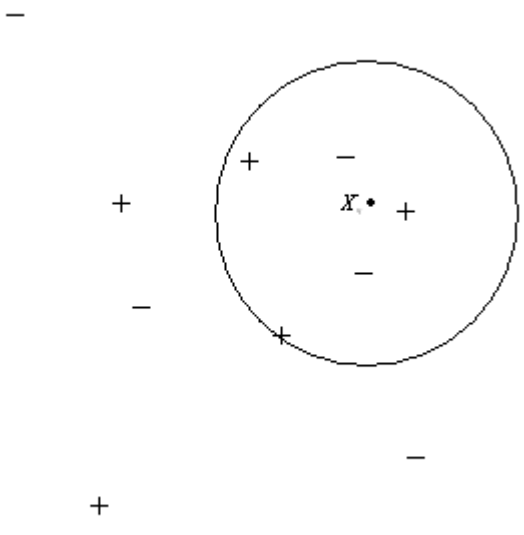
Najjednostavnija metoda iz područja učenja temeljenog na primjerima jest algoritam k – najbližih susjeda. Ovaj algoritam podrazumijeva da se svi primjeri mogu prikazati kao točke u n – dimenzionalnom prostoru $\Re^n$. Prema [2], najbliži susjedi nekog primjera se definiraju u kontekstu Euklidske udaljenosti. Primjer x se može prikazati vektorom atributa

$$\langle a_1(x), a_2(x), ..., a_n(x) \rangle$$

gdje $a_r(x)$ predstavlja vrijednost r – tog atributa primjera x. Tada se udaljenost između primjera x_i i x_j definira kao $d(x_i, x_j)$, gdje je

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} \left(a_r(x_i) - a_r(x_j)\right)^2}$$

U k – nn algoritmu učenje decizijske funkcije može se temeljiti na cjelobrojnim ili na decimalnim vrijednostima. Prvo slijedi opis funkcioniranja algoritma za cjelobrojne vrijednosti. Tu se decizijska funkcija definira kao $f : \Re^n \to V$, gdje je V konačan skup (kategorija za klasifikaciju) $\{v_1, v_2, ..., v_s\}$. Vrijednost koju vraća decizijska funkcija, *f(x)*, je zapravo pripadnost onoj grupi primjera kojoj pripada najviše susjeda klasificiranog primjera. Ako se za k odabere 1 (dakle uzima se u obzir samo jedan susjed), algoritam 1 – najbližeg susjeda će za vrijednost decizijske funkcije $f(x_q)$ odabrati vrijednost $f(x_i)$, gdje je x_i primjer iz skupa za učenje najbliži klasificiranom primjeru x_q. Za veće vrijednosti k, algoritam dodjeljuje klasifikaciju koja je najčešća među k najbližih susjeda.

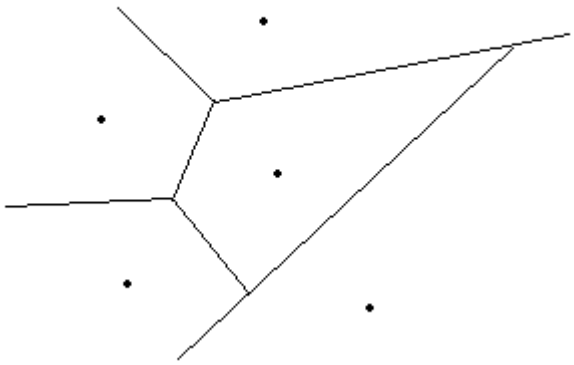


Slika 1. Prikaz rada 5-nn algoritma na dvodimenzionalnim podacima

Gornja slika pokazuje kako radi 5 – nn algoritam kada su podaci dvodimenzionalni i postoje dvije kategorije pripadnosti primjera (prikazane sa «+» i «-»). Sa točkom je prikazan novi primjer čija se klasifikacija treba

odrediti. Važno je primijetiti da će 1 – nn algoritam u danoj situaciji dati klasifikaciju «+», dok recimo 3 – nn daje klasifikaciju «-».

S obzirom da k – nn algoritam nikad ne stvara generalnu decizijsku funkciju za cijeli prostor primjera, moguće je zapitati se: kako bi takva funkcija mogla izgledati? Odgovor na to bi bio [2] rezultat ispitivanja u kojem bi se naučenim k – nn klasifikatorom klasificirao svaki mogući primjer iz prostora primjera. Rezultat za 1 – nn je prostor primjera podijeljen na poligone čije stranice određuju granice područja različite klasifikacije novog primjera.



Slika 2. Voronoi dijagram za 1-nn i dvodimnenzionalne podatke

To područje je ilustrirano slikom – točke predstavljaju primjere za učenje, a unutar svakog poligona postoji samo jedan primjer. Svaki od poligona određuje područje u kojem će se novi primjer klasificirati u kategoriju kojoj pripada primjer za učenje u tom poligonu. Taj dijagram se naziva često Voronoi – dijagramom.

Algoritam k – najbližih susjeda se lako može prilagoditi radu sa kontinuiranim vrijednostima. Potrebno je samo promijeniti način određivanja većine – umjesto brojanja koliko susjeda spada u koju kategoriju, računa se srednja vrijednost k najbližih susjeda. Tada se decizijska funkcija definira kao funkcija $f : \Re^n \to \Re$ čija se vrijednost određuje slijedećom formulom:

$$f(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

Algoritam k – najbližih susjeda sa težinskim faktorima

Jedno poboljšanje k – nn algoritma [2] koje se samo nameće je uvođenje težinskih faktora kod određivanja klasifikacije. Pri tome se bližim susjedima pridaje veća težina. Primjerice, svakog susjeda se može ponderirati težinom koja iznosi inverz kvadrata njegove udaljenosti od primjera za klasifikaciju. Tada se vrijednost decizijske funkcije određuje formulom:

$$f(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

gdje je

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

Da bi se zadovoljio slučaj kada se primjer za klasifikaciju poklapa sa primjerom za učenje - udaljenost $d(x_q, x_i)$ je jednaka nuli – primjeru za klasifikaciju se tada pridjeljuje kategorija kojoj pripada primjer za učenje s kojim se preklopio. Ako je takvih preklapajućih primjera za učenje više, dodjeljuje se klasifikacija većine njih.

Za primjenu k – nn sa kontinuiranim vrijednostima verzija decizijske funkcije glasi:

$$f(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

gdje je w_i definiran kao i za prethodni slučaj. Nazivnik u ovom slučaju služi kao normalizator težinskih faktora (koji npr. osigurava da ako vrijedi $f(x_i) = c$ za sve primjere za učenje da će tada klasifikacija novog primjera također biti $f(x_i) \leftarrow c$).

Važno je primijetiti da uvođenjem težinskih faktora nestaje potreba za ograničavanjem broja susjeda koji se uzimaju u obzir prilikom klasifikacije, jer su jako udaljeni primjeri za učenje ionako ponderirani malim težinama. Tu se naravno javlja teškoća tehničke prirode – uzimanjem u obzir svih primjera za

učenje značajno se povećava količina računanja i samim time količina računalnog vremena potrebnog za klasifikaciju. Ako se svi primjeri za učenje uzimaju u obzir kod klasifikacije, metoda se naziva globalnom, a ako se njihov broj ograničava, metoda se naziva lokalnom. Globalna metoda primijenjena na kontinuirane vrijednosti naziva se još i Shepardovom metodom.

Nedostaci algoritma k – najbližih susjeda

Algoritam k – najbližih susjeda sa težinskim faktorima je vrlo efikasna metoda induktivnog zaključivanja primjenjiva u mnogim područjima. Otporna je na šum u podacima [13] i može biti vrlo kvalitetan stroj za zaključivanje ako postoji dovoljna količina primjera za učenje.

Koja je induktivna pristranost k – nn algoritma sa težinskim faktorima? Prema [2], induktivna pristranost je sadržana u osnovnoj pretpostavci samog algoritma – da će klasifikacija novog primjera biti ista kao klasifikacija većine primjera koji su mu blizu po Euklidskoj udaljenosti.

Jedan praktični problem pri implementaciji [13] k – nn algoritama jest činjenica da se udaljenost među primjerima računa korištenjem svih postojećih atributa (tj. svih osi u Euklidskom prostoru primjera). To je u suprotnosti sa metodama kao što je metoda stabla odlučivanja, koja pri formiranju hipoteze uzima u obzir samo podskup atributa za koje se pokaže da su relevantni. Primjerice, može se uzeti u obzir primjena k – nn algoritma na slučaj u kojem je svaki primjer opisan sa 20 atributa, od kojih su samo 2 relevantna za klasifikaciju. U tom slučaju je moguće imati primjere koji imaju identične vrijednosti ta dva važna atributa, dok im ostalih 18 atributa imaju različite vrijednosti, i samim time se nalaze daleko u 20 – dimenzionalnom prostoru. Tu se pokazuje neprikladnost primjene k – nn algoritma, jer preostalih 18 atributa dominiraju u računu udaljenosti primjera, a potpuno su nevažni. Ovaj problem, koji uključuje veliki broj nevažnih atributa naziva se kletva dimenzionalnosti. K – nn algoritmi su posebno osjetljivi na taj problem.

Jedan pristup kojim se može umanjiti prethodni problem uvodi težinske faktore pomoću kojih se određuje važnost pojedinog atributa pri izračunavanju udaljenosti među primjerima. To odgovara rastezanju osi koje

odgovaraju važnijim atributima u Euklidskom prostoru primjera, a skupljanju osi koje odgovaraju nevažnim atributima. Iznos za koji se svaka os produžuje ili skraćuje može se automatski utvrditi iterativnom validacijom. Konkretno se može pretpostaviti da se os j množi sa težinskim faktorom z_j gdje su vrijednosti $z_1...,z_n$ izabrane tako da uzrokuju minimalnu pogrešku klasifikacije. Minimalna pogreška klasifikacije može se pak odrediti primjenom iterativne validacije. Iz skupa primjera za učenje nasumično se biraju primjeri koji se pridjeljuju u određeni podskup koji se koristi za učenje, a preostali primjeri se koriste za klasifikaciju. Pritom proces sam određuje težinske faktore da bi se minimizirala pogreška klasifikacije. Ponavljanjem opisane procedure nekoliko puta procjena težinskih faktora može biti točnija.

Drastičniji pristup rješavanju problema kletve dimenzionalnosti jest potpuno uklanjanje nevažnih atributa iz prostora primjera. Uklanjanje atributa se vrši modifikacijom postupka iterativne validacije – jednostavno se dopušta da težinski faktori postanu 0. Time se ti atributi ne uzimaju u obzir u računanju udaljenosti.

Još jedan praktičan problem je efikasno indeksiranje memorije. S obzirom da algoritam svo procesiranje vrši prilikom klasifikacije, potrebno je implementirati metode koje će omogućiti čim efikasnije pronalaženje susjeda. Razvijeno je više tehnika efikasnog indeksiranja spremljenih primjera za učenje nauštrb potroška memorije.

Primjena k – nn algoritma za klasifikaciju teksta

Prema [3], prvi korak je nalaženje prikladnog prikaza tekstualnih dokumenata (koji su zapravo niz znakova) u obliku pogodnom za primjenu k – nn algoritma. Najčešće korištena metoda za tu svrhu je takozvani model vektorskog prostora. U ovom modelu svaki se tekstualni dokument prikazuje kao vektor riječi. Za učenje se koristi matrica dokument – riječ koja predstavlja strukturu podataka u kojoj svaki član predstavlja broj pojavljivanja neke riječi u nekom dokumentu – npr. $A = (a_{ij})$, gdje je a_{ij} težina riječi i u dokumentu j. Postoji nekoliko načina određivanja težine a_{ij}. Neka je f_{ij} frekvencija riječi i u dokumentu j, N broj dokumenata u skupu za učenje, M

broj različitih riječi u skupu za učenje, a n_i ukupan broj puta pojavljivanja riječi i u cijelom skupu za učenje. Najjednostavniji pristup za određivanje težina su binarne težine, koje postavljaju a_{ij} na 1 ako se riječ pojavila u dokumentu, inače na 0. Još jedan jednostavan način koristi frekvenciju pojavljivanja riječi u dokumentu kao težinu, tj. $a_{ij} = f_{ij}$. Prema [3], [7], [8], [10] najpopularniji način određivanja težina je takozvana tf – idf metoda određivanja težina (od engl. *term frequency – inverse document frequency*) kod koje se težine računaju formulom:

$$a_{ij} = f_{ij} \times \log\left(\frac{N}{n_i}\right)$$

Ova se metoda može dopuniti ako se uzme u obzir da su tekstualni dokumenti imaju različite dužine, pa formula izgleda ovako:

$$a_{ij} = \frac{f_{ij}}{\sqrt{\sum_{i=1}^{M} f_{ij}^{\,2}}} \times \log\left(\frac{N}{n_i}\right)$$

Za matricu A (matricu dokument - riječ) broj redova je određen brojem različitih riječi u skupu dokumenata za učenje. S obzirom da može postojati stotine tisuća različitih riječi (sve riječi u nekom jeziku plus te riječi deklinirane, konjugirane itd.), potrebno je uvesti metode za smanjivanje dimenzionalnosti. Zato se obično uklanjaju stop – riječi (riječi koje se često pojavljuju a ne nose korisnu informaciju – npr. u hrvatskom je, se, i), zatim sufiksi riječi i neke naprednije metode, kao što su reparametrizacija i izlučivanje karakteristika.

Kod klasificiranja novog dokumenta k – nn algoritam određuje njegove susjede računanjem udaljenosti vektora dokumenata i na osnovu naziva razreda k najsličnijih susjeda određuje klasifikaciju novog dokumenta. Razredi se prilikom klasifikacije ponderiraju težinama koje su pak određene sličnošću susjeda klasificiranom dokumentu. Sličnost se određuje Euklidskom udaljenošću između vektora dokumenata ili kosinusnom

vrijednošću sličnosti između dva vektora dokumenata. Kosinusna sličnost se računa na slijedeći način:

$$sim(X, D_j) = \frac{\sum_{t_i \in (X \bigcap D_j)} x_i \times d_{ij}}{\|X\|_2 \times \|D_j\|_2}$$

gdje je X klasificirajući dokument prikaza vektorski. D_j je j – ti dokument iz skupa za učenje, t_i je riječ koja postoji i u X i u D_j, x_i je težina riječi t_i u dokumentu D_j, $\|X\|_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 + ...}$ je normala vektora X, a $\|D_j\|_2$ je normala vektora D_j - računa se analogno normali vektora X. Potrebno je odrediti prag za klasifikaciju dokumenta u neku kategoriju.

Metodologija ocjenjivanja učinkovitosti klasifikatora

Prema [4], za ocjenjivanje učinkovitosti klasifikatora koriste se standardne, poznate i usvojene metode koje za izračun učinkovitosti koriste veličine preciznost i odaziv. Prvo će biti pojašnjeno njihovo značenje i način njihovog određivanja, a zatim navedene metode za određivanje efikasnosti klasifikatora.

Preciznost i odaziv

Preciznost $\Pr_i$ se definira kao uvjetna vjerojatnost $p(ca_{ix} = 1 \mid a_{ix} = 1)$, tj. vjerojatnost da je klasifikacija nasumično odabranog dokumenta d_x u kategoriju c_i točna. Analogno tome, odaziv Re_i se definira kao uvjetna vjerojatnost $P(a_{ix} = 1 \mid ca_{ix} = 1)$, tj. vjerojatnost da se, ako dokument d_x stvarno pripada kategoriji c_i, odluka o toj klasifikaciji stvarno i dogodi. Navedene veličine su dakle vezane uz svaku kategoriju, dok je ih moguće odrediti i za čitav skup za učenje. Koristeći terminologiju iz logike, preciznost se može smatrati «stupnjem ispravnosti» skupa za učenje uz korišteni klasifikator, dok se na odaziv može gledati kao na «stupanj potpunosti» skupa za učenje uz korišteni klasifikator.

Prema njihovoj definiciji, veličine preciznost i odaziv se trebaju smatrati subjektivnim vrijednostima, koje npr. predstavljaju mjeru očekivanja korisnika da će sustav za klasifikaciju ispravno klasificirati nasumično odabrani dokument u kategoriju c_i. Ove se vrijednosti mogu procijeniti na temelju vrijednosti iz tablice odlučivanja [4], koja slijedi:

Kategorija c_i		Ispravna odluka	
		DA	NE
Odluka klasifikatora	DA	TP_i	FP_i
	NE	FN_i	TN_i

Tablica 2. Tablica odlučivanja za jednu kategoriju

U tablici FP_i (pogrešni pozitivni odgovori) predstavlja broj dokumenata koji su pogrešno klasificirani pod kategoriju c_i. TN_i (točni negativni odgovori), FN_i (pogrešni negativni odgovori) i TP_i (točni pozitivni odgovori) su definirani analogno. Iz navedenih vrijednosti preciznost i odaziv se računaju kao:

$$\mathrm{Pr}_i \cong \frac{TP_i}{TP_i + FP_i}$$

$$\mathrm{Re}_i \cong \frac{TP_i}{TP_i + FN_i}$$

Za izračun procjene preciznosti i odaziva za cijeli skup za učenje moguće je korištenje dva pristupa:

- mikroprosjek: preciznost i odaziv se dobivaju globalnim sumiranjem po svim odlukama klasifikacije, tj.:

$$\mathrm{Pr}^{\mu} \cong \frac{TP}{TP + FP}$$

$$\mathrm{Re}^{\mu} \cong \frac{TP}{TP + FN}$$

gdje se vrijednosti potrebne za izračun dobivaju na slijedeći način (globalna tablica odlučivanja [4]):

Skup svih kategorija $C = \{c_1, \ldots, c_n\}$		Ispravna odluka	
		DA	NE
Odluka klasifikatora	DA	$TP = \sum_{i=1}^{m} TP_i$	$FP = \sum_{i=1}^{m} FP_i$
	NE	$FN = \sum_{i=1}^{m} FN_i$	$TN = \sum_{i=1}^{m} TN_i$

Tablica 3. Tablica odlučivanja za sve kategorije

- makroprosjek: preciznost i odaziv se prvo računaju «lokalno», za svaku kategoriju posebno, a zatim se računaju globalne vrijednosti iz lokalnih:

$$\mathrm{Pr}^{M} \cong \frac{\sum_{i=1}^{m} \mathrm{Pr}_i}{m}$$

$$\mathrm{Re}^{M} \cong \frac{\sum_{i=1}^{m} \mathrm{Re}_i}{m}$$

Važno je primijetiti da ove dvije metode mogu dati bitno različite rezultate, pogotovo ako kategorije imaju nejednak broj dokumenata. Ako npr. klasifikator radi sa skupom za učenje koji po kategoriji ima mali broj pozitivnih primjera, njegova će učinkovitost biti bolja ako se koristi makroprosjek nego ako se koristi mikroprosjek.

Metode određivanja učinkovitosti

U praksi se pokazalo da se kod svakog klasifikatora može mijenjanjem parametara povećati preciznost na štetu odaziva i obratno. Zato je potrebno koristiti «kombinirane» metode za ocjenjivanje učinkovitosti, koje kombiniraju preciznost i odaziv pri određivanju učinkovitosti. Najčešće kombinirane metode su:

- računanje učinkovitosti pomoću (interpoliranog) prosjeka preciznosti u 11 točaka. Parametrima se ugađaju vrijednosti odaziva 0, 0.1, 0.2, ..., 1. Pri svakom od tih 11 ugađanja očitava se vrijednost preciznosti, i na kraju se računa prosjek preciznosti od tih 11 izmjerenih vrijednosti.
- učinkovitost se računa kao točka izjednačavanja, a misli se na točku u kojoj je preciznost jednaka odazivu.
- učinkovitost se računa kao vrijednost funkcije F_α, za neki $0 \le \alpha \le 1$:

$$F_\alpha = \frac{1}{\alpha \frac{1}{\mathrm{Pr}} + (1-\alpha)\frac{1}{\mathrm{Re}}}$$

U ovoj formuli se α može smatrati relativnim stupnjem važnosti koja se pridaje preciznosti i odazivu: ako je α=1, tada je F_α jednaka Pr, a ako je α=0, F_α odgovara Re. Uobičajeno je za vrijednost α uzeti 0.5, kada se funkcija ne naziva $F_{0.5}$, već F_1. Za svaki klasifikator vrijedi da je točka izjednačavanja uvijek manja ili jednaka njegovoj F_1 vrijednosti.

Kada se izabere metoda mjerenja učinkovitosti provodi se ugađanje parametara kako bi se maksimizirala učinkovitost klasifikatora.

3. OPIS APLIKACIJE

Opis i obrada ulaznih podataka

Reuters skup za učenje

Reuters skup korišten za učenje i klasifikaciju je Reuters 21450 ApteMod [5]. To je jedan od najpopularnijih Reuters skupova za učenje, a naziv mu proizlazi iz originalne brojke članaka u skupu, dakle 21450 članaka. Međutim, ovdje se radi o ApteMod verziji, koja je izmijenjena u odnosu na original. Naime, u ApteMod verziji izbačeni su članci koji nemaju kategorizaciju, što odgovara potrebama ovog rada. Nakon izbacivanja članaka bez kategorije, ApteMod verzija sadrži ukupno 11099 članaka. U izvornom obliku je dostupna u dvije datoteke, koje sadrže skup za učenje i skup za testiranje. Pritom skup za učenje sadrži 7790 članaka, a skup za testiranje 3309 članaka. Sam format članaka je strogo određen, i to na slijedeći način:

```
.I broj_clanka
.C
ime_kategorije;ime_kategorije
.T
naslov_clanka
.W
tekst_clanka
```

Dodatno pojašnjenje je potrebno samo za dio koji opisuje kategoriju. Naime, u Reuters kolekciji postoje članci koji spadaju u više od jedne kategorije, tako da red koji opisuje kategorije može imati proizvoljan broj kategorija, pri čemu je minimalan broj kategorija 1. U slučaju više kategorija, sve su kategorije međusobno odvojene znakom «;» - iza posljednje kategorije ne dolazi znak «;».

Medline skup za učenje

Medline skup korišten u ovom radu je skinut sa Interneta [6], a sastoji se od tri datoteke – MED.ALL, MED.REL, i MED.QRY. Datoteka MED.ALL sadrži svih 1033 članaka sa formatom zapisa dosta sličnim Reuters 21450 ApteMod:

```
.I broj_clanka
.W
tekst_clanka
```

Vidljiv je nedostatak kategorije. Popis članaka i njihove pripadnosti kategorijama se nalazi u datoteci MED.REL. Unutar tog popisa se ne nalaze svi članci, iz čega proizlazi da neki članci neće imati kategoriju. Rješenje tog i sličnih problema je opisano u slijedećem odjeljku.

Obrada ulaznih podataka

Iz prethodnih opisa skupova za učenje je vidljivo da imaju neke zajedničke karakteristike što se tiče formata zapisa članaka – zapis broja članka i teksta članka su identični. Međutim, u Medline bazi članaka nedostaje zapis kategorije, što je vrlo bitno, a u Reuters bazi su naslovi članaka izdvojeni iz tijela članka. Zato je potrebno odrediti univerzalni format zapisa članaka kojim će se i Reuters i Medline baze prilagoditi za parsiranje istim parserom. Univerzalni format zapisa članka izgleda ovako:

```
.I broj_clanka
.C
ime_kategorije;ime_kategorije;
.W
tekst_clanka
```

Univerzalni format zapisa ima zajednička obilježja oba skupa za učenje, s nekoliko razlika:

- u odnosu na Reuters skup – naslov članka je pridružen tijelu članka, a popis kategorija sada završava sa «;»
- u odnosu na Medline skup – dodan je dio sa kategorijom članka

Još je potrebno napomenuti da u redu sa popisom kategorija u slučaju postojanja jedne kategorije naziv ne završava znakom «;».

Osim utvrđivanja univerzalnog formata zapisa potrebno je i riješiti problem sa nedostatkom kategorija za neke članke u Medline bazi članaka. Kao rješenje problema stvaraju se dvije podvrste Medline baze:

- Medline – u kojoj su članci bez kategorije svrstani u novu kategoriju (tzv. «Kategoriju 31» - jer originalnih kategorija ima 30)
- Medline K31 – u kojoj su članci koji pripadaju «Kategoriji 31» izbačeni

Razlog za stvaranje dvije podvrste je slijedeći: ako se članci bez kategorije izbace (Medline K31), broj preostalih članaka se znatno smanjuje (na 696 članaka – 472 za učenje i 224 za testiranje), pa postaje upitna upotrebljivost takvog skupa za učenje. S druge strane, uvođenje nove kategorije u kojoj su svi članci bez kategorije uvodi velik nesrazmjer u distribuciju dokumenata po kategorijama. Koji je od navedena dva pristupa bolji, pokazat će testiranja klasifikatora.

Posljednji problem koji je potrebno riješiti jest razdvajanje Medline skupa na članke za učenje i članke za testiranje. Po uvriježenoj *a priori* pretpostavci distribucije dokumenata po kategorijama su iste u skupu za učenje i u skupu za testiranje. Zato su u skladu s tom pretpostavkom razdijeljeni Medline i Medline K31 na skupove za učenje i skupove za testiranje u slijedećim odnosima:

- Medline: skup za učenje 708 članaka (68.5% od ukupnog broja), skup za testiranje 325 članaka (31.5% od ukupnog broja)
- Medline K31: skup za učenje 472 članka (67.8% od ukupnog broja), skup za testiranje 224 članka (32.2% od ukupnog broja)

Odnos broja članaka je određen i približan je odnosu skupova za učenje i za testiranje iz Reuters kolekcije (70.1% skup za učenje, 29.9% skup za testiranje).

Sve navedene preinake su provedene programski, sa za specijalno tu svrhu napisanim programima. Ti se programi ovdje neće opisivati jer se ne tiču samog rada, već služe samo kao pomoćno oruđe za obradu ulaznih podataka.

Pretprocesiranje

Nakon osiguravanja potpuno određenih i strogo formatiranih zapisa članaka moguće je vršiti pretprocesiranje. Proces pretprocesiranja služi kao spona između ulaznih podataka i samog algoritma za klasifikaciju, a tijekom pretprocesiranja vrši se parsiranje dokumenta, koje procesu za učenje daje riječi iz dokumenta koje su zadovoljile postavke procesiranja (kao što su minimalna dužina riječi, izbacivanje stop riječi, itd.) Proces pretprocesiranja je najjednostavnije prikazati pseudokodom:

```
Proces pretprocesiranja

input := procitaj_liniju;
radi_zauvijek
{
     ako(u_input_postoji(«.W»))
     {
          input := procitaj_liniju;
          radi_dok_vrijedi(u_input_ne_postoji(«.I »))
          {
               procesiraj_string(input);
               input := procitaj_liniju;
               ako(kraj_datoteke)
                    zavrsi_proces;
          }
     }
     ako(u_input_postoji(«.I »))
     {
          input := procitaj_liniju;
          input := procitaj_liniju;
          ako(input_sadrzi_vise_kategorija)
          {
               dodaj_u_ucenje_kat(sve_kategorije);
               postavi_zastavicu(multikategorija);
          }
          inace
          {
               dodaj_u_ucenje_kat(kategorija);
          }
          input := procitaj_liniju;
     }
}
```

Navedeni pseudokod koristi funkciju procesiraj_string koja je zadužena za obradu svake linije teksta. Isto tako, ta funkcija koristi funkciju procesiraj_rijec

koja je pak zadužena za obradu svake pojedine riječi. Zato će sada biti prikazane u pseudokodu jedna ispod druge:

```
procesiraj_string(input)
{
     odbaci_prazne_znakove_na_krajevima(input);
     pretvori_u_mala_slova(input);
     ako(input_nije_prazan i input_duzi_od_1)
     {
          radi_dok_vrijedi(input_nije_prazan)
          {
               rijec := izvadi_rijec_iz_inputa;
               procesiraj_rijec(rijec);
          }
     }
}

procesiraj_rijec(rijec)
{
     izbaci_znakove_koji_nisu_slova_ni_brojke_sa_kraj
     eva(rijec);
     ako_rijec_nije_u_stop_rijecima
     {
          ako(postavljena_multikategorija_zastavica)
          {
          radi_za_svaku_pripadnu_kategoriju
          {
          dodaj_u_ucenje(rijec, tekuca_kategorija);
          }
          }
          inace
          {
          dodaj_u_ucenje(rijec, pripad_kategorija);
          }
     }
}
```

Pseudokodom je opisan proces pretprocesiranja sve do trenutka kada se procesirana riječ predaje u proces učenja. Tijekom te obrade se izlučuju riječi iz teksta i kategorije članaka. Ovisno o broju kategorija procesiranje riječi je drukčije. Time se postiže da se članci koji imaju više kategorija interpretiraju kao članci koji imaju isti tekst, a različitu kategoriju (koliko kategorija – toliko članaka). Isti se proces koristi i za klasifikaciju, doduše sa nekim minornim izmjenama – riječ se šalje na klasifikaciju, a ne na učenje i ne obrađuju se posebno članci sa više kategorija.

Integralni dio procesa pretprocesiranja je izlučivanje značajki (engl. *feature selection*). Izlučivanje značajki omogućuje povećanje kvalitete ulaznih podataka koji se koriste za učenje. Postojanje premalo riječi koje se koriste za učenje može dovesti do nemogućnosti izgradnje hipoteze za klasifikaciju, dok previše riječi izaziva šum u podacima. Prema [7], tri su koraka u izlučivanju karakteristika:

1. Izbacivanje riječi koje se rijetko pojavljuju
2. Izbacivanje riječi koje se često pojavljuju
3. Izabiranje riječi koje pružaju najviše informacije za učenje ciljne hipoteze

Od navedenih se u ovom radu koriste prve dvije. Riječi koje se često pojavljuju se izbacuju upravo u procesu pretprocesiranja, unutar funkcije procesiraj_rijec. Najčešće riječi su određene listom najčešćih engleskih riječi koja se učitava iz posebne tekstualne datoteke. Za izbacivanje riječi koje se rijetko pojavljuju potrebno je završiti parsiranje, tako da se taj dio izlučivanja karakteristika obavlja na kraju procesa učenja.

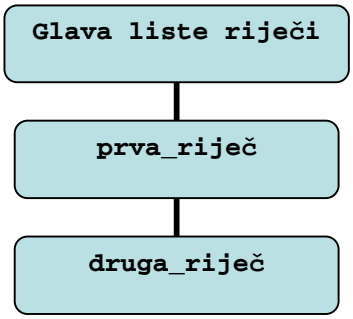
Također je potrebno odrediti da li se tijekom obrade riječi trebaju cijepati na čiste riječi ili se mogu koristiti složenice (tipa «state-of-the-art»). Isto tako, postavlja se pitanje treba li izbaciti i brojeve iz skupa riječi koje se koriste u učenju. Na ova pitanja odgovor se može naći u [1], a odgovor glasi: ovisi o primjeni. Za primjenu u ovom radu se čini korisnijim dozvoliti složenice (pogotovo zbog Medline baze, u kojoj su složenice kao «x-ray-irradiated» uobičajene). Također ima smisla koristiti i brojeve, jer u Reuters kolekciji članaka prevladavaju članci koji se tiču financija, a i Medline kolekcija ne oskudijeva brojevima.

Izvedba naivnog Bayesovog klasifikatora

Realizacija struktura podataka

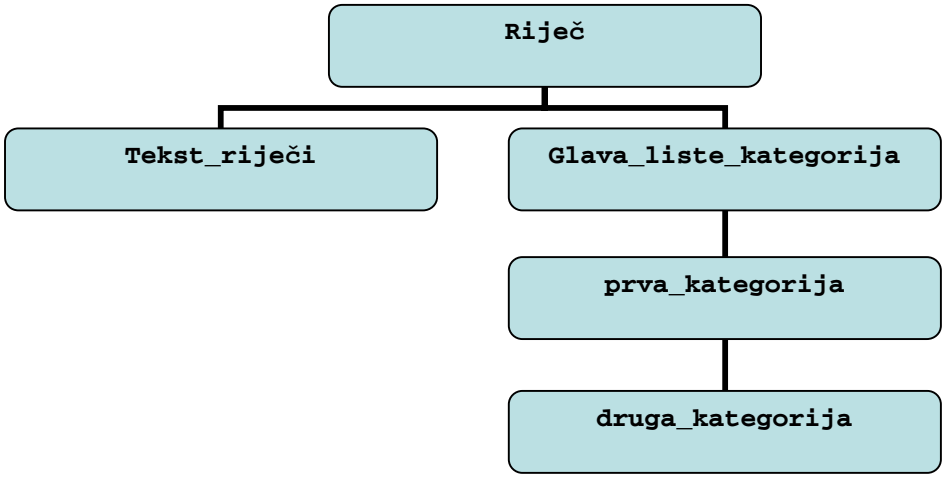
Naivni Bayesov klasifikator gradi strukturu podataka u kojoj se nalaze dvije liste: lista različitih riječi i lista kategorija.

Lista riječi sadrži sve različite riječi dobivene iz procesa pretprocesiranja. Svaka riječ u sebi sadrži i listu kategorija u kojima se pojavila (ta lista kategorija nije jednaka onoj listi kategorija koja se spominje na početku!). Zato je ova lista riječi zapravo reducirani oblik matrice riječ – kategorija kakva se obično rabi za ovakve primjene. Struktura liste riječi se najbolje može opisati slikom:



Slika 3. Struktura liste riječi

Također, svaka riječ u sebi sadrži listu kategorija u kojoj se pojavila:



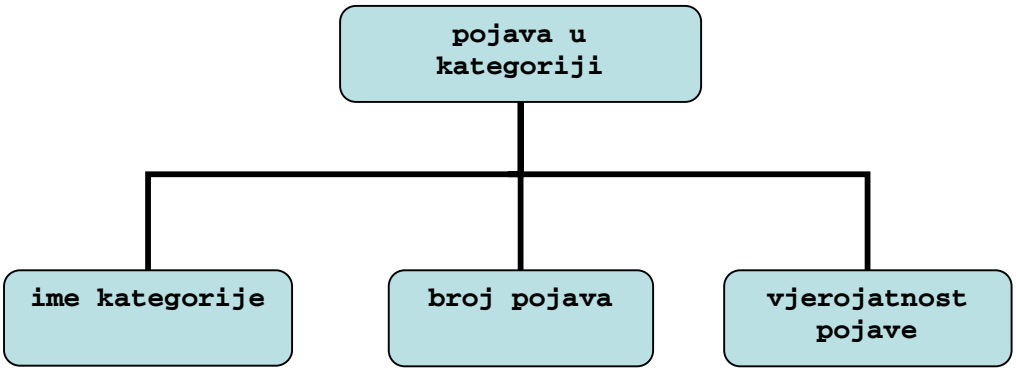
Slika 4. Struktura riječi

Uvrštavanjem u listu kategorija riječi samo onih kategorija u kojima se riječ pojavila postiže se ušteda na prostoru koji ova struktura zauzima u memoriji,

dok se funkcionalno ništa nije izmijenilo – ako se tijekom klasifikacije zatraži vjerojatnost pojave riječi u kategoriji za koju nema zapisa, vratit će se kao vrijednost vjerojatnosti 0. Tu vrijednost dakako klasifikator neće interpretirati kao vrijednost vjerojatnosti nula, već će umjesto nule u umnožak uvrstiti vrijednost koja se dobije kada se u formulu za izračun vjerojatnosti pojave [2] neke riječi uvrsti nula:

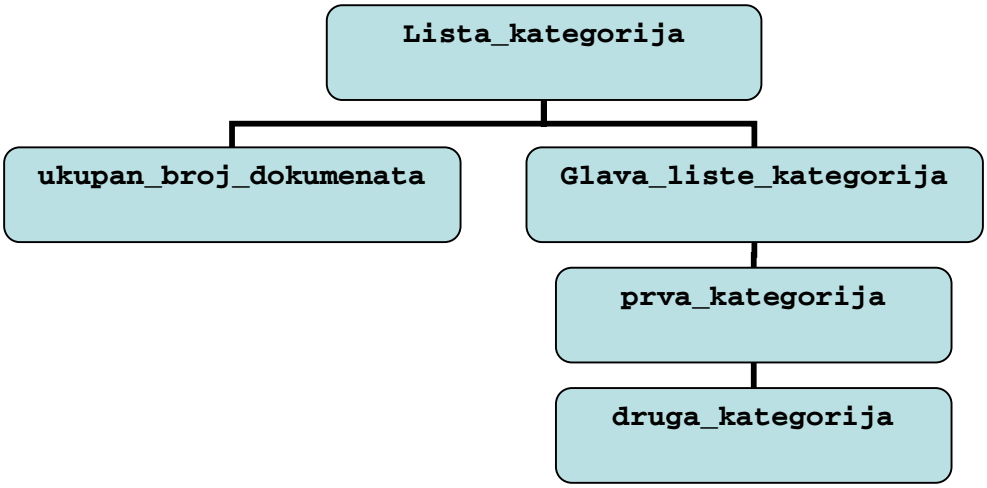
$$P(w_k \mid v_j) = \frac{1}{n + |Rijecnik|}$$

Pritom je n broj različitih riječi za kategoriju za koju se vrijednost računa, a $Rijecnik$ je broj različitih riječi koje su se pojavile tijekom učenja – odnosno broj članova liste riječi. Ta izračunata vrijednost je nazvana defaultna vrijednost. Zapis kategorije unutar liste kategorija koju sadrži svaka riječ izgleda ovako:



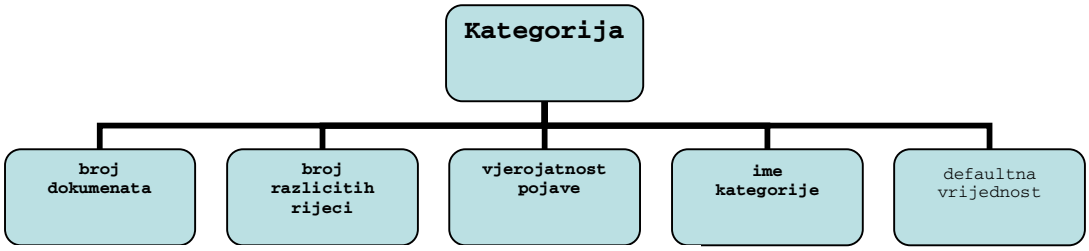
Slika 5. Struktura zapisa iz liste kategorija

Drugi dio strukture podataka je lista kategorija. Ta lista sadrži popis različitih kategorija koje su se pojavile tijekom učenja. Sadržaj liste nije predodređen, već se generira tijekom procesa učenja – time se postiže prilagodljivost klasifikatora za bilo koji skup podataka za učenje (pod uvjetom da zadovoljava format članka ili se parser prilagodi novom formatu). Sama lista kategorija je organizirana na slijedeći način:



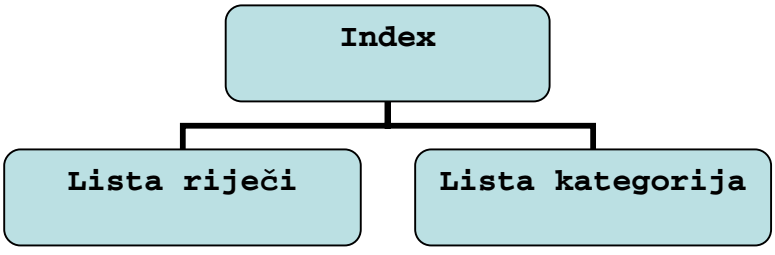
Slika 6. Struktura liste kategorija

Zapisi kategorija u ovoj listi nisu jednaki zapisu kategorija u listi kategorija sadržanoj u riječi. Zapis kategorije ovdje se sastoji od podataka relevantnih za potrebne proračune za svaku od kategorija, a izgleda ovako:



Slika 7. Struktura zapisa iz liste kategorija

Dakle, konačni izgled podatkovne strukture je slijedeći (podatkovna struktura je nazvana index):



Slika 8. Podatkovna struktura Index

Algoritam za učenje

Algoritam za učenje je izveden po uzoru na algoritam za učenje primijenjen za klasifikaciju teksta u [2]. Proces učenja je u izvedbi tijesno povezan sa procesom pretprocesiranja i zapravo se izvodi paralelno s njime. Pretprocesor izlučuje riječ po riječ iz skupa za učenje i šalje ih u proces za učenje. Osim riječi iz teksta, iz pretprocesiranja se šalju i kategorije članaka. Proces učenja je zadužen za manipulaciju tim primljenim podacima i njihovim spremanjem na odgovarajuća mjesta u Index strukturi podataka. Također se na kraju procesa učenja vrši i finalna faza izlučivanja karakteristika – odbacivanje riječi koje se rijetko pojavljuju. Procesom učenja zapravo upravlja pretprocesiranje, jer se

funkcije koje čine proces učenja pozivaju iz pretprocesiranja – zato će opis učenja u pseudokodu sadržavati opise funkcija učenja koje se pozivaju u pretprocesiranju:

```
Proces učenja

dodaj_u_ucenje(rijec, kategorija)
{
 radi_dok(ima_clanova_liste_rijeci)
 {
  clan_liste := uzmi_slijedeci_clan_liste_rijeci;
  ako(clan_liste = rijec)
  {
    lista_kategorija := dohvati_listu_kat(clan_liste);
    radi_dok(ima_clanova_liste_kategorija)
    {
      clan_kat := uzmi_slij_clan_liste_kategorija;
      ako(clan_kat = kategorija)
      {
       povecaj_broj_pojava(clan_kat);
      }
    }
    ako(nije_nadjen(clan_kat = kategorija))
    {
     clan_kat := stvori_novi_clan_kat(kategorija);
     dodaj_u_listu(clan_kat, lista_kategorija);
    }
  }
 }}
```

```
dodaj_u_ucenje_kat(kategorija)
{
  radi_dok(ima_clanova_liste_kategorija)
  {
  clan_liste_kat := uzmi_slijed_clan_liste_kategorija;
  ako(clan_liste_kat := kategorija)
  {
    povecaj_broj_dokumenata(clan_liste_kat);
  }
  }
  ako(nije_nadjen_clan_liste_kat)
  {
     clan_liste_kat :=
     stvori_novi_clan_liste_kat(kategorija);
     dodaj_u_listu_kategorija(clan_liste_kat);
  }
```

Nakon što je završilo pretprocesiranje (koje je pozivalo gornje dvije funkcije), potrebno je dovršiti proces učenja još nekim proračunima. Naime, neposredno nakon procesiranja je lista riječi popunjena, a popunjena je i lista kategorija (dakle čitava Index struktura je popunjena). No, da bi učenje završilo, potrebno je izvesti još slijedeće korake:

1. Izbaciti riječi koje se rijetko pojavljuju
2. Prebrojiti različite riječi u svakoj kategoriji
3. Izračunati vjerojatnosti pojave svake od kategorija
4. Izračunati vjerojatnosti pojave svake riječi za svaku kategoriju u kojoj se pojavila

Prvi korak je zapravo zadnji korak izlučivanja karakteristika (engl. *feature exctraction*) koje povećava kvalitetu ulaznih podataka. Pseudokod prikazuje općeniti oblik funkcije za izbacivanje riječi, koji kao parametar prima ukupni broj pojava neke riječi do kojeg se ta riječ izbacuje:

```
ukloni_rijeci_sa_manje_pojava_od(broj_pojava)
{
 radi_dok(ima_clanova_liste_rijeci)
 {
  rijec := uzmi_slijedeci_clan_liste_rijeci;
  broj := zbroji_sve_pojave_rijeci(rijec);
  ako(broj < broj_pojava)
  {
   ukloni_rijec_iz_liste_rijeci(rijec);
  }}}
```

Zbog formule za računanje vjerojatnosti pojava riječi u pojedinoj kategoriji potrebno je odrediti broj različitih riječi u svakoj kategoriji. To radi slijedeća funkcija:

```
izbroji_broj_razlicitih_rijeci()
{
  radi_dok(ima_clanova_liste_rijeci)
  {
   rijec := uzmi_slijedeci_clan_liste_rijeci;
   lista_kat := dohvati_listu_kategorija(rijec);
   radi_dok(ima_clanova_liste_kat)
   {
   clan_kat := uzmi_slijedeci_clan_liste_kat;
   pov_broj_raz_rijeci_u_listi_kategorija(clan_kat);
   }
  }
}
```

Važno je primijetiti da funkcija pov_broj_raz_rijeci_u_listi_kategorija radi sa listom kategorija sadržanom u strukturi Index, a ne sa listom sadržanom u svakom članu liste riječi.

Sada su ispunjeni svi preduvjeti za izračunavanje najbitnijih veličina – vjerojatnosti pojava kategorija i riječi. Računanje vjerojatnosti pojava po kategorijama je jednostavno, zahtijeva jednu iteraciju kroz listu kategorija u Index strukturi, a izvodi se ovako:

```
izracunaj_vjerojatnosti_kategorija()
{
     radi_dok(ima_clanova_liste_kategorija)
     {
     clan_kat := uzmi_slijedeci_clan_liste_kategorija;
     vjerojatnost_pojave(clan_kat) :=
     broj_dokumenata(clan_kat) / ukupan_broj_dok;
     }
}
```

Računanje vjerojatnosti pojava za svaku riječ u svakoj kategoriji u kojoj se pojavila zahtijeva jednu iteraciju kroz listu riječi, a za svaku riječ je potrebno iteracija onoliko koliko ima zapisa pojava u kategoriji. Dodatno, za računanje vjerojatnosti svakog zapisa potrebno je za tu kategoriju dohvatiti broj različitih riječi iz liste kategorija, što iziskuje dodatne iteracije kroz listu kategorija. Pseudokod slijedi:

```
izracunaj_vjerojatnosti_rijeci()
{
 radi_dok(ima_clanova_liste_rijeci)
 {
  rijec := uzmi_slijedeci_clan_liste_rijeci;
  lista_kat := dohvati_listu_kat(rijec);
  radi_dok(ima_clanova_liste_kat)
  {
   clan_kat := uzmi_slijedeci_clan_liste_kat;
   vjerojatnost_pojave(clan_kat) :=
   (1 + broj_pojava(clan_kat))/
   (dohvati_broj_razl_rij(kategorija(clan_kat)) +
   ukupan_broj_razlicitih_rijeci);
  }
 }
}
```

Izračunom vjerojatnosti riječi završava proces učenja. Rezultat procesa učenja je napunjena i obrađena Index struktura koja je sada spremna za proces klasifikacije.

Algoritam za klasifikaciju

Proces klasifikacije je načelno sličan procesu učenja – ponovno se provodi pretprocesiranje podataka, samo što ovaj put proces pretprocesiranja ne poziva funkcije procesa učenja, već procesa klasifikacije. Funkcije klasifikacije koriste naučene podatke iz strukture Index da bi na temelju riječi koje šalje pretprocesor odredile kategoriju dokumenta koji se klasificira. Formula za klasifikaciju naivnog Bayesovog klasifikatora iz [2] glasi:

$$v_{NB} = \underset{v_j \in [lista\_kat]}{\arg\max} P(v_j) \prod_{i=1}^{n} P(a_i \mid v_j)$$

Pritom n predstavlja broj riječi u klasificiranom dokumentu. S obzirom da treba stvoriti umnožak za svaku od kategorija posebno, najjednostavnije je iskoristiti postojeću strukturu podataka – listu kategorija (unutar strukture Index). Svaki zapis u listi kategorija sadrži dodatan podatak u koji se sprema umnožak za svaku od kategorija. Funkcije klasifikacije će koristiti te podatkovne članove za računanje umnoška, i na kraju klasifikacije svakog dokumenta odatle odrediti najveći umnožak. Slijedi opis funkcija klasifikacije u pseudokodu:

```
resetiraj_za_klasifikaciju()
{
  radi_dok(ima_clanova_liste_kategorija)
  {
   clan_kat := uzmi_slijedeci_clan_liste_kategorija;
   vjerojatnost_klasifikacije(clan_kat) :=
   vjerojatnost_pojave(clan_kat);
  }
}

dodaj_rijec_u_produkt(rijec)
{
 index_rijec := nadji_rijec_u_listi_rijeci(rijec);
 ako(index_rijec_nije 0)
 {
  radi_dok(ima_clanova_liste_kategorija)
  {
   clan_kat := uzmi_slijedeci_clan_liste_kategorija;
   vjerojatnost :=
   dohvati_vjerojatnost_pojave(index_rijec, clan_kat);
  ako(vjerojatnost > 0)
  {
   vjerojatnost_klasifikacije(clan_kat) :=
   vjerojatnost_klasifikacije(clan_kat) * vjerojatnost;
  }
  inace
  {
   vjerojatnost_klasifikacije(clan_kat) :=
   vjerojatnost_klasifikacije(clan_kat) *
   defaultna_vrijednost(clan_kat);
  }
 }
}

odredi_klasifikaciju()
{
 max := 0;
 ime_max := «»;
 radi_dok(ima_clanova_liste_kategorija)
 {
  clan_kat := uzmi_slijedeci_clan_liste_kategorija;
  ako(vjerojatnost_klasifikacije(clan_kat) >= max)
  {
   ako(vjerojatnost_klasifikacije(clan_kat) = max)
   {
   ime_max := ime_max + «;» + ime_kategorije(clan_kat);
   }
```

Pseudokod opisi funkcija zahtijevaju nekoliko dodatnih pojašnjenja. Funkcija

```
   inace
   {
     ime_max := ime_kategorije(clan_kat);
     max := vjerojatnost_klasifikacije(clan_kat);
   }
  }
 }
 vrati ime_max;
 }
```

resetiraj_za_klasifikaciju se poziva iz pretprocesiranja svaki put kada se započinje pretprocesiranje novog dokumenta za klasifikaciju. Time se svi podatkovni članovi u koje se sprema umnožak postavljaju na početnu vrijednost, koja je jednaka vjerojatnosti pojave kategorije kojoj taj podatkovni član pripada. To odgovara dijelu formule za klasifikaciju prije znaka produkta. Funkcija dodaj_rijec_u_produkt prima svaku novu riječ iz pretprocesiranja i dodaje njene vjerojatnosti pojave u odgovarajuće produkte. Za one kategorije za koje nema pojave te riječi produkt se množi sa veličinom defaultna vrijednost, koja je opisana u odjeljku o strukturama podataka. Kada pretprocesiranje dođe parsiranjem do kraja tekućeg dokumenta, poziva se funkcija za određivanje klasifikacije – odredi_klasifikaciju. Ta funkcija jednostavno iterira kroz listu kategorija i traži kategoriju za koju je produkt najveći. Na kraju vraća ime kategorije za koju je produkt bio najveći.

Optimizacija brzine izvođenja algoritama za učenje i za klasifikaciju

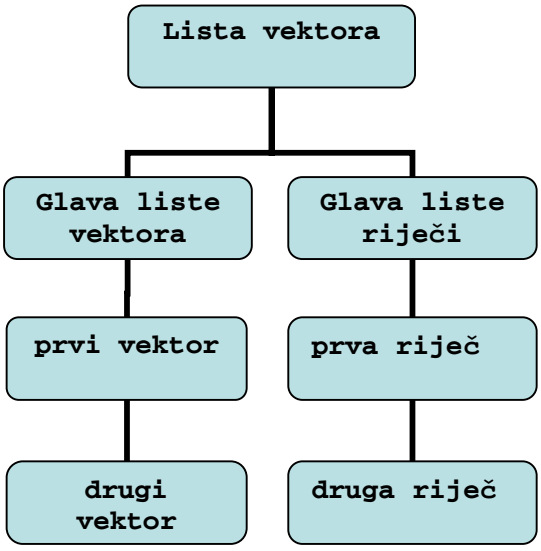
Prethodno opisani algoritmi rade ispravno, ali u svojoj originalnoj izvedbi rade dosta sporo – za najveći skup za učenje, Reuters, proces učenja traje nekoliko sati. Postavlja se pitanje – može li se ubrzati vrijeme izvođenja? Da bi se moglo odgovoriti na to pitanje, potrebno je utvrditi koje operacije algoritam najčešće izvodi. Nije potrebno mnogo da bi se utvrdilo da se radi o operacijama dohvata i umetanja riječi u listu riječi. Naime, broj različitih riječi (čak i nakon izbacivanja najčešćih i najrjeđih riječi) je značajan, za najmanji skup za učenje (Medline K31) iznosi oko 2000 riječi. U originalnoj izvedbi lista riječi je implementirana kao obična dvostruko povezana lista. S obzirom da je u najgorem slučaju za dohvat iz liste potrebno n iteracija, u

originalnoj implementaciji se dakle radi o 2000 iteracija za dohvat traženog člana liste. Jasno je da nije niti potrebno razmatrati vrijeme dohvata za Reuters skup za učenje (oko 11000 riječi!). Dakle, potrebno je skratiti vrijeme dohvata i umetanja u listu riječi. Rješenje primijenjeno je poprilično jednostavno: primjenjuje se tehnika hashiranja. Princip je slijedeći: lista se razdijeli na 28 manjih listi koje se razlikuju po svom sadržaju: svaka lista sadrži riječi koje počinju drukčijim slovom engleske abecede, a još dvije liste služe za spremanje brojeva i ostalih riječi koje (možebitno) počinju sa znakom koji nije niti broj, niti slovo. Ovim jednostavnim pristupom se (u prosjeku) broj iteracija potrebnih za dohvat i umetanje smanjuje na n/28, što je značajno poboljšanje. U praksi je opisano poboljšanje pokazalo izvrsne rezultate – trajanje procesa učenja za Reuters je smanjeno sa nekoliko sati na oko 5 minuta! Trajanje procesa klasifikacije je također skraćeno, na vrijeme kraće od procesa učenja (dakle kraće od 5 minuta).

Izvedba k-nn klasifikatora

Realizacija struktura podataka

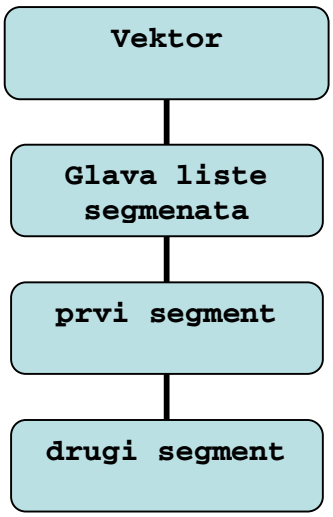
Za razliku od naivnog Bayesovog klasifikatora, k – nn klasifikator nema tako složenu strukturu podataka. Glavna struktura podataka je nazvana Lista vektora. Za predstavljanje dokumenata kao vektora se ne koristi uobičajena implementacija potpunog vektorskog prostora, već su dokumenti prikazani kao rijetki vektori – svaki je vektor dugačak onoliko koliko je dugačak dokument iz kojeg je taj vektor dobiven. Takvim pristupom se značajno štedi na zauzeću memorije, a ubrzava se i računanje udaljenosti među dokumentima. Algoritmi za računanje udaljenosti među vektorima su prilagođeni ovakvom prikazu. Spomenuta Lista vektora se sastoji od dva dijela: liste vektora koja je sačinjena od vektora dokumenata za učenje, i liste različitih riječi, koja sadrži popis različitih riječi. Popis različitih riječi služi za računanje TFIDF težina, o kojima će biti govora kasnije. Izgled glavne strukture podataka:



Slika 9. Struktura liste vektora

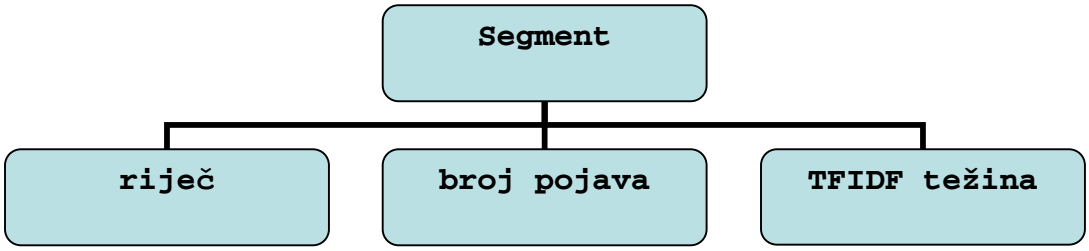
Vektori koji su sadržani unutar liste sadrže u sebi listu segmenata. Svaki se segment sastoji od nekoliko podataka: riječ kojoj odgovara taj segment, broj pojave te riječi u dokumentu kojeg predstavlja vektor čiji je segment dio, i

TFIDF težina segmenta – koja se koristi za računanje udaljenosti. Slijede shematski prikazi vektora i segmenta vektora:



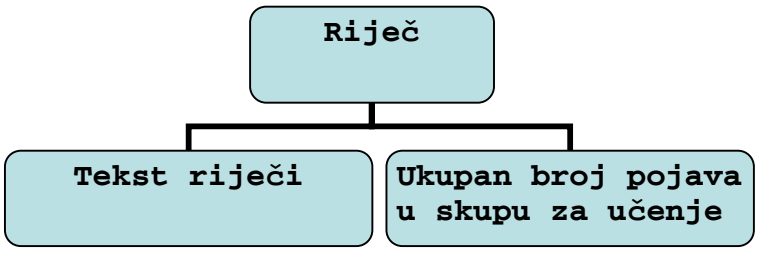
Slika 10. Struktura vektora

Organizacija pojedinog segmenta:



Slika 11. Struktura segmenta

S obzirom da je struktura liste riječi već opisana, ostaje samo opisati strukturu same riječi u toj listi:



Slika 12. Struktura riječi

Algoritam za učenje

Algoritam za učenje je, kao i dosad opisani algoritmi upravljan od strane pretprocesiranja. Kao i dosad, proces pretprocesiranja daje riječ po riječ, a proces učenja te riječi slaže u vektor. Kada se završi pretprocesiranje cijelog članka, novostvoreni vektor se dodaje u listu vektora. Pregled funkcija korištenih za učenje (u pseudokodu) slijedi:

```
dodaj_rijec_u_vektor(rijec)
{
  lista_seg := dohvati_listu_segmenata;
  radi_dok(ima_clanova_liste_segmenata)
  {
   segment := uzmi_slijedeci_clan_liste_segmenata;
   ako(rijec(segment) = rijec)
   {
    broj_pojava(segment) := broj_pojava(segment) + 1;
   }
  }
  ako(segment_nije_nadjen)
  {
   segment := stvori_novi_segment(rijec);
   dodaj_u_listu_segmenata(segment);
  }
}

spremi_vektor_u_listu_vektora(vektor)
{
  dodaj_vektor_u_listu_vektora(vektor);
}

dodaj_rijec_u_listu_rijeci(rijec)
{
 radi_dok(ima_clanova_liste_rijeci)
 {
  clan_rijec := dohvati_slijedeci_clan_liste_rijeci;
  ako(rijec(clan_rijec) = rijec)
  {
   broj_pojava(clan_rijec) := broj_pojava(clan_rijec)
                              + 1;
  }
 }
 ako(clan_rijec_nije_nadjen)
 {
  clan_rijec := stvori_novi_clan_rijec(rijec);
  dodaj_u_listu_rijeci(clan_rijec);
}}
```

Može se primijetiti da su funkcije korištene za učenje dosta jednostavne – služe samo za izgradnju vektora i održavanje popisa različitih riječi. Nakon završetka pretprocesiranja proces učenja mora provesti postupak izračunavanja TFIDF težina koje se računaju po formuli [3]:

$$a_{ij} = f_{ij} \times \log\left(\frac{N}{n_i}\right)$$

Tu je a_{ij} težina koju računamo, f_{ij} broj pojava zapisan u svakom od segmenata, N je ukupan broj različitih riječi (dakle broj članova liste riječi), a n_i je broj pojava riječi u ukupnom skupu za učenje (podatak sadržan u odgovarajućem članu liste riječi). Funkcija za računanje težina u pseudokodu:

```
izracunaj_tezine()
{
  radi_dok(ima_clanova_liste_vektora)
  {
   vektor := uzmi_slijedeci_clan_liste_vektora;
   lista_seg := dohvati_listu_segmenata(vektor);
   radi_dok(ima_clanova_liste_segmenata)
   {
    seg := uzmi_slijedeci_clan_liste_segmenata;
   rijec := dohvati_rijec_iz_liste_rijeci(rijec(seg));
   tezina(seg) := broj_pojava(seg) *
   log10(ukupan_broj_rijeci(lista_rijeci) /
   broj_pojava(rijec));
   }
  }
}
```

Nakon završetka postupka računanja TFIDF težina, završen je proces učenja. Lista vektora je napunjena vektorima sa izračunatim težinama i spremna za klasifikaciju.

Algoritam za klasifikaciju

Prvi dio koraka algoritma za klasifikaciju se uopće ne razlikuje od učenja: proces pretprocesiranja izgrađuje vektor od dokumenta za klasifikaciju. Zatim se tome vektoru računaju težine (sa dodatnim uvjetom – ako u novom dokumentu postoji riječ koja nije nađena tijekom učenja, ona se odbacuje). Zato će se dalje opisivati proces klasifikacije sa pretpostavkom da je klasificirani dokument već vektoriziran i da su mu izračunate težine.

Slijedeći korak je računanje udaljenosti svih vektora iz liste vektora od vektora koji se klasificira. Tu je neizbježna iteracija kroz cijelu listu vektora i računanje udaljenosti za svaki vektor. Slijedi prikaz načina računanja udaljenosti između dva vektora – računa se Euklidska udaljenost:

```
izracunaj_udaljenost_izmedju_vektora(vek1, vek2)
{
  udaljenost := 0;
  ako(duzina(vek1) >= duzina(vek2))
  {
   duzi := vek1;
   kraci := vek2;
  }
  inace
  {
   duzi := vek2;
   kraci := vek1;
  }
  radi_dok(ima_segmenata_duzeg_vektora)
  {
   seg_duzi := uzmi_slijedeci_segment_duzeg;
   radi_dok(ima_segmenata_kraceg_vektora)
   {
    seg_kraci := uzmi_slijedeci_segment_kraceg;
    ako(rijec(seg_duzi) = rijec(seg_kraci))
    {
     postavi_zastavicu(nadjen_par);
     prekini_petlju;
    }
   }
   ako(postavljena_zastavica(nadjen_par))
   {
    udaljenost := udaljenost +
    (tezina(seg_duzi) - tezina(seg_kraci)) *
    (tezina(seg_duzi) - tezina(seg_kraci));
    postavi_oznaku(seg_kraci);
   }
   inace
   {
    udaljenost := udaljenost + (tezina(seg_duzi)) *
    (tezina(seg_duzi));
   }
  }
  radi_dok(ima_segmenata_kraceg_vektora)
  {
    seg_kraci := uzmi_slijedeci_segment_kraceg;
```

```
     ako(nije_postavljena_oznaka(seg_kraci))
     {
      udaljenost := udaljenost +
      tezina(seg_kraci) * tezina(seg_kraci);
     }
    }
udaljenost := drugi_korijen_od(udaljenost);}
```

Paralelno sa računanjem udaljenosti za vektore se provodi i određivanje k najbližih susjeda. Naime, prije nego počne sam proces računanja stvara se privremena lista koja sadrži točno onoliko članova koliko iznosi k. Nakon svakog izračuna udaljenosti ta se udaljenost dodaje u tu listu. Dodavanje se provodi tako da je lista cijelo vrijeme sortirana po udaljenosti, i umetanjem svake nove udaljenosti, ako je lista puna, najveća udaljenost sortiranjem ispada iz liste. Opis funkcije za dodavanje u listu k najbližih susjeda:

```
dodaj_u_listu_susjeda(udaljenost, kategorija)
{
  radi_dok(ima_clanova_liste_susjeda)
  {
    susjed := uzmi_slijedeci_clan_liste_susjeda;
    ako(nije_postavljena_zastavica(premjestaj))
    {
     ako(prazan(susjed))
     {
       udaljenost(susjed) := udaljenost;
       kategorija(susjed) := kategorija;
     }
     inace ako(udaljenost(susjed) > udaljenost)
     {
       novi_susjed := stvori_novog_susjeda;
       kategorija(novi_susjed) := kategorija;
       udaljenost(novi_susjed) := udaljenost;
       pom := susjed;
       susjed := novi_susjed;
       novi_susjed := pom;
       postavi_zastavicu(premjestaj);
     }
    }
    inace
    {
     pom := susjed;
     susjed := novi_susjed;
     novi_susjed := pom;
    }}}
```

Nakon procesa računanja svih udaljenosti lista je popunjena sa k najbližih susjeda. Još preostaje odrediti većinu, tj. rezultat klasifikacije. To čini funkcija koja je dijelom po izvedbi identična onoj za određivanje rezultata klasifikacije kod naivnog Bayesovog klasifikatora. Pseudokod funkcije:

```
odredi_klasifikaciju()
{
 temp_lista := stvori_novu_temp_listu;
 radi_dok(ima_clanova_liste_susjeda)
 {
  susjed := uzmi_slijedeci_clan_liste_susjeda;
  radi_dok(ima_clanova_temp_liste)
  {
    temp_susjed := uzmi_slijedeci_clan_temp_liste;
    ako(kategorija(temp_susjed) = kategorija(susjed))
    {
     broj_clanova(temp_susjed) :=
     broj_clanova(temp_susjed) + 1;
     postavi_zastavicu(susjed_nadjen);
    }
  }
  ako(zastavica_postavljena(susjed_najden))
  {
    temp_susjed := stvori_novog_temp_susjeda;
    temp_susjed := susjed;
  }
 }
 max := 0;
 ime_max := «»;
 radi_dok(ima_clanova_temp_liste)
 {
    temp_susjed := uzmi_slijedeci_clan_temp_liste;
    ako(broj_clanova(temp_susjed) >= max)
    {
     ako(broj_clanova(temp_susjed) = max)
     {
       ime_max := ime_max + «;» +
       kategorija(temp_susjed);
     }
     inace
     {
      ime_max := kategorija(temp_susjed);
      max := broj_clanova(temp_susjed);
     }
    }
  }
 vrati ime_max;}
```

Optimizacija algoritma za učenje i klasifikaciju

Slično kao i za naivni Bayesov klasifikator, i k – nn originalna izvedba ima loše performanse što se tiče vremenskog trajanja izvođenja. Osim neizbježnog uzroka sporosti – iteracija kroz sve vektore za učenje i računanje udaljenosti za svakog, postoje i neki aspekti originalne izvedbe koji se mogu popraviti i značajno skratiti vrijeme izvođenja. Prvo se nameće ideja ubrzanja liste riječi, i to na način identičan ubrzanju liste riječi u naivnom Bayesovom klasifikatoru – što garantira barem osrednje ubrzanje rada. Važniji segment rada k – nn klasifikatora koji se također može ubrzati je računanje udaljenosti. U trenutnoj izvedbi broj iteracija za izračun udaljenosti između dva vektora je $m*n+n$, gdje su m i n dužine vektora između kojih se računa udaljenost. Međutim, smislio sam algoritam za računanje udaljenosti među vektorima različite duljine koji broj iteracija smanjuje na m, gdje je m dužina duljeg vektora u računu udaljenosti. Za ubrzavanje postojeće izvedbe k – nn algoritma potrebno je samo zamijeniti funkciju za računanje udaljenosti i sortirati vektore između kojih se udaljenost računa. Pseudokod algoritma:

```
izracunaj_udaljenosti_izmedju_vektora(vek1, vek2)
{
  udaljenost := 0;
  ako(nije_prazan(vek2))
     SK := uzmi_slijedeci_segment_vek2;
  inace
     postavi_zastavicu(SD_do_kraja);
  ako(nije_prazan(vek1))
     SD := uzmi_slijedeci_segment_vek1;
  inace
     postavi_zastavicu(SK_do_kraja);

  ako(nije_postavljena(SD_do_kraja) i
      nije_postavljena(SK_do_kraja))
  {
    radi_zauvijek
    {
     ako(rijec(SD) < rijec(SK))
     {
      udaljenost := udaljenost + tezina(SD) *
      tezina(SD);
      SD := uzmi_slijedeci_segment_vek1;
      ako(nema_vise_segmenata_u_vek1)
      {
```

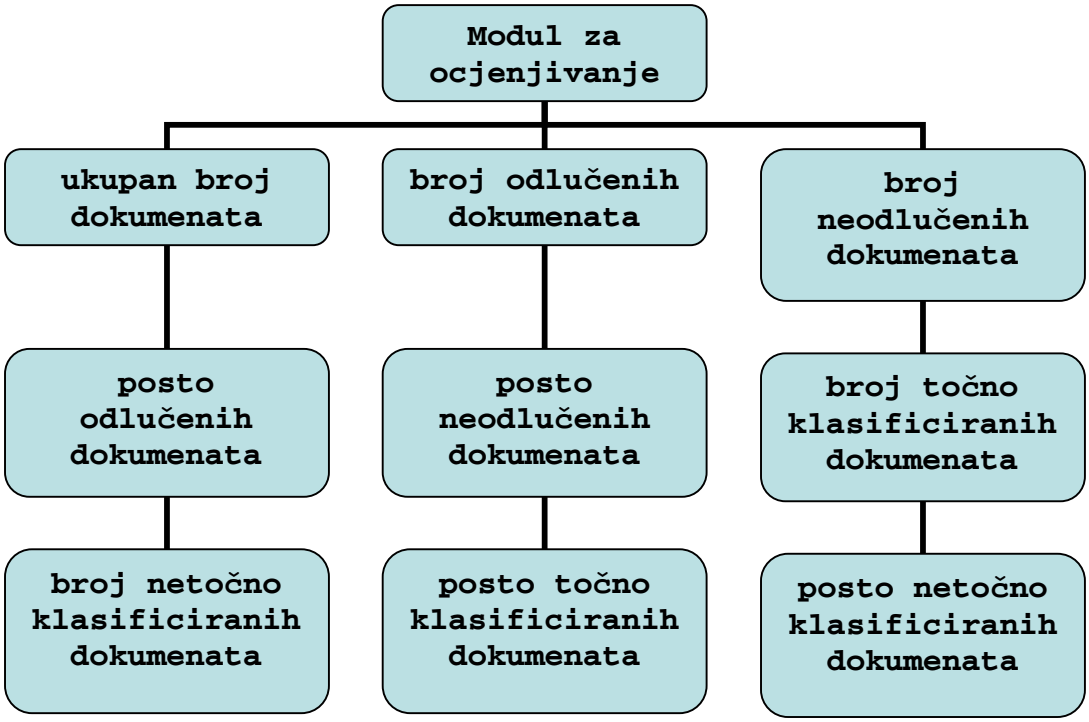
```
      postavi_zastavicu(SK_do_kraja);
      prekini_petlju;
     }
    }
    inace ako(rijec(SD) = rijec(SK))
    {
      udaljenost := udaljenost +
      (tezina(SD) - tezina(SK)) *
      (tezina(SD) - tezina(SK));
      SD := uzmi_slijedeci_segment_vek1;
      ako(nema_vise_segmenata_u_vek1)
      {
        postavi_zastavicu(SK_do_kraja);
      }
      SK := uzmi_slijedeci_segment_vek2;
      ako(nema_vise_segmenata_u_vek2)
      {
        postavi_zastavicu(SD_do_kraja);
      }
      ako(zastavica_postavljena(SK_do_kraja) ili
          zastavica_postavljena(SD_do_kraja))
      {
        prekini_petlju;
      }
    }
    inace
    {
      udaljenost := udaljenost + tezina(SK) *
      tezina(SK);
      SK := uzmi_slijedeci_segment_vek2;
      ako(nema_vise_segmenata_u_vek2)
      {
        postavi_zastavicu(SD_do_kraja);
        prekini_petlju;
      }
    }
  }
}
ako(nije(postavljena_zastavica(SK_do_kraja) i
         postavljena_zastavica(SD_do_kraja)))
{
  ako(postavljena_zastavica(SK_do_kraja))
  {
   udaljenost := udaljenost + tezina(SK) *
   tezina(SK);
   radi_dok(ima_segmenata_u_vek2)
   {
      SK := uzmi_slijedeci_segment_vek2;
```

```
        udaljenost := udaljenost + tezina(SK) *
        tezina(SK);
      }
    }
    ako(postavljena_zastavica(SD_do_kraja))
    {
     udaljenost := udaljenost + tezina(SD) *
     tezina(SD);
     radi_dok(ima_segmenata_u_vek1)
     {
      SD := uzmi_slijedeci_segment_vek1;
      udaljenost := udaljenost + tezina(SD) *
      tezina(SD);
     }
    }
   }
   udaljenost := drugi_korijen_od(udaljenost);
}
```

Opisani algoritam u primjeni, zajedno sa ubrzanjem liste riječi, daje ubrzanje koje skraćuje vrijeme klasifikacije sa nekoliko sati na 10 – 15 minuta za najveću bazu za učenje (Reuters). A priorna složenost algoritma je $O(m)$, što odgovara brojci koja [9] iznosi $O(n)$. n predstavlja dužinu vektora u klasičnom vektorskom prostoru.

Izvedba modula za ocjenjivanje

Oba klasifikatora daju rezultate klasifikacije u jednakom obliku – kao listu zapisa o klasificiranim dokumentima. Tu listu obrađuje modul za ocjenjivanje, da bi se izvukla konačna statistika o rezultatima klasifikacije. Zato modul za ocjenjivanje sadrži u sebi podatkovne članove koji se pune obradom rezultata klasifikacije. Slijedi shematski prikaz podatkovnih članova:



Slika 13. Struktura modula za ocjenjivanje

Algoritam rada modula za ocjenjivanje slijedi na slijedećoj strani:

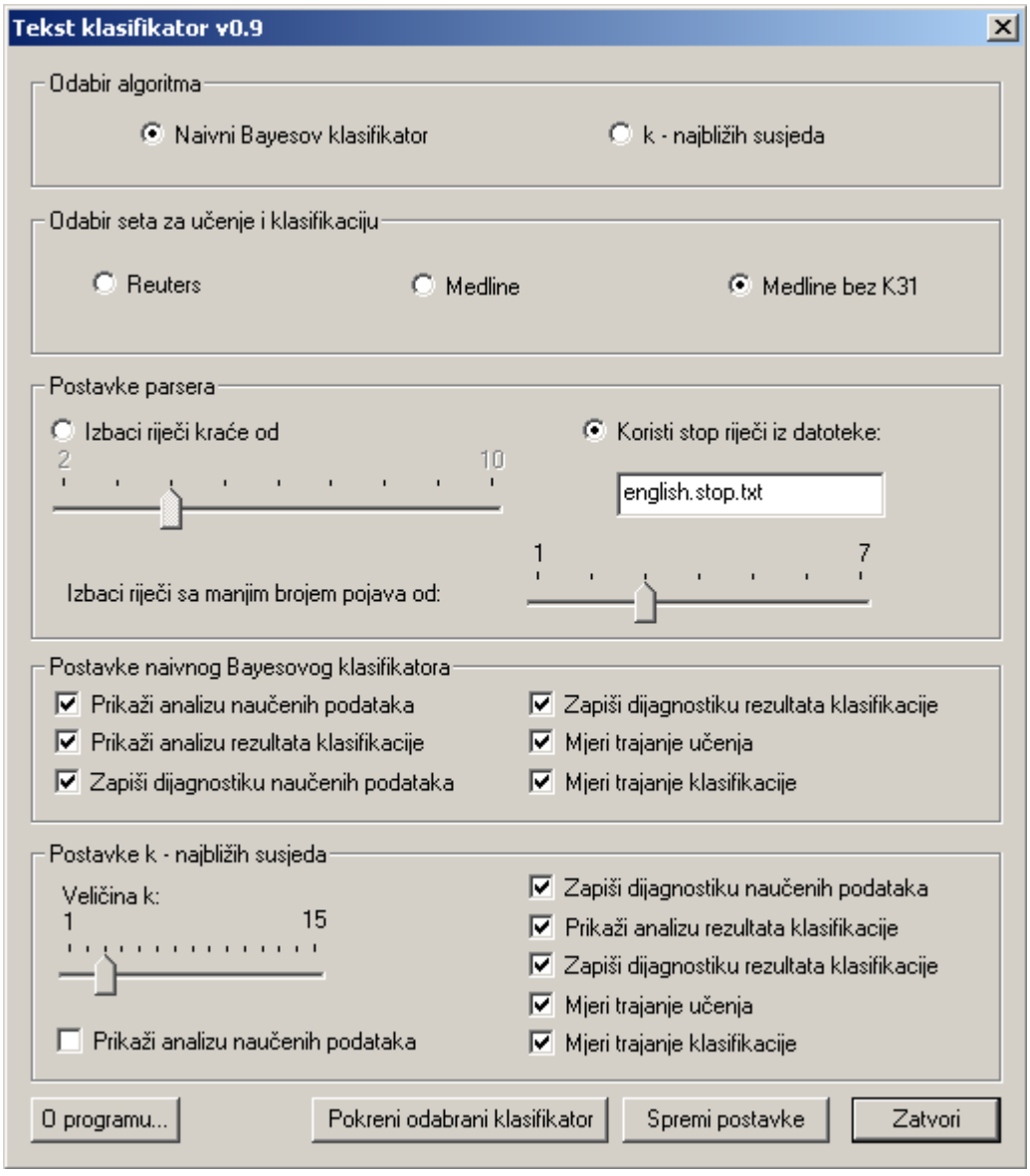
```
obrada_liste_rezultata()
{
 radi_dok(ima_clanova_liste_rezultata)
 {
  rez := uzmi_slijedeci_clan_liste rezultata;
  KK := klasificirana_kategorija(rez);
  TK := tocna_kategorija(rez);
  ukupno_dokumenata := ukupno_dokumenata + 1;
  ako(nema_vise_kateg(TK) i ima_vise_kateg(KK))
  {
   broj_neodlucenih := broj_neodlucenih + 1;
  }
  ako(nema_vise_kateg(TK) i nema_vise_kateg(KK))
  {
   ako(TK = KK)
     broj_tocnih := broj_tocnih + 1;
   inace
     broj_netocnih := broj_netocnih + 1;
   broj_odlucenih := broj_odlucenih + 1;
  }
  ako(ima_vise_kateg(TK) i nema_vise_kateg(KK))
  {
   ako(KK_postoji_u_TK)
    broj_tocnih := broj_tocnih + 1;
   inace
    broj_netocnih := broj_netocnih + 1;
   broj_odlucenih := broj_odlucenih + 1;
  }
  ako(ima_vise_kateg(TK) i ima_vise_kateg(KK))
  {
   radi_za(svaku_kategoriju_u_KK)
   {
     ako(KK_postoji_u_TK)
      broj_tocnih := broj_tocnih + 1;
     inace
      broj_netocnih := broj_netocnih + 1;
   }
   broj_odlucenih := broj_odlucenih + 1;
  }
  posto_neodlucenih := broj_neodlucenih /
  ukupno_dokumenata * 100;
  posto_odlucenih := broj_odlucenih /
  ukupno_dokumenata * 100;
  posto_tocnih := broj_tocnih /
  broj_odlucenih * 100;
  posto_netocnih := broj_netocnih /
  broj_odlucenih * 100;}
```

Osim navedenog, u modulu za ocjenjivanje je implementiran drugi, za ocjenjivanje klasifikatora uobičajeniji pristup, čija je metodologija opisana u odjeljku razmatranja teorije, a opisan je u [4]. Modul računa preciznost i odaziv pomoću mikroprosjeka, a oni će u odjeljku za obradu rezultata biti nužni da bi se izračunala učinkovitost klasifikatora. Zato slijedi pseudokod koji će prikazati način računanja preciznosti i odaziva:

```
izracunaj_preciznost_i_odaziv()
{
 TN := 0;
 FN := 0;
 TP := 0;
 FN := 0;
 radi_dok(ima_clanova_liste rezultata)
 {
   clan_rez := uzmi_slijedeci_clan_liste_rezultata;
   radi_dok(ima_clanova_liste_kategorija)
   {
    clan_kat := uzmi_slijedeci_clan_liste_kateg;
    ako((kategorija(clan_kat) postoji_unutar
        tocna_kategorija(clan_rez)) i
         (kategorija(clan_kat) ne_postoji_unutar
         klasificirana_kategorija(clan_rez)))
    {
      FN := FN + 1;
    }
    ako((kategorija(clan_kat) postoji_unutar
        tocna_kategorija(clan_rez)) i
         (kategorija(clan_kat) postoji_unutar
         klasificirana_kategorija(clan_rez)))
    {
      TP := TP + 1;
    }
    ako((kategorija(clan_kat) ne_postoji_unutar
        tocna_kategorija(clan_rez)) i
         (kategorija(clan_kat) postoji_unutar
         klasificirana_kategorija(clan_rez)))
    {
      FP := FP + 1;
    }
    ako((kategorija(clan_kat) ne_postoji_unutar
        tocna_kategorija(clan_rez)) i
         (kategorija(clan_kat) ne_postoji_unutar
         klasificirana_kategorija(clan_rez)))
    {
      TN := TN + 1;}}}}
```

Vizualni dizajn aplikacije

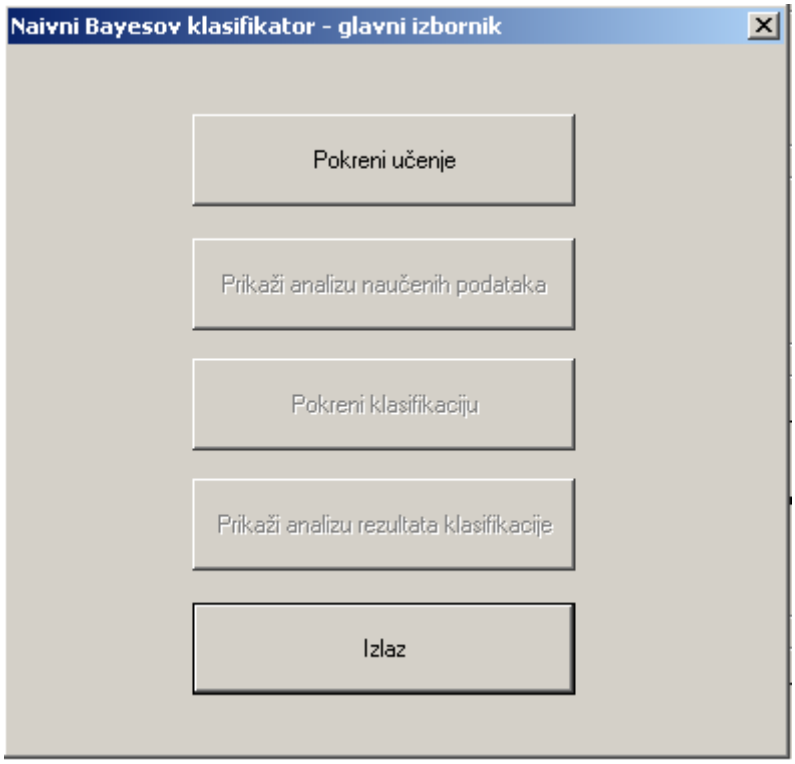
S obzirom da je u izvedbi aplikacije korišten MFC 6.0, radi se o Windows aplikaciji sa vizualnim sučeljem. Aplikacija je temeljena na nizu dijaloga koji omogućuju lako snalaženje i ugodan rad s aplikacijom. Pokretanjem se otvara prvi dijalog u kojem se može podesiti čitav niz postavki – od izbora klasifikatora, skupa za učenje, postavki parsera, pa sve do detaljnijih postavki svakog od klasifikatora. Dijalog izgleda ovako:



Slika 14. Izgled početnog dijaloga

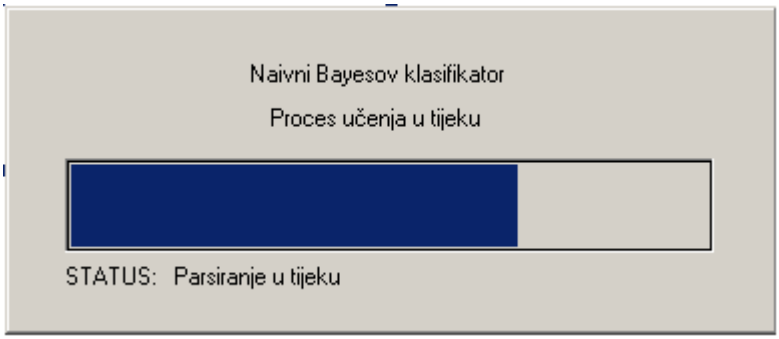
Postavke je moguće spremiti klikom na «Spremi postavke». Klikom na O programu se otvara mali dijalog sa informacijama o porijeklu programa i tehnologiji izvedbe. Odabirom tipke «Pokreni odabrani klasifikator» se otvara

slijedeći dijalog (samo je naslov različit u ovisnosti o odabranom klasifikatoru):



Slika 16. Izgled izbornika za pojedini klasifikator

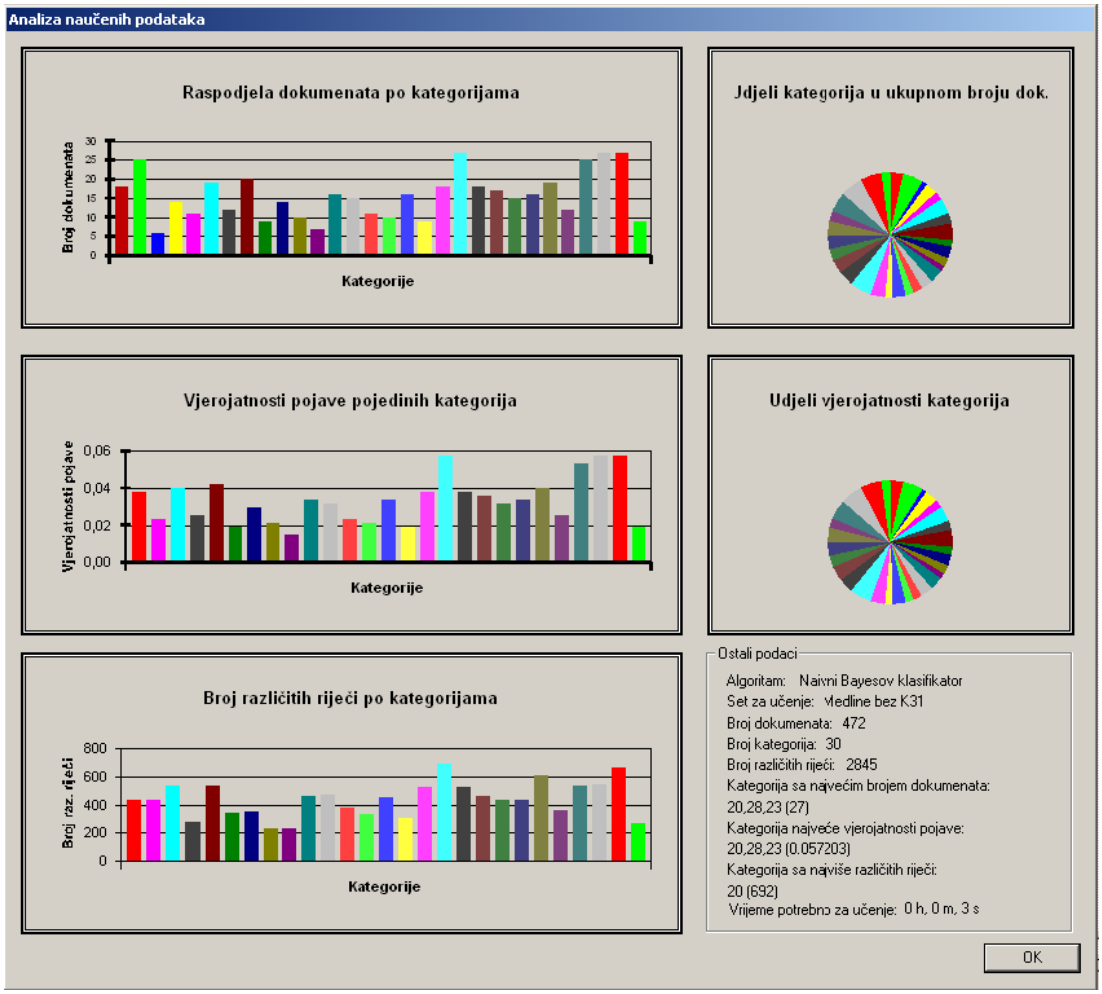
Izbornik određuje osnovne operacije koje se mogu vršiti nad odabranim skupom za učenje. Odabirom «Pokreni učenje» se pokreće proces učenja za odabrani klasifikator, o čemu se korisnik obavještava slijedećim dijalogom:



Slika 17. Izgled dijaloga za vrijeme trajanja učenja i klasifikacije

Linija pokazuje koliko je posla (vremena) preostalo do kraja procesa koji je toku. Isti dijalog se pojavljuje za svaki proces učenja i klasifikacije (sa odgovarajućim naslovom).

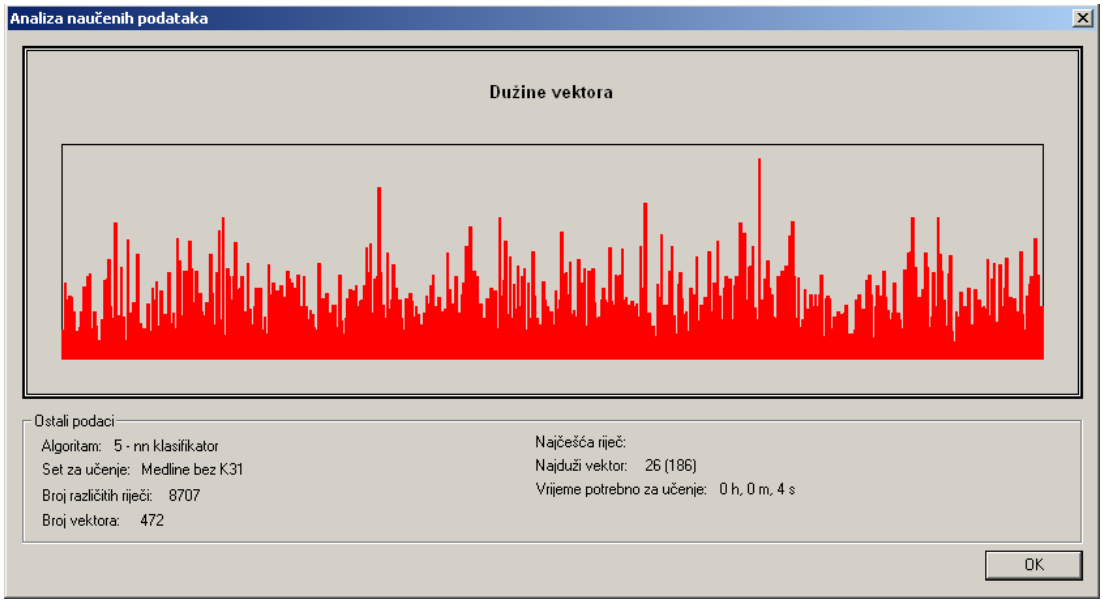
Nakon završenog procesa učenja, ovisno o postavkama, otvara se dijalog sa analizom naučenih podataka. Za naivni Bayesov klasifikator dijalog izgleda ovako:



Slika 18. Analiza naučenih podataka za naivni Bayesov klasifikator

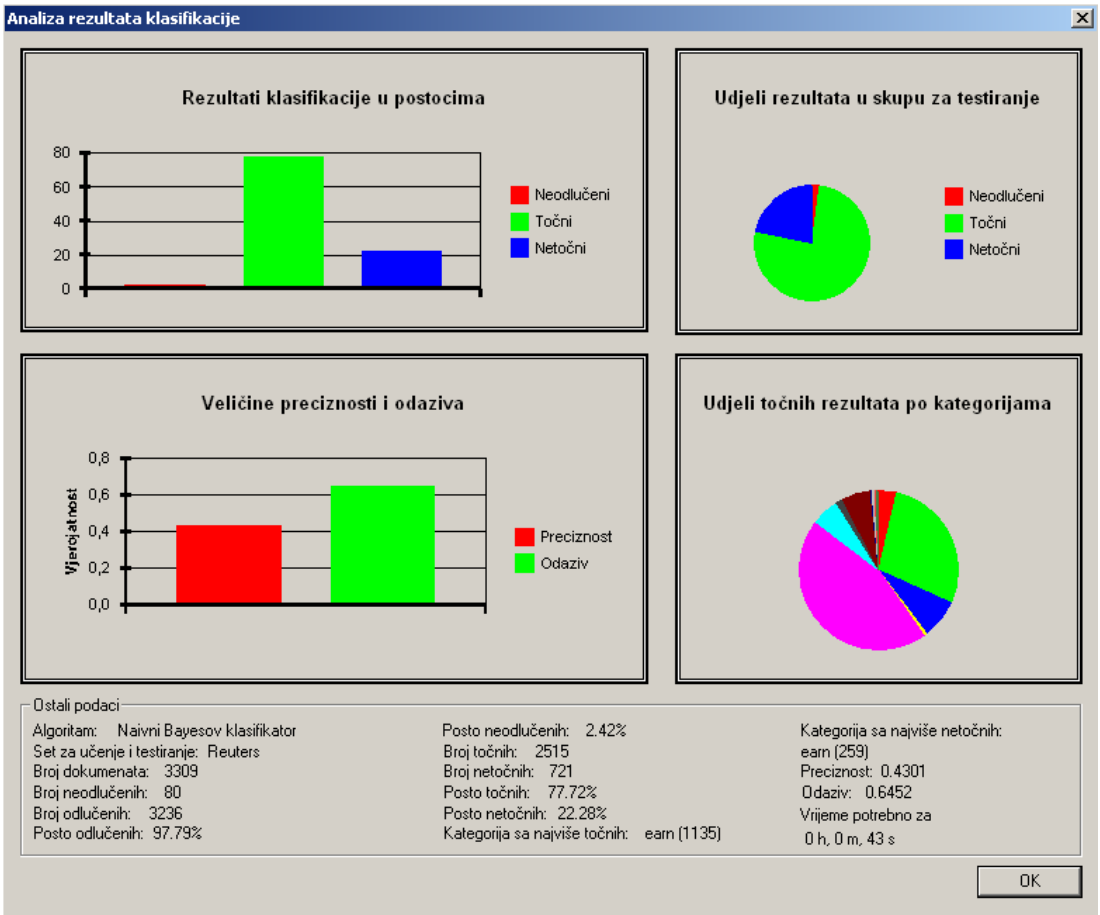
Za naivni Bayesov klasifikator prikazuju se podaci koji su dostupni s obzirom na njegov način učenja.

Za k – nn klasifikator analiza naučenih podataka daje izgled dužina vektora kao ilustraciju liste rijetkih vektora. Također su dostupni i drugi podaci. Slika je na slijedećoj strani.



Slika 19. Analiza naučenih podataka za k-nn klasifikator

Po završetku klasifikacije moguće je pregledati analizu rezultata klasifikacije, za koju se (za oba algoritma jednak) otvara novi dijalog, na kojemu se mogu vidjeti najznačajniji rezultati klasifikacije:



Slika 20. Analiza rezultata klasifikacije za oba klasifikatora

Upute za uporabu aplikacije

Potrebno za uspješan rad aplikacije

- 128 ili više MB RAM – a
- čim brži procesor
- Windows 2000 ili Windows XP
- datoteke MSCHRT20.OCX, MSCHRT20.DEP, MSCHRT20.SRG snimljene u folder Windows\system32
- datoteke TekstKlasifikator.exe, ReutersTraining.txt, ReutersTest.txt, MedlineTrainingK31.txt, MedlineTraining.txt, MedlineTestK31.txt, MedlineTest.txt, english.stop.txt spremljene na disk u isti folder

Pojašnjenja opcija na početnom dijalogu

«**Odabir algoritma**» - selektiranjem se odabire algoritam s kojim se radi
«**Odabir seta za učenje i klasifikaciju**» - selektiranjem se odabire skup za učenje

«**Postavke parsera**» - više opcija:
Moguće je izabrati izbacivanje riječi do određene veličine ili korištenje datoteke sa popisom stop riječi. Veličina riječi se određuje pomakom kliznika, dok se ime datoteke sa stop riječima može unijeti u kućicu ispod odabira opcije sa stop riječima. Pritom paziti da se upisana datoteka nalazi u istom folderu kao i aplikacija.
«**Izbaci riječi sa manjim brojem pojava od**» - drugi dio postavke za izlučivanje karakteristika – kliznikom se određuje koliko se puta najmanje riječ treba pojaviti da ne bi bila odbačena.

«**Postavke naivnog Bayesovog klasifikatora**» - više opcija:
«**Prikaži analizu naučenih podataka**» - određuje se da li se nakon učenja treba prikazati dijalog sa grafovima koji prikazuju neke statistike naučenih podataka – isti se dijalog može pokrenuti i naknadno
«**Zapiši dijagnostiku naučenih podataka**» - odabirom ove opcije se automatski po završetku učenja u folderu u kojem se nalazi aplikacija stvara

tekstualna datoteka sa imenom «NauceniPodaciDijagnostikaNB.txt», u kojoj se nalazi potpun ispis podataka iz podatkovne strukture Index naivnog Bayesovog klasifikatora – dakle ispis svih naučenih podataka.

«**Prikaži analizu rezultata klasifikacije**» - određuje da li se nakon učenja treba prikazati dijalog sa analizom rezultata klasifikacije – isti se dijalog može prikazati i naknadno

«**Zapiši dijagnostiku rezultata klasifikacije**» - odabirom ove opcije se automatski po završetku klasifikacije u folderu u kojem se nalazi aplikacija stvara tekstualna datoteka sa imenom «Rezultati.txt», u kojoj se nalazi detaljan ispis rezultata klasifikacije.

«**Mjeri vrijeme učenja**» - odabirom se određuje da li je potrebno mjeriti trajanje učenja – rezultat mjerenja se prikazuje na dijalogu sa analizom rezultata učenja.

«**Mjeri vrijeme klasifikacije**» - odabirom se određuje da li je potrebno mjeriti trajanje klasifikacije – rezultat mjerenja se prikazuje na dijalogu sa analizom rezultata klasifikacije.

«**Postavke k- najbližih susjeda**» - više opcija:

«**Veličina k**» - micanjem klizača se određuje željena veličina k, tj. broj najbližih susjeda

«**Prikaži analizu naučenih podataka**» - određuje se da li se nakon učenja treba prikazati dijalog sa grafovima koji prikazuju neke statistike naučenih podataka – isti se dijalog može pokrenuti i naknadno

«**Zapiši dijagnostiku naučenih podataka**» - odabirom ove opcije se automatski po završetku učenja u folderu u kojem se nalazi aplikacija stvara tekstualna datoteka sa imenom «NauceniPodaciDijagnostikaKNN.txt», u kojoj se nalazi potpun ispis podataka iz podatkovne strukture Lista vektora k – nn klasifikatora – dakle ispis svih naučenih podataka.

«**Prikaži analizu rezultata klasifikacije**» - određuje da li se nakon učenja treba prikazati dijalog sa analizom rezultata klasifikacije – isti se dijalog može prikazati i naknadno

«**Zapiši dijagnostiku rezultata klasifikacije**» - odabirom ove opcije se automatski po završetku klasifikacije u folderu u kojem se nalazi aplikacija stvara tekstualna datoteka sa imenom «Rezultati.txt», u kojoj se nalazi detaljan ispis rezultata klasifikacije.

«**Mjeri vrijeme učenja**» - odabirom se određuje da li je potrebno mjeriti trajanje učenja – rezultat mjerenja se prikazuje na dijalogu sa analizom rezultata učenja.

«**Mjeri vrijeme klasifikacije**» - odabirom se određuje da li je potrebno mjeriti trajanje klasifikacije – rezultat mjerenja se prikazuje na dijalogu sa analizom rezultata klasifikacije.

Pojašnjenja opcija na dijalogu određenog klasifikatora

Sve su opcije identične za oba klasifikatora, pa će zato dijalog biti opisan jednom.

«**Pokreni učenje**» - odabirom ove tipke počinje proces učenja na određenom skupu za učenje sa određenim klasifikatorom. Pojavljuje se dijalog koji prikazuje napredak učenja i status učenja. Za daljnji rad potrebno je pričekati da proces učenja završi.

«**Prikaži analizu naučenih podataka**» - funkcija ove tipke je identična istoimenoj opciji na početnom dijalogu. Ovisno o veličini skupa za učenje, ponekad je potrebno nešto više vremena za otvaranje ovog dijaloga.

«**Pokreni klasifikaciju**» - odabirom ove tipke pokreće se proces klasifikacije. Pojavljuje se dijalog koji prikazuje napredak i status klasifikacije. Za daljnji rad potrebno je pričekati da proces klasifikacije završi.

«**Prikaži analizu rezultata klasifikacije**» - funkcija ove tipke je identična istoimenoj opciji na početnom dijalogu. Ovisno o veličini skupa za učenje, ponekad je potrebno nešto više vremena za otvaranje ovog dijaloga.

4. REZULTATI

Postavke klasifikatora

Prije prezentacije dobivenih rezultata potrebno je navesti postavke klasifikatora s kojima su rezultati dobiveni.

- postavke parsera:
  - izbacuju se riječi sa manje od 3 pojave
  - izbacuju se najčešće korištene riječi [10]
- ispitivani algoritmi za klasifikaciju:
  - naivni Bayesov klasifikator
  - 1 – nn
  - 2 – nn
  - 3 – nn
  - 5 – nn
  - 7 – nn
  - 9 – nn
- ispitivani skupovi za učenje:
  - Medline
  - Medline K31
  - Reuters 21450 ApteMod

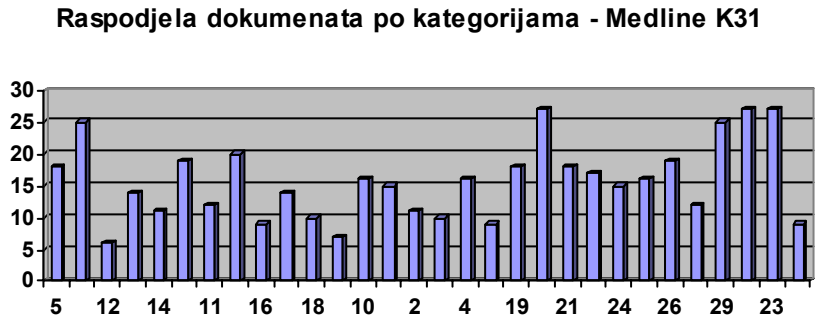
Statistike skupova za učenje

Slijede neki opći statistički podaci o svakom skupu za učenje, kako bi se dobila neka predodžba o karakteristikama skupova za učenje koji se koriste u ispitivanju.

Medline K31

- broj dokumenata za učenje: 472
- broj dokumenata za testiranje: 224
- broj kategorija: 30
- broj različitih riječi: 2845 (nakon izlučivanja karakteristika)

Medline K31 ima dosta šaroliku raspodjelu dokumenata po kategorijama, ali je u usporedbi sa drugim skupovima ta distribucija čak i najujednačenija. Glavni «nedostatak» ovog skupa je mali broj članaka.

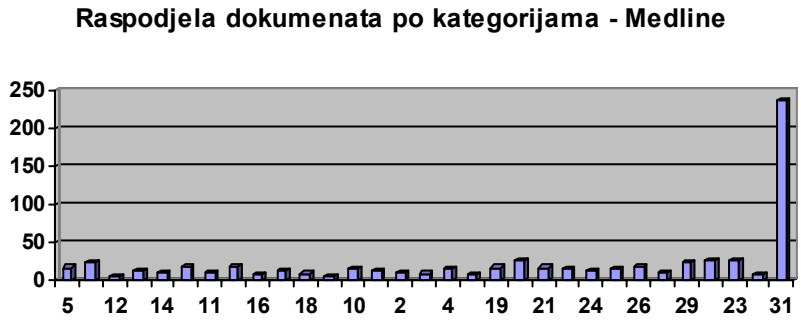


Slika 21. Raspodjela dokumenata po kategorijama za Medline K31

Medline

- broj dokumenata za učenje: 708
- broj dokumenata za testiranje: 325
- broj kategorija: 31
- broj različitih riječi: 3908

Šira verzija Medlinea ima za oko 50% više članaka, ali su ti novi članci vrlo neravnomjerno raspodjeljeni po kategorijama tako da je većina stavljena u jednu kategoriju (to su članci bez kategorije). Zato je distribucija članaka izrazito neujednačena, što bi moglo rezultirati lošijim performansama klasifikatora na ovom skupu za učenje.



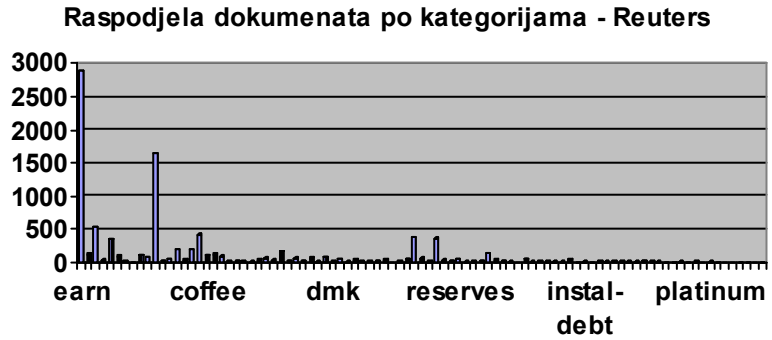
Slika 22. Raspodjela dokumenata po kategorijama za Medline

Reuters

- broj dokumenata za učenje: 7790
- broj dokumenata za testiranje: 3309
- broj kategorija: 93

- broj različitih riječi: 18514

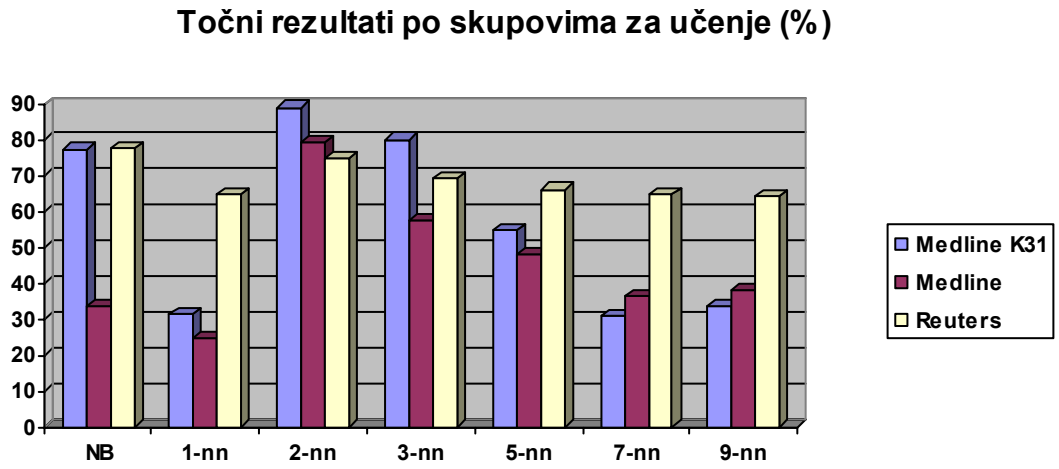
Najveći skup za učenje ima izniman broj kategorija i vrlo neujednačenu distribuciju. Međutim, broj članaka je znatno veći od Medline-a, pa bi bilo realno očekivati barem nešto bolje rezultate od Medline-a.



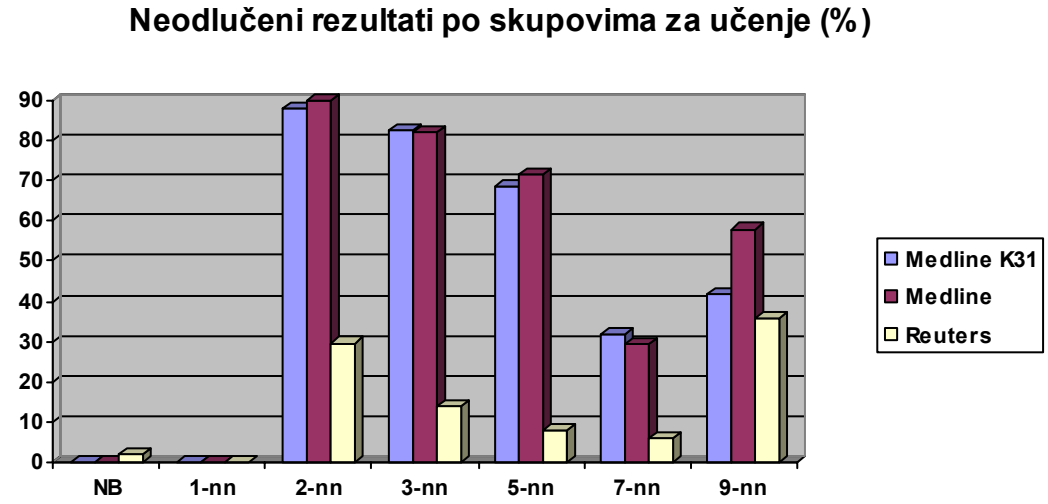
Slika 23. Raspodjela dokumenata po kategorijama za Reuters

Rezultati interpretirani jednostavnim ocjenjivanjem učinkovitosti

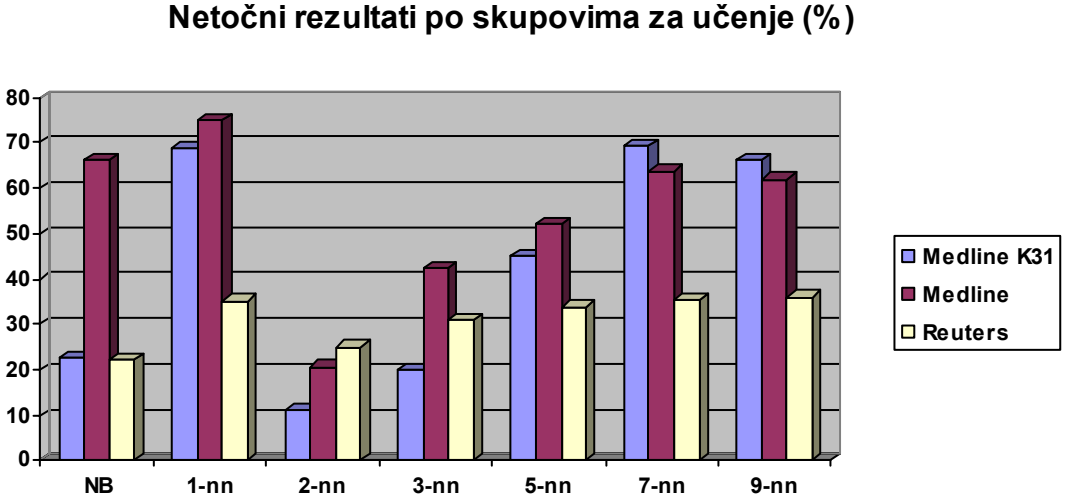
Pod trivijalnim ocjenjivanjem se smatra prvi način rada modula za ocjenjivanje, koji ocjenjuje rezultate na najjednostavniji mogući način koji se (uz nekoliko pojednostavljenja) može svesti na – ako je izlaz klasifikatora jednak kategoriji klasificiranog članka, klasifikacija je točna, inače je netočna. Također postoji slučaj kada se klasifikator nije odlučio za točno određenu kategoriju - tada se klasifikacija naziva neodlučenom. Navedeni način ocjenjivanja je detaljno predstavljen u opisu izvedbe modula za ocjenjivanje. Usporediti će se brojevi točnih, netočnih i neodlučenih rezultata po algoritmima za sve skupove za učenje.



Slika 24. Točni rezultati po skupovima za učenje



Slika 25. Neodlučeni rezultati po skupovima za učenje



Slika 26. Netočni rezultati po skupovima za učenje

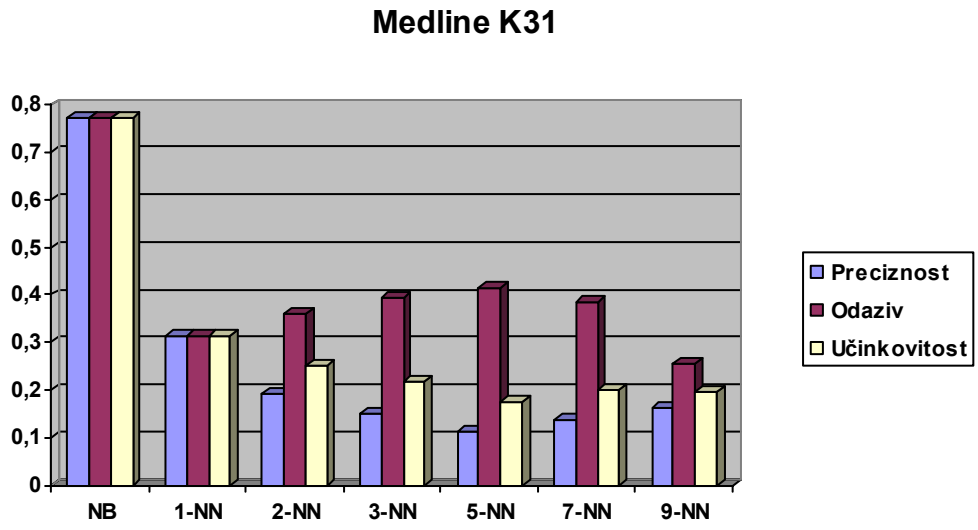
Iz prvog grafa je očito da su distribucije dokumenata u skupovima za učenje jedan od važnijih faktora koji utječe na performanse klasifikatora, s obzirom da je Medline skup polučio najgore rezultate klasifikacije u više od polovice klasifikatora. Nadalje, može se zamijetiti da k – najbližih susjeda dobro parira performansama naivnom Bayesovom klasifikatoru. Dapače, u 2 – nn inačici i prelazi njegove performanse za Medline K31 skup. No, treba imati u vidu rezultate s drugog grafa – k – najbližih susjeda ima značajan broj neodlučenih klasifikacija, što znači da k – nn daje izvrsne rezultate, ali to su rezultati koji dolaze iz onog mnogo manjeg dijela klasificiranih dokumenata za koje je klasifikator dao odluku, tako da rezultate k – nn treba uzeti sa zadrškom. Vidljiva je velika osjetljivost k – nn algoritma na broj podataka za učenje – za male skupove za učenje postoji velik broj neodlučenih dokumenata, dok se kod Reutersa taj broj značajno smanji. Istovremeno se za k – nn kod Reutersa zadržavaju prilično stabilne performanse koje osciliraju između 60 i 70% bez obzira na inačicu – dakle sveukupne performanse k – nn algoritma se značajno poboljšavaju kod većeg skupa za učenje.

Rezultati interpretirani standardnim načinom ocjenjivanja učinkovitosti

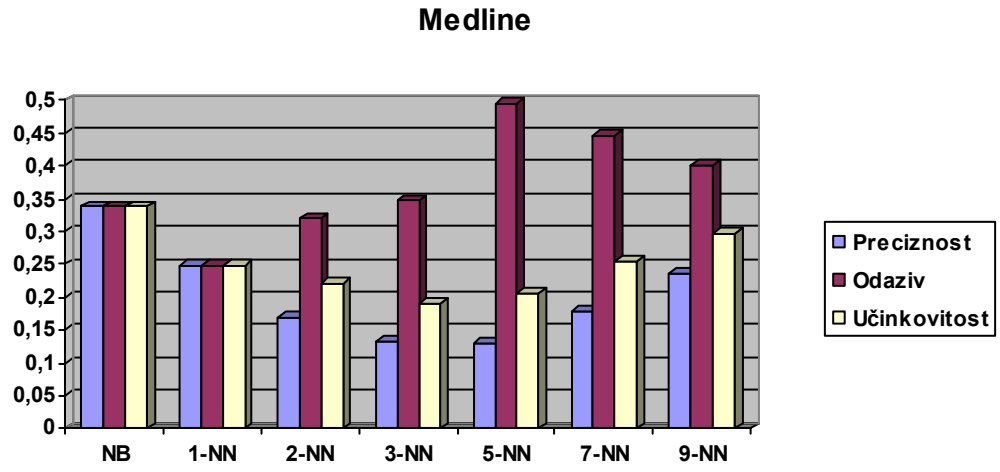
Standardni način uključuje izračun preciznosti i odaziva, i preko njih učinkovitosti klasifikatora. Detaljni teoretski prikaz načina izračuna navedenih veličina može se pronaći u odjeljku teoretskih razmatranja koji se bavi metodologijom ocjenjivanja. Potrebno je još dodatno napomenuti da se za izračun preciznosti i odaziva koristi mikroprosjek, a za izračun učinkovitosti koristit će se formula za izračun učinkovitosti iz [4]:

$$F_\alpha = \frac{1}{\alpha \frac{1}{\mathrm{Pr}} + (1-\alpha)\frac{1}{\mathrm{Re}}}$$

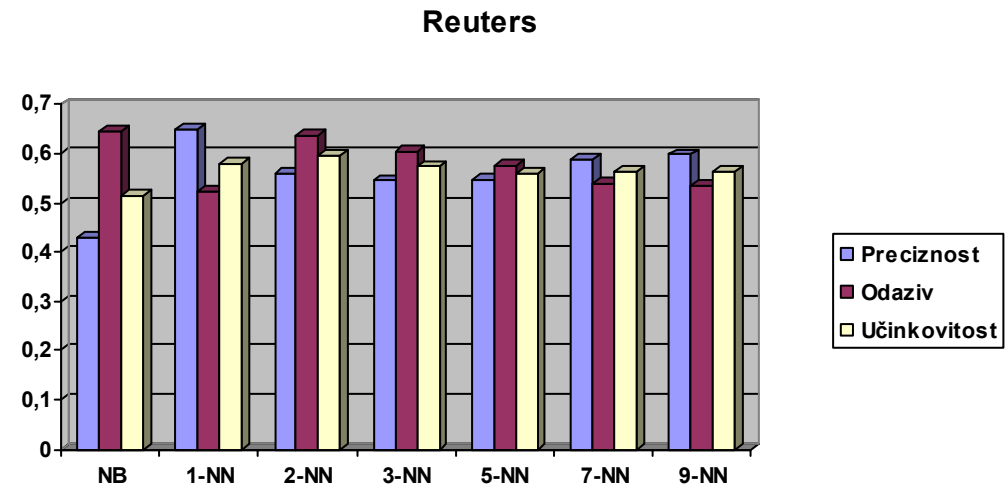
Za faktor α koristit će se 0.5 – jednaka važnost se pridaje preciznosti i odazivu. Slijedi pregled veličina preciznosti, odaziva i učinkovitosti za sve algoritme i za svaki skup za učenje posebno.



Slika 27. Rezultati za Medline K31



Slika 28. Rezultati za Medline



Slika 29. Rezultati za Reuters

Rezultati potvrđuju već utvrđeno u prošlom odjeljku i dodaju neke nove spoznaje. Jasno je vidljivo da Bayesov naivni klasifikator ima očitu prednost nad k – nn algoritmom kada se radi o malom broju dokumenata u skupu za učenje. Također je jasno da su obje metode osjetljive na izrazito neujednačenu distribuciju članaka po kategorijama kod malog broja dokumenata, s obzirom da i Medline i Reuters imaju neujednačenu distribuciju, a razlika u broju članaka je značajna. Rezultati na Reuters skupu za učenje odgovaraju rezultatima koje navodi [4] – k – nn na Reutersu ima nešto bolje rezultate od naivnog Bayesovog klasifikatora. U navedenom izvoru se radi o rezultatima na Reuters – 21578 skupu za učenje, dok se u ovom radu koristi Reuters - 21450 ApteMod, a osim toga moguće su i varijacije u načinu ocjenjivanja, pa otuda manje varijacije (oko 0.05) u odnosu na izvor. Zanimljivo je također primijetiti kako k – nn klasifikator ima ujednačene performanse bez obzira na veličinu k kod Reuters skupa za učenje – što se može objasniti većim brojem dokumenata unutar vektorskog prostora. Time se (pod uvjetom da vrijedi *a priori* pretpostavka k-nn klasifikatora – da su dokumenti iz iste kategorije bliži jedan drugome u vektorskom prostoru) povećava broj vektora koji pripadaju istoj kategoriji, pa se automatski i povećava vjerojatnost da se neki dokument svrsta u tu kategoriju, ako se nađe u blizini. Iz rezultata je očito da je ispravna *a priori* pretpostavka k – nn klasifikatora.

Naivni Bayesov klasifikator pak može svoje izvrsne rezultate kod malog broja članaka za učenje zahvaliti svom probabilističkom pristupu, zbog kojeg, za razliku od k – nn klasifikatora, ne ovisi o broju dokumenata u skupu za učenje. Logično je za pretpostaviti da performanse naivnog Bayesovog klasifikatora ovise o kvaliteti dokumenata u skupu za učenje, tj. o postojanju nekih specifičnih ključnih riječi po kojima se mogu dobro međusobno razlikovati kategorije.

5. ZAKLJUČAK

Zadatak ovog diplomskog rada je usporedba dvaju poznatih, često primjenjivanih i po performansama uspješnih metoda za klasifikaciju teksta – k – nn metode i naivnog Bayesovog klasifikatora. Usporedba se vrši na dva podskupa Medline skupa za učenje i Reuters - 21450 ApteMod skupu za učenje. Krajnji rezultat je funkcionalna Windows aplikacija koja te algoritme implementira, provodi učenje i klasifikaciju nad zadanim skupovima i te rezultate prezentira korisniku. Brzina izvođenja je zadovoljavajuća, ali se može još povećati zamjenom hashing strukture podataka sa balansiranim binarnim stablom. Time bi se vrijeme izvođenja smanjilo na minimum, s obzirom da je to trenutno najbrža raspoloživa struktura podataka.

Pregled rezultata ukazuje na temeljni zaključak koji odgovara na pitanje koje se nameće još od naslova: koji je od obrađenih klasifikatora bolji? Odgovor se slaže sa već u uvodu spomenutom tvrdnjom iz [1]: ovisi o primjeni. Rezultati su pokazali da je naivni Bayesov klasifikator bolji za skupove sa malo dokumenata za učenje, dok se performanse k – nn klasifikatora slažu sa danas prihvaćenim mišljenjem da on pokazuje bolje rezultate pri skupovima koji sadrže članke općeg sadržaja. Također je zamijećena osjetljivost na broj raspoloživih dokumenata za učenje kod jako neujednačene distribucije dokumenata po kategorijama – i originalni Medline i Reuters imaju neujednačene distribucije, ali su rezultati mnogo bolji zbog 10 puta veće količine dokumenata za učenje. K – nn metoda za Reuters skup za učenje pokazuje ujednačene rezultate bez obzira na odabrani k – također posljedica većeg broja dokumenata za učenje. Također se može zaključiti da učinkovitost naivnog Bayesovog klasifikatora ne ovisi o broju dokumenata za učenje, već o kvaliteti dokumenata (gdje je kvaliteta određena postojanjem ključnih riječi koje su specifične za pojedinu kategoriju).

6. LITERATURA

[1] Baeza – Yates, R., Ribeiro – Neto, B. (1999), «Modern information retrieval», Addison – Wesley, Reading, Massachusets, 1. izdanje.

[2] Mitchell, T. M. (1997), «Machine learning», McGraw – Hill, New York, 1. izdanje.

[3] Liao, Y., Vemuri V.R. (2002), «Using text categorization techniques for intrusion detection», URL: http://wwwcsif.cs.ucdavis.edu/~liaoy/research/text_ss02_html/text_ss02.html.

[4] Sebastiani, F. (1999), «A tutorial on automated text categorisation», Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence, pp. 17 – 19, URL: http://faure.iei.pi.cnr.it/~fabrizio/Publications/Publications.html.

[5] Izvor za Reuters 21450 ApteMod set za učenje: http://moscow.mt.cs.cmu.edu:8081/reuters_21450/apte/.

[6] Izvor za Medline set za učenje: http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/medl/.

[7] Joachims, T. (1996), «A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization», Proceedings of ICML-97, 14th International Conference on Machine Learning, pp. 3 – 5, URL: http://reports-archive.adm.cs.cmu.edu/anon/1996/CMU-CS-96-118.ps.

[8] Han, E.-H., Karypis, G., Kumar, V. (1999), «Text categorization using weight adjusted k – nearest neighbor classification», Lecture Notes in Computer Science, pp. 3, URL: ftp://ftp.cs.umn.edu/dept/users/kumar/WEB/papers.html.

[9] Duda, R. O., Hart, P. E., Stork G. D. (2000), «Pattern classification», 2. izdanje, Wiley – Interscience.

[10] Izvor za datoteku sa stop riječima: ftp://ftp.sunet.se/pub/unix/databases/full-text/smart/english.stop

[11] Ribarić, S. (2001), «Raspoznavanje uzoraka», bilješke s predavanja.

[12] Ribarić, S., Dalbelo Bašić, B. (2002), «Inteligentni sustavi», bilješke s predavanja – dio: modeliranje neizvjesnosti. URL: http://www.zemris.fer.hr/education/is/NastavniMaterijal.html

[13] Dalbelo Bašić, B. (2002), «Strojno učenje», bilješke s predavanja – dio: učenje na temelju primjera. URL: http://www.zemris.fer.hr/education/ml/NastavniMaterijal.html