UNIVERZA V LJUBLJANI

SAMODEJNO RAZVRŠČANJE IZVLEČKOV OBJAV V SLOVENSKEM JEZIKU

MAGISTRSKO DELO

BOŠTJAN JERKO

LJUBLJANA

JANUAR 2005

staremu stricu

IZVLEČEK

V magistrskem delu obravnavam samodejno razvrščanje izvlečkov iz člankov v slovenskem jeziku. Za bazo izvlečkov sem uporabil Biomedicino Slovenico, ki se uporablja za ocenjevanje člankov v medicini. Bazo vodi in vzdržuje Inštitut za biomedicinsko informatiko na Medicinski fakulteti. Izvleči se razvrščajo s pomočjo ključnih besed iz MeSH-a (Medical Subject Heading), ki so drevesno urejene in za katere skrbi National Library of Medicine iz Združenih držav Amerike. Postopek samodejnega razvrščanja sem opisal v nekaj korakih. V prvem opisujem pripravo podatkov za obdelavo, torej predpripravo, ki sestorji iz odstranjevanje nedovoljenih znakov in nepolnopomenskih besed, krnjenja. Nato pripravim vektor vseh krnov v učnih dokumentih, kar predstavlja normirani dokument. V tega dodajam tudi besed iz naslova, ki so utežene z večjo utežjo kot besed iz dokumenta, kar temelji na predvidevanju, da imajo besede iz naslova večjo informacijsko težo kot besede iz dokumenta. Naredim tudi preizkus, kako različne uteži besed iz naslova vplivajo na uspešnost razvrščanja in katera utež je najprimernejša. Naslednji korak je samo razvrščanje, ki je izvedeno z metodo TFIDF in enim najbližjim sosedom. Razvrščanje poteka na posameznih ravneh MeSH-a glede na raven v drevesu. Najprej razvrščam na najvišji ravni drevesa, kjer je 15 razredov. Nato preizkusim metodo na nižjih, podrobnejših ravneh. Na koncu sem zastavil načrt za nadaljnje delo.

ZAHVALA

Zahvaljujem se docentu dr. Juretu Dimcu za vso pomoč, ki mi jo je nudil. Za pomoč pri grafičnem prikazu rezultatov se zahvaljujem Gaju Vidmarju. Poleg tega bi se rad zahvalil ženi in hčerkama. Zahvaliti se moram tudi mentorju prof. dr. Nikoli Pavešiču, ki mi je pomagal pri izdelavi magistrske naloge in mi svetoval, ter me usmerjal pri delu na izbrani temi. Nenazadnje se zahvaljujem Mihi Mastenu za lektoriranje.

KAZALO

SLIKE

POGLAVJE 1
UVOD

V tej magistrski nalogi bom opisal postopek za samodejno dodeljevanje ključnih besed dokumentom.

Najprej bom opisal okolje uporabljeno v postopku. Na Inštitutu za biomedicinsko informatiko imamo bazo (Biomedicina Slovenica) objav slovenskih avtorjev medicinske, stomatološke, veterinarske stroke in biologije v tujih revijah, ki so vključene v Science Citation Index. Vodimo tudi evidenco citiranosti, tako da lahko avtorji pri nas dobijo podatke o svojih objavah. Ti podatki se uporabljajo pri ocenjevanju avtorjev, vključenih v raziskovalne skupine. Zapisi so sestavljeni iz imen avtorjev, naslova članka, izvlečka članka in ključnih besed. Nekateri zapisi v slovenskem jeziku imajo tudi angleški prevod naslova članka in izvlečka.

Ključne besede, uporabljene za označevanje zapisov, so izbrane iz spiska Medical Subject Headings (MeSH), ki ga vodi National Library of Medicine in ima 22.568 ključnih besed (`http://www.nhm.nih.gov/mesh/2003/MBrowser.html`). Urejene so v obliki drevesa. Na najvišji ravni je 15 vej, ki so razvejene v podrobnejše ključne besede.

V zapisih Biomedicine Slovenice uporabljamo ključne besede v dveh ravneh. Na prvi ravni so tiste, ki podrobneje opisujejo vsebino članka (največ 5), na drugi ravni pa tiste, ki opisujejo članek širše (2 do 12). Zapisom v Biomedicini Slovenici ključne besede dodeljuje strokovnjak na zahtevanem področju, kar je včasih zahtevno in problematično. Razlog je predvsem v subjektivnosti človeka oziroma strokovnjaka, ki določa ključne besede. S samodejnim dodeljevanjem ključnih besed bomo zmanjšali subjektivni vpliv človeka in omogočili bolj predvidljivo določanje vsebine zapisa.

Med različnimi postopki za samodejno dodeljevanje ključnih besed sem izbral metodi TFIDF in k-najbljižjih sosedov. Ti metodi sta preprosti za uporabo, dokaj uspešni in tudi v praksi večkrat uporabljani. Metoda, ki se zadnje čase uveljavlja, je

metoda podpornih vektorjev, ki se je v mojem primeru izkazala za računsko preveč zahtevno.

Postopek bom opisal v nekaj korakih: predpriprava, učenje in preizkus. Potek predprirave lahko razdelimo na: odstranjevanje ločil, nepolnopomenskih besed in krnjenje.

Običajni postopek priprave metode za samodejno delovanje vključuje postopek učenja metode, zato sem iz korpusa podatkov, ki sem jih imel na razpolago (1364 zapisov), določil učno in testno množico, ki sem ju uporabil v postopku učenja in preizkusa "naučene" metode. Učna množica je imela 1033 zapisov, preostali zapisi pa so bili testni (331 zapisov). V postopku učenja je bilo potrebno določiti povezavo med ključnimi in "pomembnimi" besedami v izvlečkih dokumentov, ki jih poimenujemo zapisi.

Za učenje metode je potrebna predpriprava podatkov, ki tako kot naraščanje izvlečkov poteka samodejno. Predpriprava je nujna, ker so dokumenti (izvlečki) različnih dolžin, njihove naslove sestavljajo različne in različno število besed, in ker imamo opravka z naravnim jezikom, je potrebno sorodne besede pretvoriti v osnovno obliko. To je še posebej pomembno v slovenščini.

V postopku predpriprave je potrebno odstraniti znake, ki imajo majhno informacijsko moč oziroma so težavni za razvrščanje. Gre predvsem za številke in ločila. Zapis sem želel čim bolj skrajšati, zato sem odstranil tudi nepolnopomenske besed. To so tiste besede, ki - podobno kot znaki - vsebujejo malo informacij. Gre predvsem za predloge in besede, ki se zelo pogosto pojavljajo v besedilih in jih zato ne moremo uporabiti za razločevanje pomena besedila. Za pomoč pri določanju nepolnopomenskih besed sem si pomagal z zbirko slovenskih besedil "Nova beseda"(`http://bos.zrc-sazu.si/s_beseda.html`), ki trenutno vsebuje 121 miljonov besed iz slovenskih besedil (tako originalnih kot prevodov). Z iskalnikom po celotnem spisku besed sem lahko določil najpogostejše besede v slovenščini in jih zapisal v spisek nepolnopomenskih besed. Tako očiščeno besedilo sem na koncu predpripraev še krnil. Krnjenje je postopek združevanja besed s sorodnimi pomeni. Postopek je samodejen in običajno temelji na predpostavki, da so besede, ki imajo skupen koren (v najširšem pomenu besede), sorodne. Tako lahko močno zmanjšamo število besed v vektorjih zapisov

in odstranimo vpliv slovnice na zapise besed. Besede, zapisane v različnih sklonih in časih, imajo enak koren, ki predstavljajo leme. Seveda na ta način izgubimo del informacije o pomenu besedila, vendar pa takšne informacije niso nujne za samodejno razvrščanje. Potrebno je poudariti, da je krnjenje subjektiven postopek. Včasih je težko določiti pomensko sorodnost besedam (primer: sta predprostor in prostor sorodni?) in tako je tudi več različnih postopkov krnjenja. Enega sta izdelala Popovič in Willett [22], drugega pa Dimec [21]. Sam sem izdelal še tretjega, ki temelji na preprostem postopku odstranjevanje končnic besedam. Želel sem preprečiti premočno krnjenje, torej določanje sorodnosti pomensko nesorodnim besedam, zato sem raje odstranil manj končnic kot preveč.

Na koncu sem s predpostavko, da so besede iz naslova dokumenta bolj povezane s ključnimi besedami kot besede iz izvlečka, določil utež besedam v naslovu. Po končanem postopku učenja je sledil preizkus, ki sem ga izvedel na več ravneh ključnih besed iz MeSH-a. Za to je bilo potrebno pripraviti ključne besede, ki so bile izbrane iz celotnega drevesa MeSH-a in niso imele stabilne povezave z ravnmi v MeSH-u.

POGLAVJE 2
PREGLED METOD SAMODEJNEGA RAZVRŠČANJA BESEDIL

Najprej si poglejmo definicijo pojma *samodejno razvrščanje besedil*: imamo N dokumentov, ki jih želimo razvrstiti v M razredov. Z uporabo postopka samodejnega razvrščanja besedil nam program, na podlagi določenih skupin lastnosti dokumentov, le-te razvrsti v iste razrede.

Podrobneje: imamo zbirko dokumentov, ki bi jih želeli nekako urediti, tako da bodo enostavneje dostopni. Če nas zanimajo le dokumenti z določenega področja, moramo sedaj pregledati vse dokumente zbirke in izločiti tiste, ki nas zanimajo. Ta postopek je pri večjem številu dokumentov zamuden in včasih tudi nezanesljiv, kar je odvisno predvsem od medsebojne podobnosti dokumentov. Bolj ko so si dokumenti podobni, težje je razbrati, kateri dokument nas zanima, in dlje traja iskanje pravih dokumentov.

Z uvedbo računalnikov, ki postajajo vse hitrejši in imajo vse več pomnilniškega prostora, se je pojavila ideja, da bi za postopek razvrščanja dokumentov v različne razrede uporabili računalnik. Seveda se tukaj takoj pojavi kar nekaj težav, ki jih obravnavam v nadaljevanju, tako v opisu metod za samodejno razvrščanje kot v prikazu primera uporabe.

Vedno, ko imamo opravka z naravnim jezikom, kar seveda velja za dokumente, se pojavijo težave, ki so posebne za določene jezike. Zato preizkusi kot tudi metode, ki so ustrezne za druge jezike, včasih ne delujejo pravilno in jih je potrebno prenesti v jezik obravnavanih besedil. Pojavljajo se tudi ideje za obravnavo dokumentov v različnih jezikih (npr. medjezično iskanje), tako da lahko naprimer iskalni niz zapišemo v enem jeziku, medtem ko sistem išče po dokumentih v drugih jezikih. V takih sistemih se težave sicer ne podvojijo, ker povezave med jeziki vseeno obstajajo, jih je vsekakor več, zato moramo na nekaterih področjih postopke poenostavljati.

V določenem obdobju raziskav metod samodejnega razvrščanja so se pojavile ideje "naučiti" računalnik razumevanja besedila. Nato bi se računalnik odločil, katere ključne besede oziroma kateri razredi bi bili za določen dokument primerni. V praksi se je izkazalo, da so vse zapletenejše metode obravnavanja dokumentov imele resne težave pri širši obravnavi dokumentov in so se (in žal je še vedno tako) bolj obnesle preproste statistične metode, ki so zapisovale verjetnosti povezav med izrazi v dokumentu in razredom, v katerega je dokument razvrščen.

Pri razvrščanju dokumentov v različne razrede lahko dokumente razvrstimo bodisi tako, da je vsak naenkrat le v enem razredu (angl. *nonoverlaping categories*) ali tako, da je dokument v različnih razredih (angl. *overlaping categories*) [25]. V grobem lahko metode za samodejno razvrščanje besedila razvrstimo v dve skupini [25]:

- metode, kjer je osnova dokument,
- metode, kjer je osnova razred.

V prvi skupini metod imamo določeno število dokumentov in pri vsakem novem dokumentu ugotavljamo, v kateri razred spada. Gre torej za vnaprej določene razrede in dokumente, ki sproti prihajajo. V drugi skupini (metode, kjer je osnova razred) pa razvrščanje deluje tako, da imamo določeno število dokumentov, ki jih razvrščamo v razrede, ki se sproti ustvarjajo. V praksi uporabljamo večinoma metode iz prve skupine (kjer je osnova dokument).

V nadaljevanju si bomo podrobneje pogledali različne metode za razvrščanje dokumentov. Nekatere metode so bolj uporabljane, druge manj. Poleg tega so nekatere metode časovno izredno zahtevne, vendar pa so lahko pri ustrezni strojni opremi in ureditvi podatkov tudi zelo uspešne.

2.1 Naivni Bayes

Metoda Naivnega bayesa temelji na osnovni Bayesovi forumuli:

$$P(x|A) = P(A|x)\frac{P(A)}{P(x)} \tag{2.1}$$

Enačba 2.1 nam pove kakšna je verjetnost, da je dokument x v množici dokumentov A.

Sedaj zapišimo 2.1 bolj splošno:

$$P(x_j|A) = \frac{P(A|x_j)P(x_j)}{\sum_i [P(A|x_i)P(x_i)]} \tag{2.2}$$

x_j je v tem primeru eden od dokumentov iz množice dokumentov X, torej lahko zapišemo:

$$x_i \in X \ ; \ i = 1, ..., n$$

Ker je verjetnost $P(x)$ iz 2.1 oziroma $\sum_i [P(a|x_i)P(x_i)]$ iz 2.2 za vse dogodke iz množice X, jo lahko zanemarimo, ker iščemo le najverjetnejšo pripadnost dokumentu. Torej lahko enačbo zapišemo:

$$P(x_j|A) = P(A|x_j)P(x_j) \tag{2.3}$$

Kakor že prej omenjeno, iščemo najverjetnejšo pripadnost, torej maksimum argumenta oz. največjo verjetnost:

$$P_{max} = argmax_{x_j} P(A|x_j)P(x_j) \tag{2.4}$$

Težava se pojavi pri obravnavi verjetnost $P(A|x_j)$, kjer moramo ugotoviti verjetnosti zaporedja novih n izrazov $(P(a_1, a_2, ..., a_n|x_j); A = \{a_1, a_2, ..., a_n\})$, ki se pojavijo v delu učnih podatkov, kjer je rezultat x_j [4]. To je nemogoče doseči, saj vnaprej ne vemo, kateri izrazi se bodo pojavljali v besedilu. Zato poenostavimo obravnavo, tako da določimo medsebojno neodvisnost izrazov v dokumentu a_n in na ta način določimo vsem izrazom enako verjetnost. Tako velja:

$$P(a_1, a_2, ...a_n|x_j) = \prod_{i=1}^{n} P(a_i|x_j) \tag{2.5}$$

Tako smo prišli do enačbe Naivnega Bayesa (2.6), ki jo lahko uporabimo za

postopek učenja in nato razvrščanja podatkov v razrede:

$$P_{maxNB} = argmax_{x_j \in X} \prod_{i=1}^{n} P(a_i|x_j)P(x_j) \tag{2.6}$$

Pri metodi naivnega Bayesa je potrebno z vsakim novim dokumentom ponoviti metodo učenja, kar zahteva več pomnilnika in hrambo starih podatkov [7]. Metoda je za razvrščanje dokumentov časovno zahtevnejša, njena prednost je preprostost izračuna.

2.1.1 Rocchijev razvrščevalnik

Tudi Rocchijev razvrščevalnik temelji na metodi *šarže* (angl. batch). Iz obstoječega vektorja uteži, pripravi nov vektor uteži:

$$w_j = \alpha w_{1,j} + \beta \frac{\sum_{i \in C} x_{i,j}}{n_C} - \gamma \frac{\sum_{i \notin C} x_{i,j}}{n - n_C} \tag{2.7}$$

n je število učnih elementov, $C = \{1 \leq i \leq n : y_i = 1\}$ je množica pravilno razvrščenih (pozitivnih) učnih elementov. Parametri α, β in γ pa določajo vpliv izvirne teže na pozitivne in negativne primere (nepravilno razvrščene) primere [7], $x_{i,j}$ pa so vhodni podatki.

Ta metoda se v literaturi ne opisuje tako pogosto kot naivni Bayes, je pa relativno preprosta. Njena uspešnost je na stopnji uspešnosti naivnega Bayesa. Potrebno pa je poudariti, da metoda izhaja iz okolja "povratna informacija o pomembnosti" (relevance feedback) in je manj preizkušena na področju samodejnega razvrščanja. Tudi način delovanja je zato bolj primeren za okolje "povratna informacija o pomembnosti", kot pa za samodejno razvrščanje.Poglejmo si zakaj.

Pri uporabi v "povratni informaciji o pomembnosti" uporabnik označi rezultate, ki zadovoljujejo njegove potrebe in želje. Na ta način določi pozitivne in negativne uteži. Iz teh rezultatov in po enačbi 2.7 določimo nove uteži, ki jih uporabimo pri ponovnem iskanju. Če pa želimo metodo uporabiti za samodejno razvrščanje, je potrebno uspešno razvrstitev elementa označiti s pozitivno utežjo in neuspešno z negativno. Tako lahko "naučimo" metodo z novo utežjo, ki vključuje uspešnost razvrščanja učnih ele-

mentov.

2.1.2 k-najbljižjih sosedov

Postopek temelji merjenju podobnosti med dokumentom, ki ga razpoznavamo in dokumentom iz učne množice [20]. Najprej si oglejmo kako deluje postopek 1-NN (najbližji sosed):

$$U_M = \{(x_1, c_1), (x_2, c_2), ..., (x_N, c_N)\} \tag{2.8}$$

V enačbi 2.8 je $c_j \in \{c_1, c_2, ..., c_N\}$ ime razreda, ki mu dokument (x_j) pripada. Dokument, ki ga razvrščamo, razvrstimo v razred C_i (z imenom c_i, če je ta najpodobnejši):

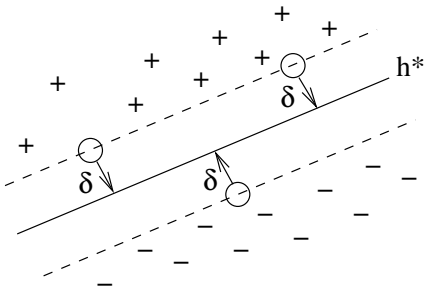
$$d[x, (x, c) \in U_M] = min_{j=1,...,N}\{d[x, (x_j, c_j)]\} \tag{2.9}$$

V enačbi 2.9 je d mera razdalje. Metodo pa lahko posplošimo na k elementov. Nerazvrščeni dokument torej razvrstimo v tisti razred C_i, ki ima k (mora biti liho število) najpodobnejših elementov.

2.1.3 Regresijske metode

Regresijske metode uporabljajo za razvrščanje dokumentov realna in ne binarna števila. Kot primer si bomo ogledali metodo *linearno ujemanje najmanjšega kvadrata*, ki sta jo prvič uporabila Yang in Chute leta 1994 [1]. V tej metodi dodelimo vsakemu dokumentu x_j dva vektorja, eden je vhodni $\overrightarrow{I}(x_j)$, ki ima τ uteženih izrazov, drugi pa izhodni $\overrightarrow{O}(x_j)$, ki ima σ uteži, ki predstavljajo razrede (te uteži so binarne za učne dokumente in nebinarne za testne dokumente). Razvrščevalnik po tej metodi zgradimo tako, da izdelamo matriko $\hat{M}$ velikosti $\sigma \times \tau$, tako da je $\hat{M}\overrightarrow{I}(x_j) = \overrightarrow{O}(x_j)$. Metoda linearnega ujemanja najmanjšega kvadrata izračuna matriko po enačbi 2.10.

$$argmin_{\hat{M}} = ||\hat{M}\overrightarrow{I} - \overrightarrow{O}||_F \tag{2.10}$$



Slika 2.1: Najbolj preprost način uporabe Metode podpornih vektorjev

V enačbi 2.10, kjer računamo absolutno vrednost cenilke $\hat{V}$, določa $argmin_{\hat{M}}(x)$ matriko $\hat{M}$, za katero je vrednost cenilke najnižja.

$$||\hat{V}||_F = \sqrt{\sum_{i=1}^{\sigma}\sum_{j=1}^{\tau}|v_{ij}|^2} \tag{2.11}$$

Enačba 2.11 je Frobeniusova norma matrike $\sigma \times \tau$.

Matrika $\hat{M}$ je izračunana z dekompozicijo singularne vrednosti na učnih podatkih. Element matrike $\hat{M}$ $\hat{m_{tk}}$ določa stopnjo povezave med razredom c_i in izrazom t_k.

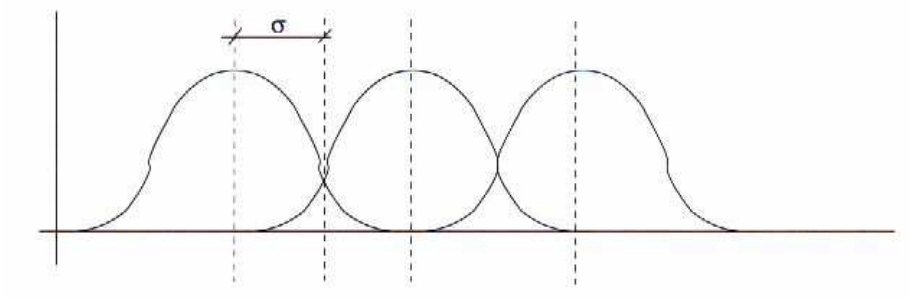
2.1.4 Metoda podpornih vektorjev

Metodo podpornih vektorjev je razvil Vapnik leta 1998. Gre za načelo določanja minimalizacije strukturnega tveganja, ki temelji na statistični teoriji [15].

Ogledali si bomo najbolj preprost primer metode podpornih vektorjev.

Imamo množico parov učnih dokumentov in njihove lege glede na ločilno hiper ravnino $(h^*)(x_i, y_j); i = 1, ..., n$, kjer je x_i vhodni dokumenti in y_i učni podatek, ki ima lahko vrednost $y_i \in \{-1, 1\}$. Z uporabo metode podpornih vektorjev iščemo hiper ravnino (h^*), ki loči elemente x_i glede na vrednost (-1 oz. 1).

Hiper ravnina je izbrana tako, da je vsak vektor, ki je v njej, pravokoten na normalni vektor ravnine. Le-ta je namreč pravokoten na ravnino [15]. Ker je med normalo ravnine in vektorjem v ravnini pravi kot, je njun skalarni produkt enak 0. V



Slika 2.2: Funkcija eksponentnega padanja s kvadratom razdalje

našem preprostem primeru imamo tako enačbo ravnine:

$$h(x) = sign(w^T \cdot w + C \sum_{i=1}^{l} \varepsilon_i) \tag{2.12}$$

Enačbo 2.12 želimo minimalizirati na w (normala ravnine), ε. $C > 0$ je v tej enačbi cenilka napake. Ali drugače [14]:

$$y_i(w^T \phi(x_i) + b) \geq 1 - \varepsilon_i; \varepsilon \geq 0 \tag{2.13}$$

V enačbi 2.13 je učni vektor preslikan v višji (lahko neskončno) dimenzijski prostor s funkcijo ϕ. Tako lahko z metodo podpornih vektorjev iščemo ločilno ravnino z največjim robom v višje dimenzijskem prostoru. Prednost takšnega pristopa je, da lahko nelinearno težavo prevedemo v linearno. Tako uvedemo pojem jedra (angl. kernel). Poglejmo si nekaj možnih jeder:

- linearno: $K(x, y) = x \cdot y$,
- Vovkov polinom: $K(x, y) = (x \cdot y + 1)^p > 0$, kjer p določi uporabnik,
- funkcija eksponentnega padanja s kvadratom razdalje: $K(x, y) = e^{\frac{(-||x-y||^2)}{2\sigma^2}}$ (slika 2.2),
- dvonivojska nevronska mreža: $K(x_i, y_i) = tanh(x \cdot y + \sigma)$, kjer je σ prag, ki ga uporabljamo, da nevronska mreža nima vrednosti izhoda 0, ko je na vhodu vrednost 0.

V [14] predlagajo naslednji postopek uporabe metode podpornih vektorjev:

- skaliraj dokumente,
- uporabi eno od jeder (npr. funkcijo eksponentnega padanja s kvadratom razdalje),
- z validacijo učenja izberi najboljša parametra C in ε,
- preizkusi.

Skaliranje je v postopku metode podpornih vektorjev zelo pomembno, da se izognemo prevelikemu nihanju podatkov. Priporočajo skaliranje v območju [-1,+1] ali [0,1] [14].

2.2 Odločitevno drevo

Odločitvena drevesa temeljijo na preprostem načelu binarne odločitve (da,ne). Ogledali si bomo primer razvrščanja dokumentov v dva razreda (šport in vreme). Razvrščali bomo z uporabo ključnih besed, ki nam bolj ali manj kakovostno določajo pripadnost dokumenta določenemu razredu. Iz podatkov glede na različne vrednosti, ki so v tabeli, oblikujmo odločitveno drevo. Iz te tabele v postopku učenja naredimo odločitveno drevo, ki je odvisno od metode za njegovo izdelavo.

V nadaljevanju opisujem naslednje metode.

2.2.1 ID3

ID3 in naslednike (C4, C4.5 in C5) je razvil Ross Quinlan [6]. V metodi ID3 uporabljamo informacijsko teorijo za določanje oznak z največjo informacijsko močjo (enačba 2.14). Imamo n dokumentov, ki jim določimo verjetnost $p_o; o = 1, ...i, ..., n$. Verjetnosti posameznih dokumentov zapišemo kot porazdelitev verjetnosti $P = (p_1, p_2, ..., p_i, ..., p_n)$.

Kraj	Dogodek	Razred
Ljubljana	predavanje	izobraževanje
Zagreb	atletika	šport
Ljubljana	dvostransko srečanje	politika
Berlin	koncert	umetnost
Sofia	nogomet	šport
Zagreb	nogomet	šport
Berlin	predavanje	izobraževanje
New York	uradni obisk	politika
Maribor	menjava župana	politika

$$I(o) = -P(o)log_2(P(o)) \tag{2.14}$$

Razliko števila bitov za da in ne nam da entropijo. Enačba zanjo je takale:

$$H = -P_{da}(o)log_2(P_{da}(o)) - P_{ne}log_2(P_{ne}(o)) \tag{2.15}$$

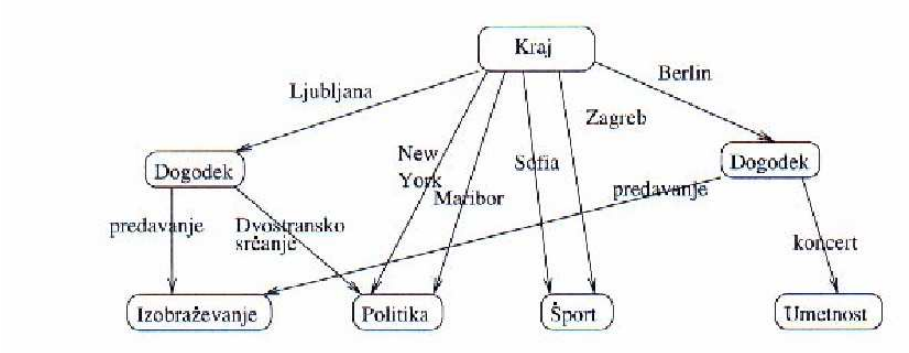
Verjetnost za posamezen dogodek, določimo iz zgornje tabele, kjer preštejemo pojave, ki povzročijo tak ali drugačen rezultat (da ali ne). Število teh pojavov pa delimo s številom vseh pojavov (število vrstic v tabeli).

Oglejmo si postopek za določanje odločitvenega drevesa s postopkom ID3:

Začnemo le z enim vozliščem. Izračunajmo celotno entropijo: [2(razred/izobraževanje), 3 (razred/šport), 3 (razred/politika), 1 (razred/umetnost)]. Entropija (2.15) je torej 1.89. Izračunajmo entropijo za posamezne pojave, ki vplivajo na rezultat. Želimo določiti entropijo podatkov za kraj. Če gremo po vrsti, vidimo:

- dvakrat Ljubljana, Zareb, Berlin, torej je $P_{Ljubljana} = P_{Zagreb} = P_{Berlin} = \frac{2}{9}$,
- enkrat Sofia, New York, Maribor, torej je $P_{Sofia} = P_{NewYork} = P_{Maribor} = \frac{1}{9}$.

Entropija podatkov za kraj je tako 0.9997. Po enakem postopku izračunamo tudi entropijo podatkov za dogodek in dobimo vrednost 0.82.



Slika 2.3: Rezultat metode ID3

Izračunali smo entropijo posameznih besed. Za določitev vozlišča moramo izračunati informacijski prispevek, ki nam pove, kolikšen je prispevek določene delitve k informaciji. Delitev, ki da največjo informacijo, nam določi vozlišče. Informacijski prispevek izračunamo po enačbi 2.16.

$$IP = H_{skupni} - \sum_i \frac{o_i}{o} H(o_i) \tag{2.16}$$

Informacijski prispevek, ki ima največjo vrednost, izbere trenutno vozlišče. Največji informacijski prispevek nam da "Kraj", torej je to prvo vozlišče. Celotno metodo izvedemo na vseh poddrevesih in na koncu dobimo odločitveno drevo (2.3).

2.2.2 C4.5

Metoda C4.5 izboljša metodo ID3 [12] na naslednji način:

- pri določanju odločitvenega drevesa lahko uporabimo tudi učne elemente (ki predstavljajo liste dreves). Za njih ne vemo, kakšen informacijski prispevek bodo imeli, tako da vrednotimo informacijski prispevek elementov (listov drevesa), katerih vrednosti poznamo;
- pri določanju odločitvenega drevesa lahko razvrščamo elemente, ki imajo neznano vrednost (list drevesa), tako da vrednostimo verjetnost posameznih dogodkov.

Če na primer ne bi poznali vrednosti za Kraj, bi lahko uporabili metodo C4.5. V odločitveno drevo bi potemtakem kot verjetnost uvedli vrednost veje.

2.3 Hevristične metode z uporabo pravil

Odločitveno drevo lahko zapišemo tudi z uporabo odločitvenih stavkov. Oglejmo si zgornji primer z odločitvenimi stavki:

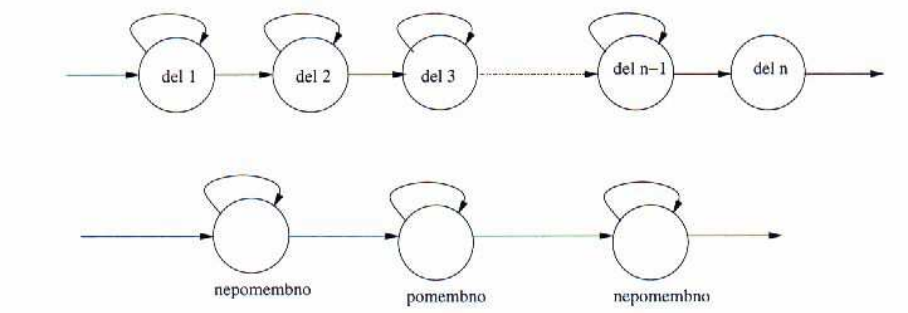
```
if Lokacija==''Ljubljana'' and Dogodek==''predavanje'' then Izobraževanje
if Lokacija==''Ljubljana'' and Dogodek==''bilateralno srečnje'' then
    Politika
if Lokacija==''Sofia'' then Šport
if Lokacija==''Zagreb'' then Šport
if Lokacija==''New York'' then Politika
if Lokacija==''Maribor'' then Politika
if Lokacija==''Berlin'' and Dogodek==''predavanje'' then Izobraževanje
if Lokacija==''Berlin'' and Dogodek==''koncert'' then Umetnost
```

Glavni problemi pri tej metodi so v določanju pravil, ki jih je lahko veliko ali pa jih je težko določiti oziroma ubesediti.

2.3.1 Metoda končnih avtomatov (Finite state machine)

Metoda končnih avtomatov temelji na računanju verjetnosti prehodov iz enega stanja v drugega. Določeno imamo začetno stanje, abecedo vhodnih podatkov in prenosno funkcijo, ki prevede trenutno stanje v naslednje [19]. Primer takšne metode je v razpoznavanju govora velikokrat uporabljen prikriti Markovov model. Velikokrat pa se prikriti Markovov model uporablja za ločevanje pomembnih delov dokumenta od nepomembnih [8]. Predlagani model dokumenta je prikazan na sliki 2.4.

Na sliki 2.4 vidimo, da razdelimo obravnavani dokument na dva dela: na pomembnega in nepomembnega. Takšno obravnavo dokumenta je možno uporabiti le



Slika 2.4: Zaporedna modela pridobivanja informacije

v nekaterih oblikah dokumentov, kjer je sestava do neke mere vnaprej določena. Za primer lahko vzamemo to magistrsko delo, ki ima takšno sestavo:

- izvleček,
- zahvala,
- uvod,
- pregled metod,
- primer uporabe,
- analiza rezultatov,
- zaključek.

To je tudi običajna oblika magistrske naloge. Vrstni red nekaterih elementov je lahko tudi drugačen (*pregled metod* in *primer uporabe* lahko med seboj zamenjamo), večina pa ima vnaprej določeno lego. Zgoraj omenjeni deli dokumentov se lahko med seboj razlikujejo po besednjaku, ki ga uporabljajo, zato že iz besednjaka posameznega dela ugotovimo, v katerem delu dokumenta se trenutno nahajamo. Sam model dokumenta modeliramo z verjetnostnimi funkcijami, tako da določimo, kakšna je verjetnost določenega dela dokumenta. Verjetnost, da *primeru uporabe* sledi *analiza rezultatov*, je zelo velika, medtem ko je verjetnost, da sledi *izvleček*, dosti manjša.

$$P(R|d) = P(d|R)\frac{P(R)}{P(d)} \tag{2.17}$$

Za določanje verjetnosti posameznih delov dokumenta lahko tako uporabimo metodo naivnega Bayesa. Enačbo te metode smo zapisali v 2.17 in z uporabo modelov za izdelavo delov dokument, ki jih strokovnjak določi kot *pomembne* (ϕ_R) oziroma *nepomembne* (ϕ_i) zapišemo enačbo 2.18.

$$log\frac{P(R|d)}{P(I|d)} = log\left(\frac{P(d|\phi_R)}{P(d|\phi_I)}\frac{P(R)}{P(I)}\right) \tag{2.18}$$

V njej obravnavamo povezavo med verjetnostmi *pomembnih* in *nepomembnih* delov dokumenta in verjetnostjo, da bo naslednji del dokumenta *pomemben* ali *nepomemben*. Preprosteje povedano, kakšna je verjetnost, da recimo določenemu pomembnemu delu dokumenta sledi njegov nepomemben del.

2.3.2 TFIDF

Kratica TFIDF pomeni "term frequency, inverse document frequency". V tej metodi torej obravnavamo pogostost pojavljanja enoličnih izrazov (besede ali besedne zveze) in obrnjeno pogostost dokumenta. Metoda se v literaturi pojavlja dokaj pogosto kot del postopka določanja pripadnosti dokumentov različnim razredom ([23], [17], [13],...). V inverzni frekvenci dokumenta določamo število dokumentov, v katerih se pojavlja določen izraz. Najprej opišimo postopek TFIDF:
Vzemimo učno množico dokumentov. Iz te ustvarimo novo množico, katere člani so enolični izrazi iz vseh učnih dokumentov. V vsakem dokumentu nato preštejemo enolične izraze. Število enoličnih izrazov nato utežimo tako, da za vsakega od izrazov ugotovimo, v koliko dokumentih se pojavlja. Tako dobimo vektor TFIDF. Vse dokumente lahko zapišemo v obliki TFIDF, tako da zapišemo dva podatka:

- za vsak dokument: število enoličnih izrazov (t_{f_i}),
- za vsak izraz: v koliko dokumentih se pojavlja ($d_{f_i}^{tf}$).

Tako naredimo množico vektorjev; po en vektor za vsak dokument v učni množici dokumentov.

V naslednjem koraku določimo vektor TFIDF dokumentu, za katerega želimo ugotoviti, kateremu dokumentu iz učne množice je najbolj podoben. Ker delamo z

vektorji, potrebujemo postopek za primerjavo dveh vektorjev. Ena od takih metod je kosinusna funkcija, ki določa podobnost med dvema vektorjema, tako da izračunamo kosinus med dvema normaliziranima vektorjema.

Zapišimo postopek TFIDF matematično:
imamo množico N dokumentov, ki jo označimo D (enačba 2.19). Nato iz množice dokumentov zapišemo množico enoličnih izrazov X (enačba 2.20). Za vsakega od teh N dokumentov zapišemo, koliko različnih izrazov vsebuje enačba 2.21.

$$D = \{d_1, d_2, ..., d_j, ..., d_N\} \tag{2.19}$$

$$X = \{x_1, x_2, ..., x_j, ..., x_M\} \tag{2.20}$$

Sedaj lahko za vsak dokument iz množice D zapišemo število izrazov, ki so se v njem pojavili. Torej lahko zapišemo enačbo 2.21 za vsak dokument iz množice D.

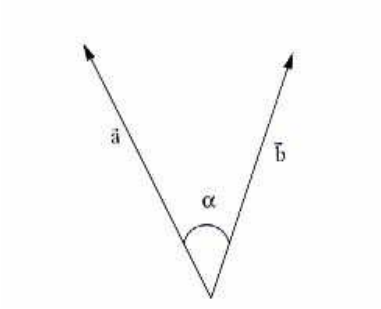
$$d_j^{tf} = \{tf_1, tf_2, ..., tf_M\}; j = 1, 2, 3, ..., N \tag{2.21}$$

Podatke iz enačbe 2.21 sedaj utežimo z inverzno frekvenco dokumenta, ki je določena kot $log(\frac{N}{df_j^{tf}})$ za vsak izraz iz množice X (enačba 2.20). Inverzna frekvenca dokumenta nam pove, v koliko dokumentih se je pojavil določen (enolični) izraz x_i. Tako dobimo končno enačbo 2.22.

$$TFIDF = \{tf_1 log(\frac{N}{df_1^{tf}}), tf_2 log(\frac{N}{df_2^{tf}}), ..., tf_i log(\frac{N}{df_i^{tf}}), ..., tf_M log(\frac{N}{df_M^{tf}})\} \tag{2.22}$$

Množice so zapisane v obliki vektorjev in tako lahko posamezne dokumente učne množice zapišemo v obliki vektorjev dolžine M (torej število enoličnih izrazov v vseh N učnih množicah). Tako smo tudi normalizirali dokumente.

Dokument, za katerega želimo določiti podobnost z enim od učnih dokumentov, "pretvorimo" v obliko TFIDF, torej v vektor, in ga primerjamo z vsakim od vektorjev, ki predstavljajo dokumente učne množice. Za primerjavo lahko uporabimo kosinusno



Slika 2.5: Primer uporabe kosinusne funkcije

funkcijo, ki se največkrat uporablja za primerjavo dveh vektorjev (enačba 2.23).

$$cos(TFIDF_i, TFIDF_j) = \frac{TFIDF_i \cdot TFIDF_j}{||TFIDF_i|| \cdot ||TFIDF_j||} \tag{2.23}$$

S kosinusno funkcijo računamo razmerje sosednih stranic pravokotnega trikotnika, zato jo lahko uporabimo tudi za ugotavljanje podobnosti dveh vektorjev.

Ker imamo opravka z vektorji, ki so lahko različnih dolžin, jih je potrebno normirati (enačba 2.24), tako da imamo razmerje dveh normiranih vektorjev.

$$TFIDF_{norm} = \frac{\overrightarrow{TFIDF}}{||\overrightarrow{TFIDF}||} \tag{2.24}$$

Postopek TFIDF si lahko razložimo z naslednjo predpostavko: predpostavimo, da imamo množico dokumentov D dolžine N. V vsakem od teh dokumentov imamo poljubno število izrazov. Iščemo izraze, ki so značilni le za določeno temo, ki jo opisujejo določeni dokumenti. Ti izrazi se torej pojavljajo le v nekaterih dokumentih (željeno le v dokumentih, ki opisujejo določeno temo). Želimo pa odstraniti izraze, ki se pojavljajo v vseh dokumentih, ker nam le-ti ne pomagajo pri razločevanju dokumentov po različnih temah. Tako za iskanje izrazov v dokumentih uporabimo postopek TF in rezultate utežimo s postopkom IDF, ki zmanjša večji TF, če se izraz pojavi v prevelikem številu dokumentov.

Največkrat pa imamo na razpolahko nekaj razredov, v katere razvrščamo dokumente, in v učni množici več dokumentov, ki sestavljajo določen razred. V tem

primeru moramo za vsak razred določiti centroid. Težišče določimo s preprostim povprečjem preko vseh elementov vektorja (enačba 2.25).

$$C = \frac{1}{|N|} \sum_{TFIDF \in D} TFIDF_i \tag{2.25}$$

Oglejmo si preprost primer uporabe metode TFIDF.

Iz množice časopisnih člankov, ki jih imamo na razpolago, želimo sestaviti časopis, ki bo imel tematsko razporejene članke. Naš časopis naj pokriva naslednje teme:

- notranjo politiko,
- zunanjo politiko,
- šport,
- vreme,
- kulturo,
- tarče.

Na razpolago imamo 5.000 člankov, ki pokrivajo vsa ta področja, ki so bila predhodno ročno razvrščena v posamezne skupine tem. Množico 5.000 člankov razdelimo v dve skupini. V prvi skupini imamo 4.000 člankov in to skupino člankov označimo kot učno množico. Ostalih 1.000 člankov označimo kot testno množico.

V zgornjem spisku vidimo, da bomo članke razvrščali v 6 razredov. Ker tema tega opisa ni predpriprava dokumentov, bomo predpostavili, da so članki že ustrezno pripravljeni za obdelavo in se bomo takoj lotili postopka. Ker imamo v enem razredu več člankov, moramo najprej za vsakega od njih pripraviti svoj vektor. Tako članke normiramo na enotno dolžino. Ko imamo pripravljene vektorje člankov iz enega razreda, izračunamo centroid tega razreda. Centroid izračunamo z enačbo 2.25, tako da izračunamo povprečje elementov vseh vektorjev določenega razreda. Postopek računanja cetroida ponovimo za vse razrede člankov in kot rezultat dobimo 6 vektorjev (centroidov), ki predstavljajo neke vrste splošni model članka za določeni razred.

Tako smo izvedli postopek učenja našega sistema. Sedaj pa sistem preizkusimo s testnimi članki, ki smo jih na začetku pripravili iz množice 5.000 člankov. Tudi za postopek preizkušanja je potrebno za vsak članek pripraviti vektor, nato pa vsak vektor iz preizkusne množice primerjamo s centroidi (ki so tudi vektorji) posameznih razredov. Za primerjavo teh dveh vektorjev uporabimo kosinusno funkcijo, katere rezultat je realno število. Preizkusni članek razvrstimo v tisti razred, ki da v postopku primerjave centroida razreda in vektorja preizkusnega članka največjo vrednost kosinusne funkcije. V preizkusnem postopku torej primerjamo posamezne članke iz preizkusne množice s centroidi posameznih razredov. Nato razvrstimo preizkusni članek v tisti razred, katerega centroid se najbolj ujema s preizkusnim člankom. Tako smo preizkusili sistem razvrščanja po postopku TFIDF. Prednost sistema je v računanju centroidov, kar nam omogoča, da shranimo le toliko vektorjev (centroidov), kot je razredov.

2.4 Nevronske mreže

V osnovi imamo dve vrsti nevronskih mrež:

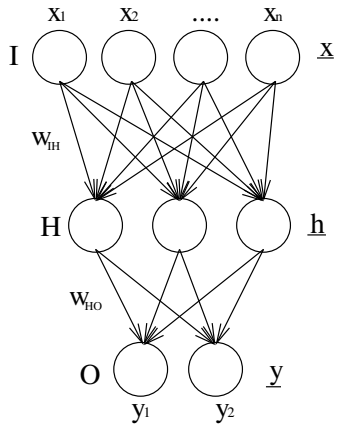
- nevronsko mrežo z nadzorovanim učenjem in
- nevronsko mrežo s samodejnim učenjem.

Kot že ime pove, je razlika med obema vrstama v načinu učenja. Če uporabimo nevronsko mrežo z nadzorovanim učenjem, moramo pripraviti učno množico podatkov in jih uporabiti za določitev vrednosti uteži, ki povezujejo vozlišča na različnih ravneh (vhodna, skrita in izhodna). Pri uporabi nevronske mreže s samodejnim učenjem pa učne množice ni potrebno pripravljati, saj nevronska mreža spreminja svojo "obliko" glede na vhodne podatke in združuje podobne vhodne podatke.

2.4.1 Nevronska mreža z nadzorovanim učenjem

Najprej si oglejmo preprost primer nevronske mreže (NN s povratno zanko) (sl. 2.4).

Naš primer nevronske mreže ima vhodni vektor z n nevroni, skriti vektor s tremi nevroni in izhodni vektor z dvema nevronoma. Metoda temelji na postopku učenja.



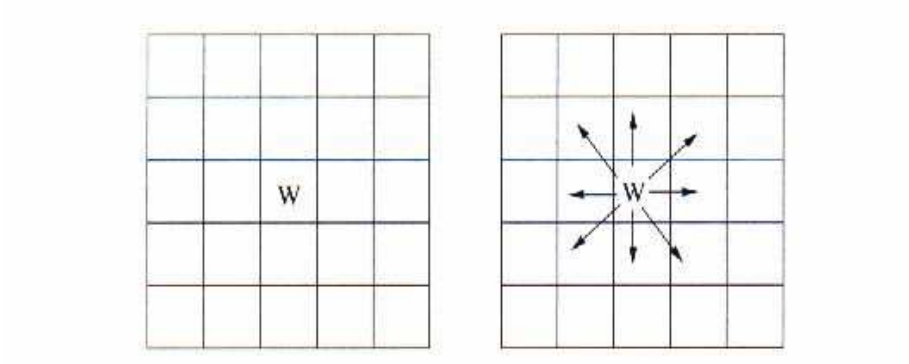
Slika 2.6: Nevronska mreža s povratnim učenjem

Določiti moramo željene vrednosti nevronov izhodnega vektorja pri določenih vrednostih nevronov vhodnega vektorja. Postopek učenja je sledeč:

- določimo vrednosti uteži in pragov (povezava med posameznimi nevroni ravni). Ta vrednost je lahko naključna (običajno na začetku) ali določena s predhodnim učenjem,
- na vhod postavimo vektor vhodnih elementov in izračunamo izhodni vektor,
- na podlagi napake med želeno in dobljeno vrednostjo izhodnega vektorja določimo nove vrednosti uteži povezav med vozlišči izhodnega in skritega vektorja ter nato nove vrednosti uteži povezav med vozlišči skritega in vhodnega vektorja,
- postopek ponavljamo za vse vhodne vektorje,
- postopek je končan, ko dosežemo vnaprej določeno število korakov ali ko je napaka med želenim in dobljenim izhodnim vektorjem manjša od vnaprej določene vrednosti ε.

Nevronska mreža je torej sestavljena iz nevronov [3], ki imajo na vsakem nivoju funkcijo:

$$x_j^l = \theta(\sum_{i=0}^{d^{l-1}} w_{ij}^l x_i^{l-1}) \tag{2.26}$$



Slika 2.7: Kohonenova nevronska mreža

V enačbi (2.26) je θ:

$$\theta(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}} \tag{2.27}$$

Funkcija napake na izhodu oziroma za vsak nevron pa je:

$$\varepsilon = (g(x_n - y_n))^2 \tag{2.28}$$

Enačba napake (2.28) nam pove, kako dobro se učne množice izhodnih vrednosti prilegajo izračunanim izhodnim vrednostim. To napako želimo minimalizirati. Izračunamo torej gradient napake, ki ga nato uporabimo za spremembo uteži.

$$w^l_{ij,novo} = w^l_{ij,staro} - \sigma \frac{\delta\varepsilon}{\sigma w^l_{ij}} \tag{2.29}$$

V enačbi (2.29) pomeni σ hitrost učenja oziroma spreminjanja uteži. Ta postopek ponovimo tudi za uteži, ki povezujejo vhodno in skrito raven.

Celoten postopek lahko uporabimo za pripravo samodejnega razvrščevalnika. Na vhod lahko pripeljemo normalizirane dokumente in na izhodu dobimo pripadnost določenemu centroidu. Seveda je potrebna predpriprava dokumenta, kar pa si bomo pogledali v poglavlju *Izvedba samodejnega razvrščanja.* V nevronski mreži s povratnim učenjem lahko spreminjamo tak*ó* število elementov skrite ravni, kot tudi število skritih ravni. V praksi obstaja veliko postopkov za določanje teh elementov, medtem ko je za uporabo samodejnega razvrščanja število vhodnih in izhodnih nevronov poznano.

2.4.2 Nevronska mreža s samodejnim učenjem

Primer te mreže je Kohonenova nevronska mreža, ki se uči s spreminjanjem sestave. Kohonenova nevronska mreža [16] se uči tako, da išče nevrone (vozlišča), ki so najbolj podobni vhodnemu podatku. Ko mreža najde tak nevron, spremeni le-tega in tudi vse sosednje nevrone (sl. 2.7). Na sliki 2.7 pomeni W zmagoviti nevron, torej nevron, ki se najbolj ujema z vhodnimi podatki. Zmagovitemu nevronu spremenimo utež in nato spremenimo utež sosednjim nevronom z uporabo enačbe 2.29. Zmagoviti nevron torej vpliva tudi na uteži sosednjih nevronov. Enačba 2.30 določa moč vpliva zmagovitega nevrona na sosednje nevrone. Oglejmo si kakšen je postopek učenja Kohonenove mreže:

- vsak nevron Kohonenove mreže prejme celotno kopijo vhodnih podatkov,
- poiščemo zmagoviti nevron (nevron, ki ima najmanjšo razdaljo $d_j = \frac{1}{2}\sum_{i=1}^{m}(x_i - w_{ij})^2$),
- zmagovitemu nevronu in njegovim sosedom spremenimo uteži po enačbi 2.30
- ponovimo zgornje korake za vsak vhodni podatek,
- ponavljamo prejšnji korak, dokler niso obdelani vsi sklopi vhodnih podatkov,
- ponavljamo prejšnji korak n-krat (n je vnaprej določen).

$$W_{ij}(t+1) = W_{ij}(t) + \alpha(t) \cdot \gamma(t) \cdot [X_i - W_{ij}(t)]$$
$$\gamma(t) = e^{-\frac{1}{2}\cdot(\frac{r_{ij}}{\sigma(t)})^2} \tag{2.30}$$

V enačbi 2.30 je α hitrost učenja uteži in upada s časom. Začetna vrednost za α je med 0 in 1, r_{ij} pa je razdalja med zmagovito celico in celico, ki ji spreminjamo utež (torej polmer kroga celic, ki jim spreminjamo uteži). σ je polmer soseščine, ki upada s časom.

2.5 Komite razvrščevalnikov

Komite razvrščevalnikov temelji na preprosti predpostavki, da več "glav" več ve. Tako uporabimo večje število razvrščevalnikov, ki obravnavajo isti problem, in nato na nek način izberemo najboljši rezultat oziroma združimo rezultate. Postopkov za združevanje rezultatov je veliko. Najpreprostejši je *glasovanje z večino*, kjer imamo k razvrščevalnikov in je pravilna tista razvrstitev, ki jo določi večina (vsaj $\frac{k+1}{2}$) razvrščevalnikov. Seveda mora biti k liho število. Lahko pa uporabimo tudi *uteženo linearno kombinacijo*, kjer je prispevek posameznih razvrščevalnikov k skupni vsoti utežen. Utež w_j posameznega razvrščevalnika pove, kako pomemben je njegov prispevek oziroma kako kakovosten je prispevek razvrščevalnika k skupnemu rezultatu. *Dinamičnina izbira razvrščevalnika* izbere med razvrščevalniki v komiteju tistega, ki je najbolj učinkovit pri l elementih testne množice in z njegovimi rezultati nauči ostale razvrščevalnike, da dajo enak rezultat. *Adaptivno kombiniranje razvrščevalnikov* pa sešteva razvrstitve posameznih razvrščevalnikov, tako da uteži posamezne rezultate glede na njihovo uspešnost v zaporedju l elementov testne množice.

Komite razvrščevalnikov je v literaturi prikazan z različno uspešnostjo [25], nekateri so imeli več uspeha kot drugi. Velikokrat je bil problem premajhen korpus besedila.

2.6 Dvigovanje

Metodo *dvigovanja* (angl. boosting) so uvedli Schapire et al [24]. Imamo komite k razvrščevalnikov z enako metodo. Ti razvrščevalniki pa ne delujejo vzporedno in neodvisno, temveč jih uporabljamo zaporedno. Tako uporabimo posamezne razvrščevalnike zaporedno preko podatkov in poskušamo izboljšati tiste primere, ki imajo najslabše rezultate. Tako v postopku učenja vsakemu paru (d_j,c_i) (dokument, razred) dodelimo utež w_{ij}^t, ki nam predstavlja, kako težko je dobiti pravi par za vsak razvrščevalnik od 1 do t-1. Te uteži nato uporabimo za "prilagoditev" razvrščevalnika t, ki ga pripravimo tako, da pravilno razvršča pare z večjo utežjo, torej pare, pri katerih je razvrščanje neuspešno. Ta razvrščevalnik nato uporabimo z elementi testne množice

in dobljene uteži w_{ij}^t popravimo v w_{ij}^{t+1}. Postopek ponavljamo, dokler niso uspešni vsi razvrščevalniki. Ko zgradimo vse k razvrščevalnike, uporabimo pravilo linearne kombinacije za končni komite razvrščevalnikov.

2.7 Metode testiranja uspešnosti razvrščanja

2.7.1 Učinkovitost razvrščanja

Učinkovitost razvrščanja običajno določimo z natančnostjo (τ) in s ponovnim priklicem (σ). Natančnost razreda c_i je določena z verjetnostjo [25] v enačbi 2.31, kar pomeni:
če je nakljućen dokument d_x, ki spada v razred c_i, razvrščen v ta razred, je razvrstitev pravilna. Ali drugače, če je vsak dokument razvrščen v pravi razred, je natančnost 100-odstotna ($P(\phi(d_x, c_i)) = 100\%$). V enačbah 2.31 in 2.32 pomeni T oznaka za uspeh (angl. TRUE), medtem ko je oznaka $\check{\phi}$ približek pravi vrednosti ϕ.

$$P(\check{\phi}(d_x, c_i)) = T|\phi((d_x, c_i) = T) \tag{2.31}$$

Enaka enačba kot 2.31 velja tudi za ponovni priklic, kjer pa obravnavamo c_i, tako ima enačba obliko 2.32. Tukaj se vprašamo, kakšna je verjetnost, da določen že razvrščen dokument ponovno prikličemo. Če lahko vsak dokument, ki že razvrščen, ponovno prikličemo, je verjetnost za ponovni priklic 100-odstotna.

$$P(\phi(d_x, c_i(\sigma_i))) = T|\check{\phi}(d_x, c_i(\sigma_i) = T) \tag{2.32}$$

Ker vrednosti natančnosti in priklica obravnavamo po posameznih razredih, si moramo pogledati tudi, kako takšne posamezne razrede posplošimo. Poudariti je potrebno, da sta vrednosti τ_i in σ_i subjektivni verjetnosti uporabnika, da bo sistem razvrstil obravnavani dokument v določeni razred. Torej gre za *pričakovano* verjetnost razvrščanja. Tako lahko ti vrednosti ocenimo z uporabo pojmov FP_i (napačno pozitivni), FN_i

(napačno negativni), TP_i (pravilno pozitivni), TN_i (pravilno negativni).

$$\hat{\tau_i} = \frac{TP_i}{TP_i + FP_i} \tag{2.33}$$

$$\hat{\sigma_i} = \frac{TP_i}{TP_i + FN_i} \tag{2.34}$$

V enačbah 2.33 in 2.34 iščemo približek za določitev natančnosti in odpoklica po posameznih razredih. Za obravnavno vseh odločitev pa imamo dva postopka:

- mikropovprečje, kjer dobimo τ in σ s seštevanjem posameznih odločitev (enačbi 2.33 in 2.34),
- makropovprečje, kjer natančnost in odpoklic najprej obravnavamo "lokalno" za vsak razred in nato "globalno", tako da izračunamo povprečje različnih razredov (enačbi 2.35 in 2.37).

$$\hat{\tau^{\mu}} = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FP_i)} \tag{2.35}$$

$$\hat{\sigma^{\mu}} = \frac{TP}{TP + FN} = \frac{\sum_{u=1}^{|C|} TP_i}{\sum_{i=1}|C|(TP_i + FN_i)} \tag{2.36}$$

$$\hat{\tau^{M}} = \frac{\sum_{i=1}^{|C|} \hat{\tau_i}}{|C|}; \hat{\sigma^{M}} = \frac{\sum_{i=1}^{|C|} \hat{\sigma_i}}{|C|} \tag{2.37}$$

Oba razreda se lahko razlikujeta v rezultatih, še posebej če obravnavamo metodo, ki razvršča v razrede z malo pozitivnimi učnimi dokumenti. Postopek bo bolje utežen, če uporabimo makropovprečje. Za določanje učinkovitosti razvrščanja lahko uporabimo tudi določanje natančnosti in napake 2.38.

$$\hat{A} = \frac{TP + TN}{TP + TN + FP + FN} \hat{E} = \frac{FP + FN}{TP + TN + FP + FN} = 1 - \hat{A} \tag{2.38}$$

Vendar pa metoda natančnosti in napake ni velikokrat uporabljana, ker, kot je ugotovil Yang, naredijo običajno velike vrednosti imenovalca metodo neobčutljivo na spremembe števil pravilne razvrstitve (TP+TN).

2.8 Zaključek pregleda

Področje samodejnega razvrščanja besedil sem poskušal podati kar se da pregledno in ravno zato sem se odločil za način formalizacije opisa posameznih metod. Metode sem rezdelil v več skupin in jih poskušal predstaviti ter primerjati med seboj. Trudil sem se opisati predvsem metode, ki so največkrat prikazane v literaturi in to v njihovi osnovni obliki, ker se v literaturi pojavljajo tudi prikazane metode z nekaj spremembami. Poleg tega so spremembe predvsem zato, da se metoda prilagodi posameznemu problemu. Pojavi se sicer vprašanje ali je tako pravilno ali ne. Odgovora sta dva:

- prilagajanje metode posameznemu problemu omogoča njegovo reševanje z metodo, ki mu je pisana na kožo in s tem lahko metodo izboljšamo,
- če prilagodimo metodo posameznemu problemu, dobimo na koncu veliko razdrobljenost metod, ki so same sebi namen oziroma ne formalizirajo izdelave splošne metode, ki bi bila uporabna na večjem številu problemov. Težava je tudi pri izbiri ustrezne metode za obravnavo problema.

Zanimivo je, da se še vedno uporabljajo metode, ki so bile uvedene v šestdesetih letih in nič ne kaže, da se bo to v kratkem spremenilo. Sicer je res, da je najbolje staviti na konja, ki zmaguje. Problem pa je, da ta konj ne zmaguje ravno pogosto, oziroma da so rezultati včasih tudi neuporabni za praktično uporabo. Ena od možnosti za iskanje popolnoma novih metod samodejnega razvrščanja je interdisciplinarnost. Na drugih področjih namreč obstajajo metode, ki še niso bile uporabljane za postopek samodejnega razvrščanja, vendar bi se, če bi bile poznane na tem področju, najbrž zelo dobro obnesle. Kot primer lahko navedem metodo podpornih vektorjev, ki je trenutno "vroča" tema na področju samodejnega razvrščanja in je že nekaj časa znana na področju statistike. Seveda samo zanimivost metode ni dovolj za njeno uspešno uporabo, kar nam dokazuje mnogo metod, ki so zasijale le za trenutek. Morda pa je razlika pri metodi podpornih vektorjev ta, da je že dobro poznana na drugih področjih, in da ima močno teoretično podlago. Hkrati je sama priprava metode formalizirana, tako enostavneje opisljiva in tudi do neke mere laže uporabna (predvsem v svoji formalizaciji). Kako se bo metoda v resnici obnesla, bo pokazal čas. Zanimivo je tudi

kombiniranje več metod za doseganje boljših rezultatov. Tako sem v svojem postopku uporabil dve metodi: TFIDF in k najbljižjih sosedov. S prvo sem določil centroide, z drugo pa iskal najprimernejše ključne besed.

POGLAVJE 3
IZVEDBA SAMODEJNEGA RAZVRŠČANJA

3.1 Opis problema

Imamo bazo izvlečkov Biomedicina Slovenica, ki jim želimo vnaprej določiti oznake. Vsak izvleček ima lahko poljubno število oznak, ki so zbrane v spisku z imenom MeSH (Medical Subject Heading) in so razdeljene na več ravni. Ogledali si bomo torej uspešnost razvrščanja po posameznih ravneh.

3.2 Biomedicina Slovenica

Biomedicina Slovenica je slovenska nacionalna računalniška bibliografija za področje biomedicine (`http://www.mf.uni-lj.si/ibmi/info-viri/index.html`), ki obsega medicino, stomatologijo, veterinarstvo, biologijo in njihove bazične vede. Baza se gradi že več kot 20 let in vključuje literaturo, objavljeno v Sloveniji, in dela slovenskih avtorjev, objavljena v tujini, ki morajo zadoščati določenim kriterijem (objavljeno delo, slovenski avtor, ne sme biti recenzija, izvleček, poročilo s kongresa, ...). Za izgradnjo zbirke so bili povzeti MEDLARSovi (Medical Literature Analysis and Retrieval System) standardi, ki jih uporablja svetovna medicinska zbirka MEDLINE. Vsebinska obdelava poteka z angleškimi ključnimi besedami iz hierarhično urejenega tezavra MeSH. Vsak zapis vsebuje ključne besede 1 do 5, ki opisujejo najpomembnejše vidike članka, in ključne besede 2 do 12 za manj pomembne teme. Pomen ključnih besed pa lahko natančneje opišemo s kvalifikatorji, ki jim po potrebi dodamo dodatne izraze za koncepte, ki jih MeSH ne pokriva.

Biomedicina Slovenica je dostopna tudi preko pregledovalnika (sl. 3.1) in teče na Oraclu. V osnovi služi za vodenje bibliografije za raziskovalce, ki delajo na področju medicine oziroma na področjih, ki jih Biomedicina Slovenica pokriva.

Slika 3.1: Internetni dostop do BS

Primer zapisa iz Biomedicine Slovenice v XML obliki:


```
<record>
<id>26620</id>
<title>Bronhoalveolarna lava‘a v diagnostiki in zdravljenju intersticijskih
plju˜nih bolezni</title>
<translated_title>Bronchoalveolar lavage in diagnosis and treatment of
interstitial lung diseases</translated_title>
<eng_abstract>Background. Bronchoalveolar lavage (BAL) has been a valuable
diagnostic method in interstitial lung diseases for more than 15 years
enabling both the differential diagnosis as well as the follow-up. Only
occasionaly BAL offers a specific diagnosis while more frequently only
one parameter of disease. Methods. The data on clinical value of BAL,
differencies in differential cellular count and lymphocytes subtypes as
well as on morphology of alveolar macrophages in interstitial lung diseases
are presented. Experiences form the Institute of Respiratory Diseases Golnik
have been included since 1982. Additionally to cytological the new immunological
(cytocines) analyses of BAL supernatant have been recently introduced.
Conclusions. BAL is a useful sensitive and rarely specific method for diagnosis
of interstitial lung diseases. It offers signs on the immunologic type and
activity of alveolitis. Clinical pulmonologists are comparing BAL with perpheral
blood cell analysis.</eng_abstract>
<slo_abstract>Izhodi{˜a. Bronhoalveolarna lava‘a (BAL) je ‘e nad 15 let pomembna
diagnosti˜na metoda v pulmologiji. Omogo˜a predvsem diferencialno diagnostiko
intersticijskih plju˜nih bolezni in spreminjanje aktivnosti med zdravljenjem.
Posami˜ je BAL celo diagnosti˜na, ve˜krat pa le eden od parametrov diagnoze.
Metode. V obliki preglednega poro˜ila so zbrani podatki o klini˜ni vrednosti
BAL, o razlikah v diferencialnem razmerju celic in vrstah limfocitov v njej
pri posamezni bolezni, dodatno pa {e o morfologiji alveolarnih makrofagov.
Med podatke iz literature so vpletene tudi doma˜e izku{nje, med njimi nekatere
```

‘e objavljene, saj v In{titutu za plju˜ne bolezni in TBC Golnik izvajamo BAL ‘e od leta 1982 dalje. Poleg analize celic se uvajajo tudi imunolo{ke analize teko˜inskega dela BAL. Zaklju˜ki. BAL je uporabno ob˜utljiva in posami˜ celo specifi˜na metoda v diagnostiki intersticijskih plju˜nih bolezni. Med zdravljenjem posreduje podatke o aktivnosti in imunolo{kem tipu alveolitisa. Ugledni pulmologi jo po pomenu vzporejajo s periferno krvno sliko.</slo_abstract>
<descriptors1>
<des>LUNG DISEASES, INTERSTITIAL</des>
<qual>diagnosis/therapy</qual>
</descriptors1>
<descriptors1>
<des>BRONCHOALVEOLAR LAVAGE</des>
</descriptors1>
<descriptors2>
<des>BRONCHOALVEOLAR LAVAGE FLUID</des>
</descriptors2>
</record>

3.3 Medical Subject Heading

Mesh (`http://www.nlm.nih.gov/mesh/meshhome.html`) je tezaver, ki ga vzdržuje National Library of Medicine iz ZDA. Ključne besede so urejene hierarhično od najbolj splošnih (*Anatomy*, *Mental Disorders*...) do podrobnejših opisov. Razvrščene so v 11 ravni in jih je 22.568 (sl. 3.3).

3.4 Samodejno dodeljevanje ključnih besed

V postopku priprave sistem za samodejno določanje ključnih besed je potrebno sistem najprej naučiti. Iz dostopnih podatkov je potrebno določiti učno in testno množico. Postopek za to je lahko intuitiven ali formalen. Če se problema lotimo intuitivno, potem z izbrano predpostavko določimo mejo med učno in testno množico. Moramo

1. ⊟ Anatomy [A]
   - Body Regions [A01] +
   - Musculoskeletal System [A02] +
   - Digestive System [A03] +
   - Respiratory System [A04] +
   - Urogenital System [A05] +
   - Endocrine System [A06] +
   - Cardiovascular System [A07] +
   - Nervous System [A08] +
   - Sense Organs [A09] +
   - Tissues [A10] +
   - Cells [A11] +
   - Fluids and Secretions [A12] +
   - Animal Structures [A13] +
   - Stomatognathic System [A14] +
   - Hemic and Immune Systems [A15] +
   - Embryonic Structures [A16] +
   - Integumentary System [A17] +
2. ⊞ Organisms [B]
3. ⊞ Diseases [C]
4. ⊞ Chemicals and Drugs [D]
5. ⊞ Analytical, Diagnostic and Therapeutic Techniques and Equipment [E]
6. ⊞ Psychiatry and Psychology [F]
7. ⊞ Biological Sciences [G]
8. ⊞ Physical Sciences [H]
9. ⊞ Anthropology, Education, Sociology and Social Phenomena [I]
10. ⊞ Technology and Food and Beverages [J]
11. ⊞ Humanities [K]
12. ⊞ Information Science [L]
13. ⊞ Persons [M]
14. ⊞ Health Care [N]
15. ⊞ Geographic Locations [Z]

Slika 3.2: Izgled MeSH-a (prva in druga raven)

pa vedeti, da je bolje imeti čim večjo učno množico, saj lahko tako dosežemo boljše učenje sistema. Obstajajo tudi drugi pristopi, kot na primer prečno ovrednotenje preko podatko velikosti ν (ang: ν-fold cross validation). Največkrat se uporablja prečno ovrednotenje preko podatkov velikosti 10 (ang: 10-fold cross validation). Ta postopek se uporablja na naslednji način:

- množico dostopnih podatkov razdeli na deset delov,
- vzemi desetino, podatkov, kar je testna množica, ostalo pa učna,
- preizkusi sistem,
- gornja dva koraka ponavljaj za vse desetine podatkov,
- rezultat testa je povprečje rezultatov preizkusa sistema.

Tako preizkusimo postopek preko vseh možnih oblik podatkov.

3.4.1 Kako pripraviti učno in testno množico

Opis postopka

Iz baze Biomedicina Slovenica sem izbral takšne članke, ki so imeli tako slovenske kot angleške izvlečke. Angleških nisem uporabil in tako sem dobil 1364 zapisov, od teh sem jih 1033 uporabil kot učno množico in ostale (331) kot testno množico. Pri vsakem zapisu sem upošteval le ključno besedo 1, ki je tesno povezana z vsebino dokumenta. Ogledali si bomo naslednje:

- vpliv izbire ravni ključne besede iz MeSH-a na uspešnost razvrščanja,
- potrditev predpostavke, da imajo besede iz naslova večji vpliv na razvrščanje.

Pred začetkom samodejnega določanja ključnih besed posameznim izvlečkom moramo pripraviti podatke na obdelavo. Postopek predobdelave podatkov je sestavljen iz naslednjih korakov:

- odstrani nedovoljene znake (ločila, številke ...),

- odstrani nepolnopomenske besede,
- prekodiraj besedila v standard ISO 8859-2 (Latin2),
- krni besede.

Odstranitev nedovoljenih znakov in nepolnopomenskih besed

Postopek odstranjevanja nedovoljenih znakov in nepolnopomenskih besed sem združil, ker sta dokaj preprosta in temeljita na spisku besed oziroma znakov, ki jih je potrebno izločiti. Nedovoljeni znaki so v našem primeru:

```
locila=('"','-","!",".",",",";",":","(",")","'","%"," ","*","+","=","/","#")
```

nepolnopomenske besede pa sem določil na več načinov:

- prvih 10 najbolj pogosto pojavljanih besed v slovenskih besedilih (`http://bos.zrc-sazu.si/s\_beseda.html`),
- besede, ki se pojavljajo enakomerno preko vseh dokumentov učne množice.

Tako sem sestavil spisek besed, ki je naveden v Prilogi A.

Podrobneje si bomo ogledali zadnja dva koraka. Besede, ki se pojavljajo enakomerno preko vseh dokumentov učne množice, so besede, ki nam pri samodejnem razvrščanju ne prinesejo "dodane vrednosti". V dokumentih namreč iščemo ključne besede, značilne le za dokumente, ki pripadajo določeni množici (centroid, imajo enako ključno besedo). V idealnem primeru bi imeli dokumente med seboj strogo ločene oziroma bi našli ključne besede, ki so značilne le za določen centroid. Da bi se čimbolj približali idealnemu primeru, iz dokumentov potemtakem izločamo besede, ki se pojavljajo v vseh dokumentih. Seveda bi te besede lahko izločili tudi po spisku 10 najpogosteje pojavljajočih se besed, vendar pa tema dokumentov določa izrazoslovje oziroma besednjak v dokumentih. Ker imamo opravka z medicinskimi dokumenti, lahko pričakujemo, da se bodo zelo pogosto pojavljale besede, ki so značilne za medicino, npr.:

- zdravnik,

- bolnik,
- bolezen,
- bolnica,
- ,...

Te besede se v javni rabi ne uporabljajo tako pogosto, da bi jih lahko uvrstili v prvo skupino (10 najpogosteje pojavljajočih se besed). Če bi želeli dokumente razvrščati v področja, ki jih opisujejo, bi besede, ki so značilne predvsem za medicino, uporabili kot ključne besede za naš razvrščevalnik. Izbor nepolnopomenskih besed je torej zelo odvisen od področja uporabe in zato ni mogoče narediti "absolutnega" seznama nepolnopomenskih besed. Prekodiranje v ISO standard 8859-2 je bilo potrebno, ker je bilo besedilo zapisano v starem formatu YUASCII, ki pa ni bil primeren za nadaljno obravnavo (teževe pri urejanju, krnilnik je bil narejen za ISO 8859-2).

3.4.2 Krnjenje

Krnjenje je postopek normalizacije besed, ki so po skupnem korenu pomensko sorodne. Za krnjenje ni nujna povezava s slovnico, tako da so lahko krni drugačni, kot so koreni besed. Leta 1968 je Salton ugotovil, da z uporabo krnjenja ključnih besed dosegamo enake ali boljše rezultate kot pri izrazih, dobljenih z ročnim indeksiranjem (citiran vir:[21]). Frakes in Baeza-Yates [11] sta razdelila krnilnike v več skupin:

- različnost končnic,
- krnilnik n-teric,
- krnilnik priponk.

Oglejmo si opise posameznih krnilnikov.

Različnost končnic

Gre za delo, ki sta ga začela Hafer in Weiss leta 1974 (citiran vir: [21]) in temelji na določanju krna glede na distribucijo fonemov v obsežnejšem govorjenem korpusu. Ker

pa želimo določati krne v pisanem besedilu, uporabimo namesto fonov črke. Haffer in Weiss sta opisala metodo takole:
Naj bo α beseda dolžine n, pri tem velja, da je α_i i-to mesto v besedi α. D naj bo korpus besed in D_α, del korpusa D, ki vsebuje tiste besede, katerih prve i črke ustrezajo α_i. Število različnih končnic α_i, označeno kot S_{α_i}, je določeno kot število enoličnih črk, ki zasedajo i+prvo mesto besed v D_{α_i}. Testna beseda dolžine n ima n različnih končnic S_{α_i}, S_{α_2},...,S_{α_n}. Metoda je uporabna v korpusu z velikim številom besed. Ko enkrat določimo različnost končnic, to informacijo uporabimo za razčlenitev besed. Hafer in Weiss sta opisala štiri načine za ta postopek:

- izbira *metode reza*, kjer vnaprej izberemo pragovno število l in ko imamo možnih l različnih končnic, naredimo rez. Problem pri tem postopku je, da lahko izberemo premajhno vrednost reza in dobimo prekratek krn ali preveliko in izgubimo pravilni krn;

- z *metodo vrh in ravnina*, kjer je rez narejen, ko različnost končnic na določenem znaku preseže različnost končnic znaka pred njim in naslednjega znaka. Tako se izognemo vnaprejšnjemu določanju vrednosti reza,

- z *metodo celotne besede* je rez narejen, ko se konča del besede, ki predstavlja celotno besedo v korpusu;

- z *metodo entropije* določimo rez tako, da upoštevamo porazdelitev črk različnih končnic. Poglejmo, kako deluje ta metoda:

$|D_{\alpha_i}|$ naj bo število besed v korpusu, ki se začnejo z zaporedjem črk α in imajo dolžino i. $|D_{\alpha_{ij}}|$ pa naj bo število besed v D_{α_i} s končnico j. $\frac{|D_{\alpha_{ij}}|}{|D_{\alpha_i}|}$ določa verjetnost, da ima član D_{α_i} končnico j. Entropijo $|D_{\alpha_i}|$ pa določa enačba 3.1.

$$H_{\alpha_i} = \sum_{p=1}^{26} -\frac{|D_{\alpha_{ij}}|}{D_{\alpha_i}} log_2 \frac{|D_{\alpha_{ij}}|}{|D_{\alpha_i}|} \tag{3.1}$$

Z uporabo te enačbe lahko določimo skupek entropij za določeno besedo. Tak*ó* lahko določimo vrednost reza in krn je določen, ko je dosežena vrednost reza.

Hafer in Weiss sta preizkusila vse omenjene metode in ugotovila, da nobena ni popolna, vendar pa je s kombinacijo metod mogoče doseči zadovoljive rezultate. Oglejmo si primer z uporabo krajšega korpusa: preizkusna beseda: lopatanje
preizkusni korpus: branje, kramp, lopatanje, lopatah, lopatal, lopata, lopatama, lopa, lopov

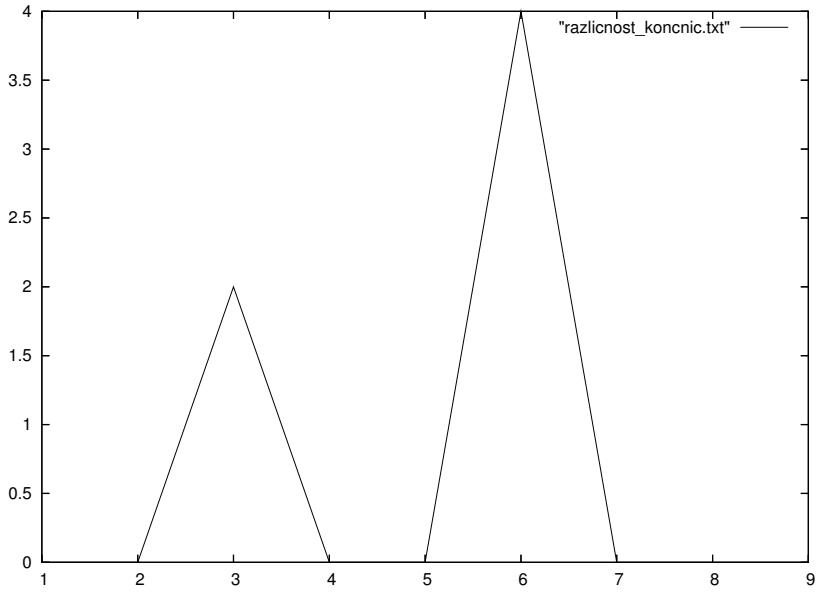
predpona	različnost nadaljevanja	črke
l	1	o
lo	1	p
lop	2	a,o
lopa	1	t
lopat	1	a
lopata	4	n,h,l,m
lopatan	1	j
lopatanj	1	e
lopatanje	1	∅

V diagramu 3.3 vidimo, da je prelom narejen pri četrti črki. Sklepamo lahko torej, da je "lopat" prvi segment besede "lopatanje" in "anje" drugi segment. Uporabili smo metodo celotne besede. Tudi metoda *vrh in ravnina* bi dala enak rezultat. Poudariti pa je potrebno, da Hafer in Weiss takšne osnovne besede še nista proglasila za krn, ampak sta določila pravilo: če se prvi segment pojavi v največ 12 besedah korpusa, je le-ta krn. V nasprotnem primeru je krn drugi segment.

Zgornja trditev temelji na naslednji predpostavki; če je segment v več kot dvanajstih besedah, je najverjetneje predpona in je tako praiv krn drugi segment. Pri tem je potrebno povedati, da je ta predpostavka postavljena za angleščino in jo je potrebno za slovenščino predhodno preveriti.

Krnilnik n-teric

Adamson in Boreham sta določila metodo zlivanja gesel, ki temelji na metodi deljenega diagrama. Diagrami so zaporedne črke. Metoda je dobila ime po možnosti uporabe trojčka ali n-teric. Hkrati moramo vedeti, da ta metoda v bistvu ni metoda



Slika 3.3: Primer uporabe metode različnosti končnic

krnjenja, ker rezultat metode ni krn, ampak gre le za statistiko diagramov v različnih besedah. Pa si poglejmo primer:

- statistika ⇒ st ta at ti is st ti ik ka ⇒ (enolični diagrami) st ta at ti is ik ka
- statistično ⇒ st ta at ti is st ti ič čn no ⇒ (enolični diagrami) st ta at ti is ič čn no

Ko določimo enolične diagrame, preštejmo, koliko je ujemajočih diagramov. Nato lahko z uporabo mere enakosti izračunamo podobnost obeh besed. Adamson in Boreham sta vzela za primer izračun podobnosti z uporabo Dice-ovega koeficienta (enačba 3.2).

$$S = \frac{2C}{A + B} \tag{3.2}$$

V enačbi 3.2 je A število enoličnih diagramov v prvi besedi, B število enoličnih diagramov v drugi besedi in C število enoličnih diagramov, ki so v obeh besedah. Za naš primer je Dice-ov koeficient (glej enačbo 3.3) $\frac{2}{3}$.

$$S = \frac{2 \cdot 5}{7 + 8} = \frac{2}{3} \tag{3.3}$$

Primerjavo naredimo za vse pare besed v besedilu in naredimo matriko parov izrazov (besed) v besedilu. Ko je matrika podobnosti narejena, določimo podobne izraze z uporabo algoritmov za določanje grozdov (clustering).

Krnilnik priponk

Krnilnik priponk je preprosta metoda zaporednih odstranjevanj končnic in predpon. Najpreprostejši primer te metode je odstranjevanje končnic, ki določajo množinsko obliko. To je uporabno predvsem v angleščini (odstranitev končnice -s). V tem postopku gre največkrat za ponavljalne krnilnike najdaljšega ujemanja, ki odstranjujejo najdaljšo možno končnico. Postopek ponavljamo, dokler ni več znakov, ki bi jih odstranili. Na koncu metode lahko dobimo nepravilno zlite krne zato, je potrebno uporabiti še tehniko rekodiranja ali delnega ujemanja. Rekodiranje je kontekstno občutljiv postopek pretvorbe besed, kot naprimer [21] krnjenje besede želodec, ki ji odstranimo končnico -ec in dobimo želod. Beseda želod pa je krn besede želod in ima popolnoma drug pomen. Zato je potrebno krn besede želodec še rekodirati iz oblike želod v obliko želodc. Enega bolj znanih angleških krnilnikov je izdelal dr. Martin Porter. Ta krnilnik spada v skupino krnilnikov priponk in je sestavljen iz pravil pogoj/dejanje. Prvi slovenski krnilnik je izdelal dr. Jure Dimec z Inštituta za Biomedicinsko informatiko [21], vendar je bolj znan krnilnik, ki sta ga izdelala Popović in Willet leta 1992 [22]. Glede krnilnikov se krešejo mnenja, ali je krnjenje sploh smiselno ali ne. Za slovenščino, ki je morfološko dosti bolj zapletena kot angleščina, obstaja velika verjetnost, da krnjenje pozitivno prispeva k uspešnosti obravnave besedila. Dr. Martin Porter je razvil tudi orodje oziroma programski jezik *Snowball* za razvoj krnilnikov [18], v katerem jih je razvitih kar nekaj: angleški, francoski, španski, portugalski, italijanski, nemški, nizozemski ... Sam sem razvil krnilnik za slovenščino. Krnilnik je dokaj preprost in ga je potrebno še preizkusiti, vendar pa se je v mojem primeru dobro obnesel. Koda krnilnika je dodana kot priloga B.

3.4.3 Normalizacija

Pred postopkom razvrščanja je potrebno dokumente normalizirati. Normalizacija je postopek pretvorbe dokumentov na skupni imenovalec. V grobem delimo postopke na dve skupini:

- določanje skupnih lastnosti,
- vreča besed (angl.: bag of words).

Določanje skupnih lastnosti

Osnovno načelo tega postopka je iskanje skupnih lastnosti dokumentov, ki sodijo v isti razred, in lastnosti, ki bodo pripomogle k čimboljšemu razvrščanju dokumentov, ki sodijo v različne razrede. Določanje skupnih lastnosti lahko temelji na določanju [5]:

- morfemov ali
- n-gramov.

Določanje morfemov je uporabno predvsem v morfološko bogatejših jezikih, n-grami pa imajo to prednost, da jih je mogoče preprosto izračunati (z uporabo statistike), vendar pa je težko izbirati med različnimi n-grami. Za določanje morfemov potrebujemo tudi ustrezen slovar.

Vreča besed

Iz vseh učnih dokumentov pripravimo vektor besed (krnov), ki ga poimenujemo vreča besed. Tako imajo vsi dokumenti enako velikost in jih lahko medsebojno primerjamo z ustreznimi postopki (npr. kosinusna funkcija). Posamezne besede iz vreče besed utežimo glede na njihovo pojavljanje v posameznih dokumentih.

3.4.4 Snowball

Snowball (`http://snowball.tartarus.org`) je programski jezik, ki ga je dr. Martin Porter razvil za razvoj krnilnikov. Vhod je datoteka, ki opisuje delovanje krnilnika za določeni jezik, na izhodu pa lahko dobimo C-jevsko kodo programa ali program v programskem jeziku Java. Razlog, da se je dr. Porter odločil za poseben programski jezik za razvoj krnilnika, leži v nerazumevanju osnovnega algoritma njegovega krnilnika. Ta je bil velikokrat narobe razumljen in tudi narobe uporabljan, in to vedno z oznako Porterjev krnilnik, zato se je Porter odločil razviti sistem za določanje algoritmov krnjenja. Nastal je snowball, jezik z enostavno sestavo. Snowball programiramo tako, da napišemo skripto, ki se nato prevede v enega od programskih jezikov, tega ša je potem potrebno prevesti s prevajalnikom (za C) oziroma se poganja z emulatorjem (Java). Besedilo je obravnavano zaporedno in ko program naleti na besedo, pregleda svoja pravila in ustrezno krni besedo. Elementi obdelave so lahko:

- odstranitev določene končnice,
- postavitev na določeno mesto v besedi (npr. na zadnji samoglasnik),
- dodajanje končnice,
- povratna obdelava (od konca proti začetku besede),

3.4.5 Slovenski krnilnik v snowball-u

Krnilnik temelji na preprostem odstranjevanju končnic. Odločil sem se odstranjevati le končnice, ne pa tudi predpone, ker menim, da predpona besedi določa drug pomen in zato beseda s predpono in beseda brez nje ne sodita v isto skupino. Uporabil sem preprosta pravila za določitev, katere končnice je potrebno odstraniti. Poleg tega sem omejil tudi dolžino besed, ki jo krnim, in pa tudi dolžino že krnjene besede. Tako ne krnim besed, ki so dolge štiri znake ali manj. Celoten postopek odstranjevanja končnic ponovimo štirikrat, kar je bilo določeno empirično. Eden pomembnejših ciljev je bil preprečiti premočno krnjenje. To sem želel doseči tudi na račun slabšega krnjenja. Prva predpostavka pri krnilniku je bila odstranitev samoglasniških končnic,

če je beseda daljša od petih znakov, da preprečimo krnjenje kratkih besed, kot je naprimer beseda mati. Ostale končnice so bile določene že bolj empirično oziroma z ugotavljanjem, kakšne končnice dobijo besede pri spreganju (primer delati: dela-m, dela-š, dela, dela-va, dela-ta, dela-ta, dela-mo, dela-te, dela-jo). Oglejmo si spiske končnic, ki jih odstranjujem besedam dolgim 9 znakov ali več:

-ovski

-evski

-anski

Končnice, ki jih odstranjujem besedam dolgim 8 znakov ali več:

-stvo

-štvo

Ugotovimo novo dolžino besed in uporabimo naslednje postopke. Končnice, ki jih odstranjujem besedam dolgim 7 znakov ali več:

-šen -ski

-ček -ovm

-ega -ovi

-ijo -ija

-ema -ste

-ejo -ite

-ila -šče

-ški -ost

-ast -len

-ven -vna

-čan -iti

Po ugotavljanju nove dolžine besed ponovno odstranimo končnico besed dolgim 7 znakov ali več:

-al -ih

-iv -eg

-ja -je

-em -en

-ev -ov

-jo -ma

-mi -eh

-ij -om

-do -oč

-ti -il

-ec -ka

-in -an

-at -ir

Novim (krnjenim besedam, ki so dolge 6 znakov ali več odstranimo naslednje končnice:

-š

-m

-c

-a

-e

-i

-o

-u

Dobljenim krnjenim besedam dolgim več kot 6 znakov odstranimo samoglasnike, če sledijo soglasniku. Primer je beseda: vzajemno, kjer odstranimo končnico -o in nato v drugi zanki (od štirih) odstranimo še soglasnik, ki sledi samoglasniku (torej končnica -n). Na koncu zanke pa besedam dolgim 6 znakov ali več odstranimo soglasnik. Celoten postopek ponovimo štirikrat in na koncu dobimo krnjene besede.

3.4.6 Učenje

Dobili smo učno množico podatkov, ki je sestavljena iz slovenskih izvlečkov, njihovih naslovov in spiska ključnih besed 1. Izvlečki in naslovi so krnjeni in prekodirani v ISO 8859-2.

Za učenje moramo elemente učne množice še normalizirati (da jih lahko med seboj primerjamo). Normalizacijo sem naredil tako, da sem zbral krne vseh učnih dokumentov in ustvaril vektor z nespremenljivo dolžino, ne glede na dokument (vreča

besed). Dimenzija vektorja je bila 13167 besed. Nato sem določil vrednosti za tf_i (število določene besede iz vreče besed) in $d_{f_i^{tf}}$ (število dokumentov, ki imajo določeno besedo) za vsak dokument, kjer pa sem za besede iz naslova dokumenta uporabil različne uteži in primerjal uspešnost metode glede na uteži besed iz naslova. Po preizkusu uspešnosti metode glede na spremembo uteži od 1 do 10 se je izkazalo, da je v tem primeru najbolj uspešna utež 3.

Besede iz naslova sem utežil s predpostavko, da tako kot ključne besede tudi besede iz naslova podrobneje opisujejo besedilo in je zato verjetnost ujemanja ključnih besed iz naslova med dokumenti, ki so v istem razredu, večja.

Izkazalo se je, da je v tem primeru najbolj uspešna utež 3.

Besede iz naslova sem utežil predvsem s predpostavko, da tako kot ključne besede tudi besede iz naslova podrobneje opisujejo besedilo in je zato verjetnost ujemanja ključnih besed iz naslova med dokumenti, ki so v istem razredu, večja. Zanimivo pa je, da je najboljša utež relativno nizka. Predopostavljam, da je razlog za to predvsem v kratkosti naslova v primerjavi z dokumentom, ki ga obravnavam (izvlečki člankov). Ker pa obravnavam izvlečke člankov, ki zgoščeno opisujejo vsebino članka, lahko sklepamo, da je njihova vsebina dokaj tesno povezana s temo članka. Iz ključnih besed posameznih dokumentov (elementov učne množice) sem določil imena razredov (centroidov) in izračunal elemente posameznih razredov. Preizkus postopka razvrščanja sem izvajal na različnih ravneh MeSH-a, zato je bilo potrebno temu prilagoditi tudi ključne besede učnih množic. Na začetku sem uporabil najvišjo raven MeSH-a (15 centroidov). Temu je bilo potrebno prilagoditi ključne besede, ki opisujejo učne dokumente, potem pa sem lahko izvedel preizkus na tej ravni. Za nižje ravni MeSH-a (podrobnejše opise) je bilo potrebno ključne besede, ki opisujejo učne množice, znova spremeniti, da so bile upoštevane nove ključne besede, zato sem ponovno izvedel preizkus.

3.4.7 Preizkus metode TFIDF

Tudi množico preizkusnih dokumentov je bilo potrebno "pripraviti", da bi lahko izvedli primerjavo z razredi (centroidi). Izvedel sem naslednje korake:

- odstranil nedovoljene znake (ločila, številke ...),
- odstranil nepolnopomenske besede,
- prekodiral besedila v standard ISO 8859-2 (Latin2),
- krnil besede.

Normalizacijo sem izvedel na vektorju besed iz učne množice s predpostavko, da gre za omejeno področje (medicina) in da se besede s tega področja pojavljajo v dokumentih (torej v učni in testni množici) pogosteje kot pa splošne besede, ki bi jih bilo težko zapisati v vektor omejene dolžine.

Najprej sem izbral najvišjo raven ključnih besed in tako dobil 15 centroidov (razredov) v katere sem lahko razvrstil dokumente. Ker je bila učna množica prilagojena tem ključnim besedam, sem lahko preizkus razvrščevalnika izvedel na tej ravni. Rezultat je prikazan v diagramu, ki je dodan v prilogi C.

Nato sem izvedel preizkuse razvrščanja še na nižjih ravneh, kjer je bilo vedno več centroidov zato je postopek postajal vedno težji. Poglejmo si naraščanje števila centroidov s spreminjanjem ravni v MeSH-u:

- raven 0: 15 ključnih besed
- raven 1: 170 ključnih besed
- raven 2: 1308 ključnih besed
- raven 3: 6218 ključnih besed

V prilogi D so podati izvlečki rezultatov uspešnosti razvrščanja po posameznih ravneh.

3.4.8 Preizkus metode naivnega Bayesa

Za primerjavo sem izvedel razvrščanje dokumentov po metodi naivnega Bayesa. Uporabil sem aplikacijo z imenom Rainbow (`http://www-2.cs.cmu.edu/~mccallum/bow/rainbow/`), ki temelji na knjižnici Bow (`http://www-2.cs.cmu.edu/~mccallum/bow/`). Na začetku sem moral podatke pripraviti v obliki, kakršno sprejme aplikacija. Nato

sem iz učnih podatkov določil model podatkov. Naj poudarim, da je potrebno pri metodi naivnega Bayesa z vsakim novim elementom učne množice ustvariti nov model podatkov iz vseh podatkov učne množice.

Ko je bil model učnih podatkov pripravljen, sem izvedel preizkus razvrščevalnika z enakimi preizkusnimi podatki kot pri preizkusu metode TFIDF.

Knjižnica Bow vsebuje tudi krnilnik, ki pa je namenjen angleškim besedilom, tako da sem kot vhodne podatke v program uporabil že krnjene besede.

3.4.9 Metoda podpornih vektorjev

Izvedel sem tudi preizkus z uporabo metode podpornih vektorjev, vendar se je pri hitrem preizkusu z uporabo jedra - funkcije eksponentnega padanja s kvadratom razdalje - izkazalo, da je metoda že na osnovni ravni časovno prezahtevna za računalniški sistem, ki sem ga uporabljal. Najbolj osnovni preizkus razvrščanja v dva razreda je trajal več dni, seveda ob običajni uporabi računalnika. Sistem bi verjetno deloval hitreje na računalniku posebej namenjenemu tej aplikaciji.

Ker se bo aplikacija uporabljala med delom, je hitrost obdelave podatkov izrednega pomena. Potrebno pa je poudariti, da se lahko hitrost obdelave spremeni tudi z uporabo preprostejšega jedra.

3.5 Analiza rezultatov

3.5.1 Analiza predpriprave

V predpripravi sem odstranil znake, ki besedilu ne prinesejo veliko informacij oziroma bi prinesli več težav kot koristi. To so predvsem cifre (od 0 do 9) in ločila. Tako sem zmanjšal dolžino vektorja, ker bi vsak tak znak pomenil en element več v vektorju. Vektor besed normiranega besedila se je tako skrajšal za 10 znakov (cifre) in še dodatnih 10 (ločila).

3.5.2 Analiza krnjenja

V naslednji fazi predpriprave sem krnil besede in na ta način še dodatno skrajšal vektor besed. Tukaj moram poudariti, da se s krnjenjem vektor normiranega besedila lahko krajša tudi na račun premočnega krnjenja, kar je lahko problematično pri razločevanju pripadnosti vektorja določeni ključni besedi. To je tudi eden od razlogov, zakaj ne želim premočnega krnjenja (dodatnega krajšanja vektorja).

Obstaja več načinov, kako preizkusiti krnilnik. Vsi pa temeljijo na predhodni pripravi preizkusnih besed, ki jih je potrebno ročno uvrščati v skupine po sorodnosti. Šele nato lahko preizkusimo krnilnik, ki naj bi krne razvrstil v iste skupine. Premočno krnjenje bi povzročilo večje število krnov v skupini, kot jih je bilo na začetku, preslabo pa manjše.

Želel sem primerjati svoj slovenski krnilnik s Porterjevim krnilnikom za angleški jezik. Ker nisem želel datotek z besedami, razvrščenimi v skupine, pripraviti ročno, kar bi moral narediti v slovenščini in angleščini, sem se problema lotil drugače. Odločil sem se uporabiti dvojezični lematizirani korpus, ki so ga na Inštitutu Jožef Štefan pripravili v okviru Multext-East projekta [10]. Besedilo je urejeno v obliki TEI (Text Encoding Initiative), ki je zapis v obliki XML za označevanje besedila. V korpusu je 15 elementov (besedil v slovenščini in identičnega besedila v angleščini) [9]:

- Ustava Republike Slovenije,
- govor predsednika RS M. Kučana,
- delovanje Državnega zbora,
- publikacija Ekonomsko ogledalo (13 številk),
- Nacionalni program varstva okolja,
- evropski sporazum,
- evropski sporazum - priloga 2,
- strategija Slovenije za vključevanje v EU,

- državni program za prilagajanje zakonodaje - kmetijstvo,
- državni program za prilagajanje zakonodaje - gospodarstvo,
- vademecum Lekove domače lekarne,
- uredba sveta ES št. 3290
  94,
- namestitev in začetek dela z Linuxom,
- lokalizacijske datoteke GNU PO,
- roman G. Orwella: 1984.

Sam korpus je, kot že omenjeno, zapisan v obliki TEI. Vse besede v korpusu imajo dodatek, ki opiše njihove slovnične lastnosti [2]. Poglejmo si primer:

```
<w ana="Ncnsn" lemma="kmetijstvo">Kmetijstvo</w>
```

V zgornjem primeru oznaka w določa, da gre za besedo, ana="Ncnsn"pa določa slovnične lastnosti besede, in sicer:

- N(oun) - samostalnik,
- c(ommon) - občno ime,
- n(euter) - srednji spol,
- s(ingular) - ednina,
- n(ominativ) - imenovalnik.

Tako so označene vse besed v korpusu. Za preizkus krnilnika teh podatkov nisem potreboval, so pa zelo uporabni pri postopku strojnega učenja prevajalskega programa, modeliranju jezika ...

Za preizkus krnilnika sem potreboval le besedo in njeno osnovno obliko (lemo). Iz osnovne oblike besed sem lahko pripravil skupine besed z enakim pomenom. Na ta

način sem lahko združil besede v skupine besed z enakim pomenom. Ker je korpus v angleščini in slovenščini, sem lahko preizkusil in primerjal uspešnost obeh krnilnikov. Najprej sem preizkusil krnilnik za slovenščino in dobil pri 257.761 besedah 4.321 besed, ki so bile krnjene premočno, in 3.005 besed, ki so bile preslabo krnjene. Naslednji preizkus je bil izveden s Porterjevim krnilnikom za angleščino, ki je dostopen na `http://snowball.tartarus.org`. Preizkus je bil narejen na pomensko enakih besedilih kot v slovenščini in rezultat je bil: pri 247.997 besedah je bilo 1.996 premočno krnjenih besed in 357 preslabo krnjenih.

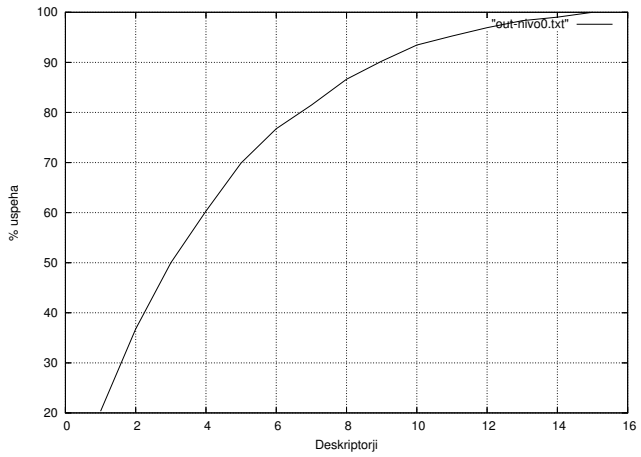
Iz rezultatov vidimo, da je krnilnik za angleščino uspešnejši kot krnilnik za slovenščino, vendar je potrebno upoštevati tudi kompleksnost slovenskega jezika (več sklonov, trije spoli ...).

3.5.3 Analiza rezultatov preizkusa metode TFIDF

Pri preizkusu metode na ravni 0 z utežjo 10 za besede iz naslova sem naredil grafični izpis (Priloga C), ki prikazuje uspešnost razvrščanja z barvno skalo od temno zelene (najuspešnejša razvrstitev) do temno rdeče (najslabša razvrstitev).

Stolpci v grafu predstavljajo posamezne razrede, ki jih ne na ravni 0 15. Vsak stolpec pa je razdeljen še na polovico. Leva polovica predstavlja binarni prikaz razvrščenosti (črno) oziroma nerazvrščenosti (belo). Na tej strani prikazujemo pravo pripadnost razredu. Desna polovica stolpca prikazuje uspešnost razvrščevalnika z barvno skalo od najtemnejše zelene do najtemnejše rdeče. Stolpci so dodatno urejeni od najuspešnejše do najslabše uvrščenega. Ob hitrem pregledu diagrama opazimo večje število zelenih polj (torej uspešne razvrstitve) in tako lahko rečemo, da je metoda dokaj uspešno dodelila ključne besede preizkusnim dokumentom. Z dodatnim preizkušanjem z različnimi utežmi za besede v naslovu se je izkazalo, da je najprimernejša utež za besede iz naslova 3 in tako sem še malo izboljšal rezultate razvrščanja. Pa si podrobneje oglejmo rezultate na ravni 0 pri uteževanju besed v naslovu z utežjo 3.

Imamo 331 preizkusnih dokumentov, ki imajo dodeljeno eno ključno besedo ali več iz množice 15 ključnih besed. Zanima nas uspešnost metode v primerjavi z izvornimi



Slika 3.4: Uspešnost razvrščanja metode TFIDF na ravni 0

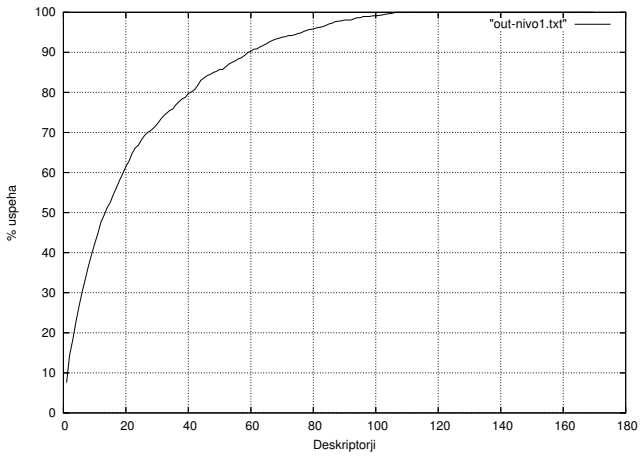
ključnimi besedami (tistih, ki jih je dodelil strokovnjak).

V povprečju imamo na dokument dodeljenih 2.17 ključne besede. Torej je dodeljenih več ključnih besed, kot je dokumentov. V celit je na tej (osnovni) ravni 718 dodeljenih ključnih besed. Ker je njihovo povprečje malo višje kot 2, bosta dodeljeni ključni besedi v najboljšem primeru zasedali prvo in drugo mesto, torej je najuspešnejši tisti algoritem, ki ima na prvih dveh mestih pravilni ključni besedi. Naredimo preizkus, tako da spreminjamo mejo uspešnosti in opazujemo, kako uspešen je naš algoritem (sl. 3.4). S pregledom preizkusnih dokumentov se je pokazalo, da je strokovnjak enemu od dokumentov dodelil 7 ključnih besed. Tako lahko postavimo našo mejo uspešnosti na 7 dodeljenih ključnih besed oziroma je uspešno dodeljena ključna beseda vsaj na sedmem mestu. Če pogledamo diagram, vidimo, da je v tem primeru uspešnost naše metode okoli 80-odstotna.

Uspešnost metode potem hitro narašča in doseže 90 odstotkov že na devetem mestu.

Oglejmo si, kako se spreminja uspešnost metode, če vzamemo podrobnejše ključne besede, torej ključne besede z naslednje ravni v MeSH-u. Na tej ravni imamo 170 možnih ključnih besed. Upešnost naše metode prikazuje diagram 3.5.

Prvi pomemben podatek je, da je strokovnjak enemu od preizkusnih dokumentov dodelil 10 ključnih besed. Ker imamo na razpolago kar 170 možnosti, vidimo, da se

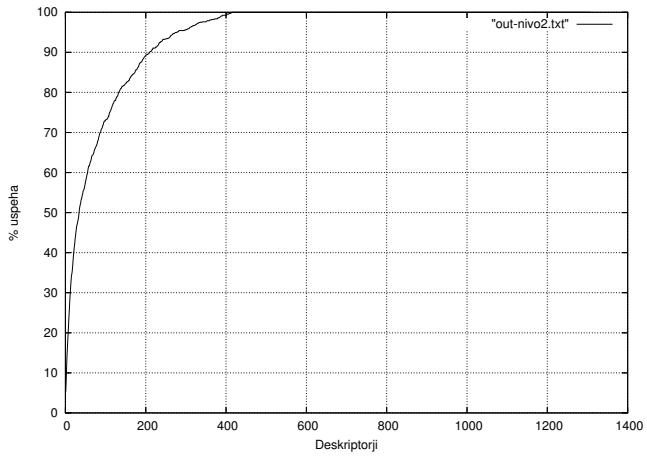


Slika 3.5: Uspešnost razvrščanja metode TFIDF na ravni 1

uspešnost metode zmanjša, če postavimo mejo na največje število dodeljenih ključnih besed (v tem primeru 10). Metoda je takrat 40-odstotno uspešna. Tudi na tej ravni uspešnost hitro narašča in je pri meji 20 ključnih besed 60-odstotna. Torej obstaja 60-odstotna verjetnost, da je prava ključna beseda med prvimi 20 dodeljenimi. Zanimivo pa je, da metoda doseže 100-odstotno uspešnost pri okoli 100 ključnih besedah. Če torej prikažemo prvih 100 dodeljenih ključnih besed, so med njimi vse prave ključne besede. Seveda se tukaj pojavi problem ustreznega intuitivnega prikaza dodeljenih ključnih besed, med katerimi ključno besedo uporabnik izbira sam.

Na ravni 2 v MeSH-u je možnih 1308 ključnih besed in enemu od preizkusnih dokumentov je strokovnjak dodelil 12 ključnih besed. Po tej metodi je pri meji 12 dodeljenih ključnih besed 30-odstotno uspešna. Uspeh pa narašča še hitrjr kot na prvi ravni, tako da pri 133 dodeljenih ključnih besedah dosežemo 80-odstotno uspešnost. Metoda je 100-odstotno zaneslija pri 416 dodeljenih ključnih besedah.

Na ravni 3 imamo največ 11 ključnih besed, ki jih je dodelil strokovnjak, in metoda je na tem mestu 22-odstotno uspešna. 100-odstotno zanesljivost metoda doseže pri 727 dodeljenih ključnih besedah, kar je pri 6218 možnih ključnih besedah zelo dober uspeh.



Slika 3.6: Uspešnost razvrščanja metode TFIDF na ravni 2

3.5.4 Hitrosti metode TFIDF

Program je napisan v programskem jeziku Python (`http://www.python.org`), ki je skriptni jezik, določeni deli obdelve, predvsem postopki za izvedbo TFIDF in kosinusne primerjave so napisani v C-ju. Zaradi uporabe skriptnega jezika, so hitrosti temu primerno nižje, poleg tega pa se nisem spuščal v optimizacijo algoritma, kjer bi lahko pridobil še nekaj procesorskega časa.

V postopku učenja najprej pripravim centroide in nato v dveh delih določim besede, ki pripadajo posameznim centroidom.

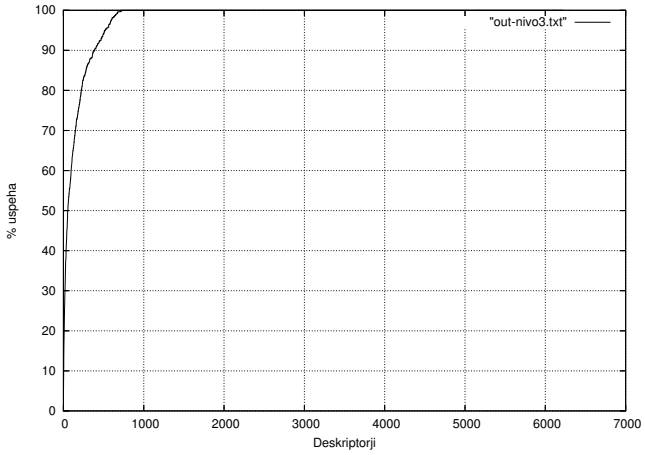
Oglejmo si hitrosti učenja metode na ravni 0:

določanje besed centroidov po delih	1609 sek.
določanje besede centroidov-vsota delov	242 sek.

Pri običajni uporabi lahko metodo učimo takrat, ko program ni v uporabi, tako da čas učenja ni tako pomemben (da le ne traja več dni). Kakšne pa so hitrosti programov pri preizkusu na ravni 0:

testni centroidi	217 sek.
računanje kosinusa centroidov	23 sek.

V postopku preizkušanja najprej izračunamo centroide za vseh 331 preizkusnih dokumentov, kar programu vzame 217 sekund. Če predpostavimo, da je čas določanja centroida dokumenta enakomerno porazdeljen čez vse dokumente, bi v normalni uporabi,



Slika 3.7: Uspešnost razvrščanja metode TFIDF na ravni 3

ko razvrščamo le en dokument, program potreboval manj kot sekundo za določitev centroida dokumenta, ki mu določamo ključno besedo, in okoli 70 ms za izračun kosinusa centroidov.

Zanimala me je tudi hitrost učenja na ravni 1:

določanje besed centroidov po delih	1723 sek.
določanje besede centroidov-vsota delov	368 sek.

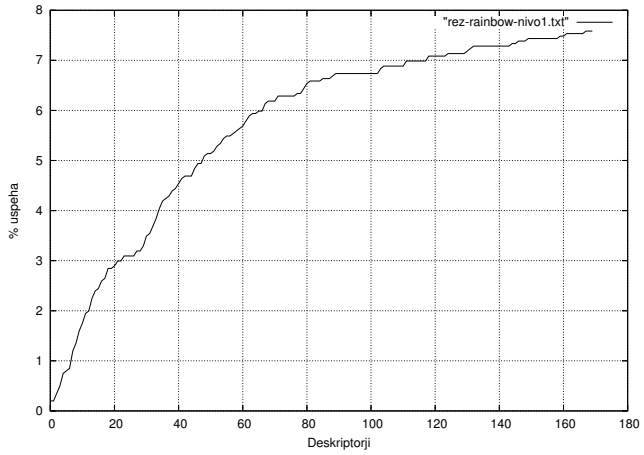
Rezultati so podobni kot na ravni 0, torej se čas obdelave metode pri večjem številu ključnih besed in posledično centroidov pretirano ne spremeni.

Kako pa je s preizkusom na ravni 1? Za učenje smo namreč že ugotovili, da ga lahko izvajamo takrat, ko program ni v uporabi (npr. ponoči):

preizkusni centroidi	212 sek.
računanje kosinusa centroidov	73 sek.

Vidimo torej, da se je čas računanja kosinusa centroidov občutno podaljšal, kar ustreza večjemu številu centroidov, zato je potrebnega tudi več računanja.

Pri časovnih preizkusih pa je potrebno poudariti, da analiziramo čas delovanja posameznega postopka pri običajno obremenjenem računalniku (uporabnikov računalnik), kar seveda lahko prinese odstopanje v časovnih rezultatih. To je lahko problematično za medsebojno primerjavo časov delovanja na različnih ravneh. Nikakor



Slika 3.8: Uspešnost razvrščanja z metodo naivnega Bayesa na ravni 1

pa to ni problematično pri normalni uporabi programa, saj se bo le-ta izvajal na uporabnikovem, torej običajno obremenjenem računalniku.

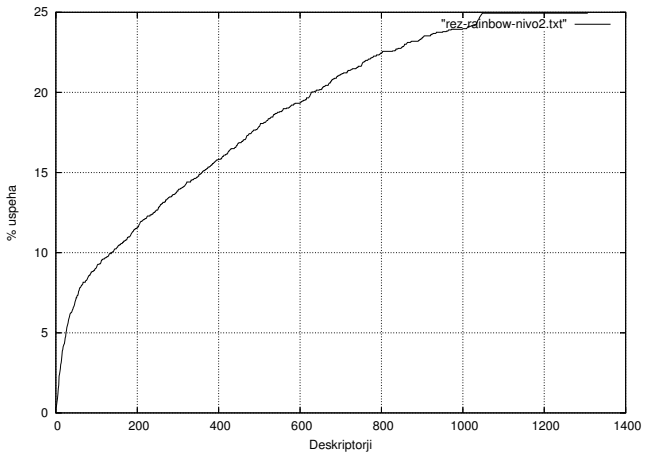
3.5.5 Analiza rezultatov preizkusa metode naivnega Bayesa

Po preizkusu z metodo TFIDF sem za primerjavo izvedel še preizkus metode naivnega Bayesa. Podatki so bili enaki kot pri preizkusu metode TFIDF. Rezultati na sliki 3.8 prikazujejo, da je metoda sicer dokaj uspešna na prvi ravni, a je že na tej osnovni ravni boljša metoda TFIDF. Iz tega lahko sklepamo, da bo tudi na višjih ravneh (pri večjem številu razredov) metoda TFIDF uspešnejša.

Iz prikazanih grafov 3.9 in 3.10 vidimo, da metoda ne doseže 100-odstotne uspešnosti. Razlog za to je zelo majhna verjetnost razvrstitve dokumenta v določen razred in zato je takšna vrednost zanemarjena oziroma ne pride do izraza.

Slika 3.8 prikazuje uspešnost metode naivnega Bayesa na ravni 1, kjer imamo 15 ključnih besed (deskriptorjev). Zanimivo je, da ne dosežemo popolne zanesljivosti dodeljevanja ključnih besed tudi, ko dodelimo vse možne ključne besede. Takrat dosežmo uspešnost nekaj manj kot 60 %.

Sliki 3.9 in 3.10 prikazujeta uspešnost obravnavanje metode na ravneh 2 in 3, kjer se število ključnih besed občutno poveča. Izkaže se, da uspešnost metode močno pade (največ 8 % uspešnost). Zanesljivost metode, torej znantno upade pri večjem številu

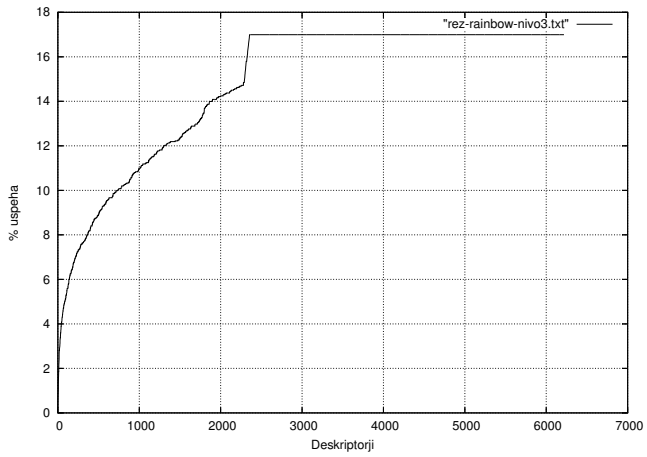


Slika 3.9: Uspešnost razvrščanja z metodo naivnega Bayesa na ravni 2

ključni besed.

Preizkus metode naivnega Bayesa sem naredil na treh ravneh podatkov in izkazalo se je, da je metoda TFIDF dosti uspešnejša.

Oglejmo si primerjavo metod TFIDF in naivnega Bayesa na prvi ravni:



Slika 3.10: Uspešnost razvrčanja z metodo naivnega Bayesa na ravni 3

meja ključnih besed	n.Bayes	TFIDF
1	19.36	20.33
2	30.67	36.77
3	37.41	50.0
4	41.81	60.31
5	47.38	69.91
6	49.46	76.74
7	51.37	81.48
8	52.62	86.63
9	53.78	90.25
10	54.53	93.45
11	55.36	95.26
12	56.11	96.94
13	56.94	98.33
14	58.19	99.02
15	58.93	100.0

V tabeli primerjamo procentualno uspešnost metod TFIDF in naivnega Bayesa in vidimo, da je metoda TFIDF uspešnejša kot metoda naivnega Bayesa. Razlog za to lahko iščemo v tem, da je možno metodo TFIDF pripraviti tako, da bo pisana na kožo

problemu, ki ga obravnava, ker na ta način omogočimo najboljše delovanje sistema. Pri podatkih iz tabele je zanimivo, da obe metodi, tako metoda naivnega Bayesa kot TFIDF, začneta s približno enako verjetnostjo ključne besede. Pri eni dodeljeni ključni besedi ima metoda naivnega Bayesa uspešnost 19.36 % in TFIDF 20.33 %. Tudi pri dveh dodeljenih besedah, je uspešnost metod dokaj podobna. Pri treh dodeljenih besedah, pa se metoda TFIDF pokaže uspešnejša. Uspešnost metode TFIDF nato močno narašča in tudi doseže 100 % uspešnost. Uspešnost metode naivnega Bayesa, narašča enakomerneje in doseže največjo vrednost pri 58.93 % uspešnosti.

Kot vidimo iz rezultatov, je metoda naivnega Bayesa, če gledamo le najverjetnejši razred dokumenta, skoraj enako uspešna kot metoda TFIDF. Pri višjih ravneh pa se to razmerje spremeni. Ker pa je metoda naivnega Bayesa slabša že na prvi ravni, lahko sklepamo, da je njena uspešnost z višjimi ravnmi še slabša, kar nam dokazujejo tudi slike uspešnosti razvrščanja metode naivnega Bayesa (slike 3.8, 3.9 in 3.10).

POGLAVJE 4
PRIHODNOST

Sistem za samodejno dodeljevanje ključnih besed je trenutno narejen v uporabniku dokaj neprijazni obliki, kar bi bilo možno popraviti na naslednja načina:

- z uporabo grafičnega uporabniškega vmesnika (GUI),
- s primernim (grafičnim) prikazom možnih ključnih besed.

Metodo je potrebno prilagoditi sprotnemu učenju, tako da se potrjen prikazan rezultat uporabi kot element v učni množici in na ta način pripomore k izboljšanju postopka dodeljevanja ključnih besed.

Potrebno je tudi formalizirati preizkus krnilnika, ki je uporabljen v tej metodi, in ga morda izboljšati za splošno uporabo kot prvi krnilnik za slovenski jezik, napisan v jeziku snowball. Svoj krnilnik bi želel primerjati tudi z drugima slovenskima krnilnikoma.

Eden od ciljev je tudi uporaba metode na drugih področjih medicine, predvsem v iskanju poimenovanja genov v medicinskih besedilih.

Uspešnost razvrščanja naše metode je relativno dobra. Na uspešnost razvrščanja ne vpliva število deskriptorjev, ki jih ima določen dokument. Seveda pa se lega dodeljenega pravega deskriptorja pri velikem številu možnih deskriptorjev močno povečuje in tako lahko na nivoju 3 (kjer je na razpolago 6218 deskriptorjev), dobimo predlagani pravi deskriptor na mestu okoli 300, kar je zelo uspešno razvrščanje. Rezultat je podan s 5 % odstopanjem. Problem pa se pojavi, kako tak rezultat primerno prikazati, ker mora v tem primeru uporabnik prečesati 300 nepravilnih deskriptorjev, da pride do pravega. To bi lahko rešil s primernim prikazom rezultata, kar pa je naloga za pozneje. Žal nisem imel možnosti uporabe različnih ekspertov, ki bi lahko verificirali tako deskriptorje, ki so bili dodeljeni učnim podatkom, kot tudi uspešnost metode pri dodeljevanju deskriptorjev, čeprav ni bilo ujemanja z deskriptorji, ki jih je dodelih

ekspert. Zaradi včasih zelo podrobnih deskriptorjev je metoda lahko dodelila dokumentu deskriptor, ki je dovolj dober za opis dokumenta, medtem ko ga ekspert ni uporabil.

POGLAVJE 5
NADALJNJE DELO

Celotna metoda je trenutno narejena v dokaj uporabniku neprijazni obliki in jo je potrebno bolj približati uporabniku:

- Uporaba GUI
- Primeren prikaz možnih deskriptorjev

Potrebno je tudi prilagoditi metodo sprotnemu učenju, tako da se uporabi potrjen prikazan rezultat, kot element v učni množici in na ta način pripomore k izboljšanju postopka dodeljevanja deskriptorja.

Želel bi tudi preizkusiti metodo podpornih vektorjev, ki izgleda zelo obetavno in do neke mere združuje statistični pristop (določitev napake metode), kot tudi pristop strojnega učenja. Nenazadnje pa je potrebno formalizirati preizkus krnilnika, ki je uporabljen v tej metodi in ga morda izboljšati za splošno uporabo, kot prvi krnilnik za slovenski jezik napisan v jeziku Snowball [18].

LITERATURA

[1] An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, pages 252–277, 3 julij 1994.

[2] Specifications and notation for multext-east lexicon encoding. 21 marec 2001.

[3] Neural networks and backpropagation. *http://www.work.caltech.edu/cs156/00/handouts/bp.pdf*, 9 februar 2004.

[4] Anand Venkataraman. Naive Bayes Classifier. *http://www.speech.sri.com/people/anand/771/html/node11.htm*, 23 januar 2004.

[5] Thomas Bayer, Ingrid Renz, Michael Stein, and Ulrich Kressel. Domain and language independent feature extraction for statistical text categorization. In L. Evett and T. Rose, editors, *Proc. of the Workshop on Language Engineering for Document Analysis and Recognition*, pages 21–32, Brighton, UK, 1996. Sussex University.

[6] Bill Wilson and Claude Sammut. Induction of Decision Trees. *http://www.cis.temple.edu/~ingargio/cis587/readings/id3-c45.html*, 8 februar 2004.

[7] David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training Algorithms for Linear Text Classifiers. *http://citeseer.nj.nec.com/lewis96training.html*, 27 januar 2004.

[8] L Denoyer, H Zaragoza, and P Gallinari. Hmm-based passage models for document classification and ranking. *http://research.microsoft.com/users/cambridge/hugoz/pubs/pdf/hugoz_ecir%01.pdf*, 3 marec 2004.

[9] Tomaž Erjavec. The elan slovene-english aligned corpus. *http://nl.ijs.si/et/Bib/MT99*, 3 marec 2004.

[10] Tomaž Erjavec. Multext-east home page. 11 marec 2004.

[11] W.B Frakes and R Baeza-Yates. *Information Retrieval.* Prentice Hall, 1992.

[12] Giorgio Ingargiola. Building Classification Models: ID3 and C4.5. *http://www.cis.temple.edu/~ingargio/cis587/readings/id3-c45.html*, 8 februar 2004.

[13] Eui-Hong Han and George Karypis. Centroid-based document classification: Analysis and experimental results, 2000.

[14] C Hsu, C Chang, and C Lin. A practical guide to support vector classification. *http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf*, 9 marec 2004.

[15] Janez Brank. Metoda podpornih vektorjev. *http://www.brank.org/svm/*, 5 februar 2004.

[16] Cho Sun Jin. Kohonen neural network. *http://mmlin1.pha.unc.edu/~jin/QSAR/KNN2/som.html*, 23 januar 2004.

[17] Cornelis H.A. Koster. Document classification. *http://www.cs.kun.nl/~kees/ir2/papers/h03.pdf*, 20 avgust 2004.

[18] Martin Porter. Snowball. *http://snowball.tartarus.org*, 9 februar 2004.

[19] NIST. finite state machine. *http://www.nist.gov/dads/HTML/finiteStateMachine.html*, 8 februar 2004.

[20] Nikola Pavešić. *Razpoznavanje vzorcev.* Fakulteta za elektrotehniko in računalništvo, Ljubljana, 1992.

[21] Polona Vilar and Jure Dimec. Krnjenje kot osnova nekaterih nekonvencionalnih metod poizvedovanja. *http://www.mf.uni-lj.si/~jure/my-hp/VilarDimec4-2000.html*, 9 februar 2004.

[22] Marko Popovič and Peter Willett. The effectiveness of stemming for natural-language access to slovene textual data. *Journal of the american society for information science*, pages 384–390, 1992.

[23] Park S and B Zhang. Large scale unsructured document classification using unlabeled data and syntactic information. In *PAKDD 2003*. LNAI 2637, 2003.

[24] Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, pages 135–168, 2000.

[25] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, pages 1–47, March 2002.

POGLAVJE 6
PRILOGA A

a ali bi bil bila bili bilo bo bodo bolj brez če da do ga
ima in iz je jih jo kaj kako kar ker ki ko kot lahko le leta
med mu na naj ne nekaj ni niso o ob od pa po prav pred pri
s saj samo se sem si sicer smo so sta še ta tako tega tem
ter to tudi v več veliko vendar vse z za zaradi zato zdaj
zelo že 0 1 2 3 4 5 6 7 8 9

POGLAVJE 7
PRILOGA B

```
integers (
p1
)

 groupings (
samoglasniki
crke
soglasniki
)

routines (
remove_ovski
remove_evski
remove_anski
remove_stvo
remove_shtvo
remove_shen
remove_ski
remove_chek
remove_ovm
remove_ovi
remove_ega
remove_ijo
remove_ija
remove_ema
remove_ste
remove_ejo
remove_ite
remove_ila
remove_shche
remove_shki
remove_ost
remove_ast
remove_len
remove_ven
remove_vna
```

```
remove_chan
remove_iti
remove_al
remove_ih
remove_iv
remove_eg
remove_ja
remove_je
remove_em
remove_en
remove_ev
remove_ov
remove_jo
remove_ma
remove_mi
remove_eh
remove_ij
remove_om
remove_do
remove_och
remove_ti
remove_il
remove_ec
remove_ka
remove_in
remove_an
remove_at
remove_sh
remove_ir
remove_m
remove_c
remove_a
remove_e
remove_i
remove_o
remove_u
)

define crke 'abcčdefghijklmnoprsštuvzž'
define samoglasniki 'aeiou'
define soglasniki crke-samoglasniki
```

```
externals (
stem
)

backwardmode (
define remove_ovski as (
 [substring] among (
'ovski' (delete)
)
)
define remove_evski as (
 [substring] among (
'evski' (delete)
)
)
define remove_anski as (
 [substring] among (
'anski' (delete)
)
)
define remove_stvo as (
 [substring] among (
'stvo' (delete)
)
)
define remove_shtvo as (
 [substring] among (
'štvo' (delete)
)
)
define remove_shen as (
 [substring] among (
'šen' (delete)
)
)
define remove_ski as (
 [substring] among (
'ski' (delete)
)
)
define remove_chek as (
 [substring] among (
```

```
'ček' (delete)
)
)
define remove_ovm as (
 [substring] among (
'ovm' (delete)
)
)
define remove_ovi as (
 [substring] among (
'ovi' (delete)
)
)
define remove_ega as (
 [substring] among (
'ega' (delete)
)
)
define remove_ijo as (
 [substring] among (
'ijo' (delete)
)
)
define remove_ija as (
 [substring] among (
'ija' (delete)
)
)
define remove_ema as (
 [substring] among (
'ema' (delete)
)
)
define remove_ste as (
 [substring] among (
'ste' (delete)
)
)
define remove_ejo as (
 [substring] among (
'ejo' (delete)
)
```

```
)
define remove_ite as (
 [substring] among (
'ite' (delete)
)
)
define remove_ila as (
 [substring] among (
'ila' (delete)
)
)
define remove_shche as (
 [substring] among (
'šče' (delete)
)
)
define remove_shki as (
 [substring] among (
'ški' (delete)
)
)
define remove_ost as (
 [substring] among (
'ost' (delete)
)
)
define remove_ast as (
 [substring] among (
'ast' (delete)
)
)
define remove_len as (
 [substring] among (
'len' (delete)
)
)
define remove_ven as (
 [substring] among (
'ven' (delete)
)
)
define remove_vna as (
```

```
 [substring] among (
'vna' (delete)
)
)

define remove_chan as (
 [substring] among (
'čan' (delete)
)
)
define remove_iti as (
 [substring] among (
'iti' (delete)
)
)
define remove_al as (
 [substring] among (
'al' (delete)
)
)
define remove_ih as (
 [substring] among (
'ih' (delete)
)
)
define remove_iv as (
 [substring] among (
'iv' (delete)
)
)
define remove_eg as (
 [substring] among (
'eg' (delete)
)
)
define remove_ja as (
 [substring] among (
'ja' (delete)
)
)
define remove_je as (
 [substring] among (
```

```
'je' (delete)
)
)
define remove_em as (
 [substring] among (
'em' (delete)
)
)
define remove_en as (
 [substring] among (
'en' (delete)
)
)
define remove_ev as (
 [substring] among (
'ev' (delete)
)
)
define remove_ov as (
 [substring] among (
'ov' (delete)
)
)
define remove_jo as (
 [substring] among (
'jo' (delete)
)
)
define remove_ma as (
 [substring] among (
'ma' (delete)
)
)
define remove_mi as (
 [substring] among (
'mi' (delete)
)
)
define remove_eh as (
 [substring] among (
'eh' (delete)
)
```

```
)
define remove_ij as (
 [substring] among (
'ij' (delete)
)
)
define remove_om as (
 [substring] among (
'om' (delete)
)
)
define remove_do as (
 [substring] among (
'do' (delete)
)
)
define remove_och as (
 [substring] among (
'oč' (delete)
)
)
define remove_ti as (
 [substring] among (
'ti' (delete)
)
)
define remove_il as (
 [substring] among (
'il' (delete)
)
)
define remove_ec as (
 [substring] among (
'ec' (delete)
)
)
define remove_ka as (
 [substring] among (
'ka' (delete)
)
)
define remove_in as (
```

```
 [substring] among (
'in' (delete)
)
)
define remove_an as (
 [substring] among (
'an' (delete)
)
)
define remove_at as (
 [substring] among (
'at' (delete)
)
)
define remove_ir as (
 [substring] among (
'ir' (delete)
)
)

define remove_sh as (
 [substring] among (
'š' (delete)
)
)
define remove_m as (
 [substring] among (
'm' (delete)
)
)

define remove_c as (
 [substring] among (
'c' (delete)
)
)
define remove_a as (
 [substring] among (
'a' (delete)
)
)
define remove_e as (
```

```
 [substring] among (
'e' (delete)
)
)
define remove_i as (
 [substring] among (
'i' (delete)
)
)
define remove_o as (
 [substring] among (
'o' (delete)
)
)
define remove_u as (
 [substring] among (
'u' (delete)
)
)
)

define stem as (
$p1 = limit

backwards  (
do (
loop 4 (try (($p1>8) try (remove_ovski or remove_evski or remove_anski))
try (($p1>7) try (remove_stvo or remove_shtvo))
$p1 = size
try (($p1>6) try (remove_shen or remove_ski or remove_chek or
remove_ovm or remove_ega or remove_ovi or remove_ijo or remove_ija or remove_ema or
remove_ste or remove_ejo or remove_ite or remove_ila or remove_shche or remove_shki or
remove_ost or remove_ast or remove_len or remove_ven or remove_vna or remove_chan or rem
$p1 = size
try (($p1>6) try (remove_al or remove_ih or remove_iv or
remove_eg or remove_ja or remove_je or remove_em or remove_en or remove_ev or
remove_ov or remove_jo or remove_ma or remove_mi or remove_eh or remove_ij or
remove_om or remove_do or remove_och or remove_ti or remove_il or remove_ec or
remove_ka or remove_in or remove_an or remove_at or remove_ir))
$p1 = size
try (($p1>5) try (remove_sh or remove_m or remove_c or remove_a or
remove_e or remove_i or remove_o or remove_u))
```

```
$p1 = size
try (($p1>6) try (
[soglasniki] test soglasniki delete
)
)
$p1 = size
try (($p1>5) try (remove_a or remove_e or remove_i or remove_o or remove_u)))

)
)
)
```

POGLAVJE 8
PRILOGA C

POGLAVJE 9
PRILOGA D

POGLAVJE 10
PRILOGA E