UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dušan Šmitran

Napovedovanje branosti spletnih novic s tehniko podpornih vektorjev

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: prof. dr. Blaž Zupan

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00464/2009
Datum: 01.09.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DUŠAN ŠMITRAN**

Naslov: **NAPOVEDOVANJE BRANOSTI SPLETNIH NOVIC S TEHNIKO PODPORNIH VEKTORJEV**

PREDICTION OF POPULARITY OF NEWS IN WEB MAGAZINES USING SUPPORT VECTOR MACHINES

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

V nalogi eksperimentalno preverite, ali lahko iz vsebine spletne novice napoveste njeno branost. V ta namen izberite spletni časopis, ki poleg novic objavlja tudi informacije o branosti. Iz časopisa z ustreznim programom izluščite novice in podatke o branosti, ter te shranite v lastni bazi podatkov. Novice razdelite v manj in bolj brane. S tehnikami vrednotenja algoritmov za uvrščanje v skupine izmerite, kako dobro napoveduje branost novice tehnika podpornih vektorjev.

Mentor:

prof. dr. Blaž Zupan

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo! ouioiu jkhkj

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a Dušan Šmitran,

z vpisno številko 63040373,

sem avtor diplomskega dela z naslovom:

Napovedovanje branosti spletnih novic s tehniko podpornih vektorjev

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom
  prof. dr. Blaža Zupana
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 22.03.2010 Podpis avtorja/-ice:

Zahvala

Rad bi se zahvalil mentorju prof. dr. Blažu Zupanu in Lanu Žagarju, ki sta me vodila in zelo strokovno svetovala pri pristopih, potrebnih za izdelavo celotne diplomske naloge. Prav tako bi se rad zahvalil vsem članom Laboratorija za umetno inteligenco, ki so mi pomagali pri izdelavi diplome, predvsem Juretu Žbontarju, Lanu Umeku in prof. Janezu Demšarju, ki me je navdušil nad Pythonom.

Najbolj bi se pa rad zahvalil za ljubezen mojih staršev.

Diplomsko nalogo posvečam svobodnim slovenskim medijem, ki s svojim delom razsvetljujejo slovenski narod.

Kazalo

Seznam uporabljenih kratic in simbolov

- SVM - (angl.Support vector machine) - tehnika podpornih vektorjev
- BOW - (angl. Bag of words) - množica besed
- TF-IDF - (angl. Text frequency inverse document frequency) - mera
- N-gram - podniz dolžine n
- Stop-words - besede, ki so nepomembne za dani problem
- String kernel - znakovno jedro
- SSK - (angl. Substring sequence kernel) - jedro podzaporedij
- W-S - (angl. Word sequence kernel) - jedro zaporedja besed
- AUC - (angl. Area under the curve) - površina pod krivuljo ROC

Povzetek

Metode strojnega učenja se uspešno uporabljajo pri klasifikaciji teksta. Tehnika podpornih vektorjev je v zadnjih letih doživela razcvet na tem področju in dokazala vsestransko uspešnost pri problemih, ki jih sicer težko zapišemo v atributnem zapisu. V nalogi smo raziskali njeno uporabnost za napovedovanje najbolj branih novic.

Uspešnost tehnike podpornih vektorjev pripisujemo predvsem uporabi jedrnih funkcij, ki primere preslika v višjo dimenzijo in tehniki podpornih vektorjev vrne le podobnost dveh primerov. Odločili smo implementirati tehnike znakovnih jeder in opravili primerjavo jeder, ki delujejo na nivoju znakov in besed. Vzpostavili smo bazo novic in na njej testirali natančnost razvitih klasifikacijskih modelov.

Rezultati so pokazali, da vsekakor obstaja zakonitost o tem, kaj ljudje radi berejo na spletu. Naš model je to zakonitost zajel. Z eksperimenti smo pokazali, da zna razvita tehnika modeliranja zanesljivo napovedati, katera današnja novica bo v prihodnje postala najbolj brana.

Ključne besede:

analiza besedil, strojno učenje, rangiranje novic, tehnika podpornih vektorjev

Abstract

Machine learning methods are successfully used in text classification. The usage of support vector machines, has experienced a boom in the recent years on classifying text. Support vector machine proved its success with comprehensive performance on problems that do not have explicitly defined attributes.

Its success is attributed mainly due to the usage of string kernel, which maps examples into a higher dimensional space. SVM is calling the string kernel to get the information on how much 2 examples are similar. Our goal is to use support vector machine to predict the most read news of tomorrow. We develop the idea of using string kernels for our particular problem and compare kernels operating on different levels. One operating on word level and one on character level. A database was build up, containing 2500 news to test our classification models and string kernels.

We searched for the optimal SVM kernel parameters and compared them with a technique called learning curve. A real world environment was build up, simulating how good a model can predict which of today's news, will become highly readable in the future.

Key words:

text mining, machine learning, news ranking, Support vector machine

Poglavje 1

Uvod

Dandanes večina časopisov ponuja novice na svojih internetnih portalih. Javnosti je tako omogočen enostaven dostop do dnevno svežih novic. Tako imamo informatiziran pregled nad tem, kaj ljudje dejansko berejo, komentirajo in katere novice opredelijo kot kvalitetne. Ti podatki so zelo dragoceni, saj lahko iz njih sklepamo, kaj javnost dejansko zanima in se na njihovi podlagi poskusimo naučiti ter predvideti, katere novice bodo v prihodnosti najbolj brane in bodo oblikovale naslovnice portalov. Prav tako bomo s tem izvedeli katere teme nas dejansko zanimajo in predstavljajo naše vsakodnevno branje.

Standardni sistemi učenja (kot so nevronske mreže ali odločitvena drevesa) delujejo nad podatki, ki jih pretvorimo v atributni zapis, torej v vektorje enake dolžine. V tako opredeljenem prostoru lahko točke preprosto ločimo in nad njimi uporabimo standardne tehnike učenja (interpolacija, razvrščanje v skupine). Obstaja pa kar nekaj problemov, kjer vhodne podatke ne moremo eksplicitno pretvoriti atributni zapis. Primer takih so slike, grafi in tekstovni dokumenti. Problema klasifikacije internetnih novic smo se zato lotili z modernimi metodami za strojno učenje analizo besedil, med katere spada tehnika podpornih vektorjev, tehnike iskanja za klasifikacijo najbolj informativnih besed, ter tehnike vrednotenja napovedanih modelov.

Poglavje 2

Metode in orodja

2.1 Problem podatkov

Za podatke so naprimer besedila, katere ne moremo enostavno pretvoriti v atributni zapis, je ekstrakcija atributov kompleksna in draga kot rešitev celotnega problema. Ne le, da je potrebno domensko znanje, ampak je tudi mogoče izgubiti pomembne podatke med procesom ekstrakcije atributov.

Tekstovni nizi oziroma besedila nimajo eksplicitno definiranih atributov, zato večina standardnih sistemov učenja ni primerna za njihovo klasifikacijo. Besedilo je potrebno najprej pretvoriti v obliko, primerno za obdelavo. Pri tem uporabimo metode kot sta normalizacija in lematizacija. Pri normalizaciji odstranimo vse znake, ki niso črke, ter pretvorimo ostalo besedilo, da to uporablja samo v male črke. Lematizacija pa je proces, kjer odstranimo besedam sklanjatev in množino. Tako dobimo osnovno obliko besede in s tem omogočimo bolj zanesljivo in hitrejše primerjanje. Pomembna metoda je tudi preprosto odstranjevanje neinformativnih besed (šuma), ki ne nosijo informacije o danem problemu. Primer takih besed so "je", "da", "kako", žakaj".

2.2 Jedrna funkcija

Jedrna funkcija je učinkovita alternativa eksplicitni ekstrakciji atributov [**?**]. Temelji na ideji, ki jo je prvi predlagal Vapnik (1995), kjer je jedro funkcija K, ki za dan par primerov vrne njuno podobnost. Pri tem mora K zadostiti Mercerjevemu izreku:

Vse učenje, domensko specifično znanje in ekstrakcija atributov se tako nahaja samo v jedrni funkciji. Jedro kot tako loči klasifikacijski problem (al-

goritem) od domenskega znanja, saj lahko znotraj jedra učne primere pretvorimo v visoko dimenzijo. Za te primere klasifikator ne potrebuje vedeti ničesar drugega kot podobnost med primeri. Tako je problem dobre klasifikacije v večjem delu prenesen na jedro. Če je jedro dobro formulirano in zna dobro ločevati med primeri, bo imel klasifikator zelo lahko delo. Tu pa nastopi problem jedra. Jedrna funkcija je klicana pogostokrat, zato želimo, da je njeno računanje hitro.

Če lahko jedro napišemo tako, da primeri nastopajo le v notranjih produktih z drugimi točkami, potem lahko nad njim uporabljamo več linearnih algoritmov za razvrščanje v skupine, klasifikacijo in regresijo. Najbolj znan primer klasifikatorja, ki temelji na jedrih, je tehnika podpornih vektorjev.

2.3 Tehnika podpornih vektorjev

2.3.1 Ideja

Tehnika podpornih vektorjev je relativno nova družina metod s področja strojnega učenja [3]. Ima dobro teoretično podlago, obenem pa se je izkazala za uspešno tudi v praksi na različnih področjih. Precej dobro se obnese tudi pri problemih z velikim številom atributov in se uspešno izogiba pretiranemu prilagajanju. Ima pa tudi nekaj slabosti:

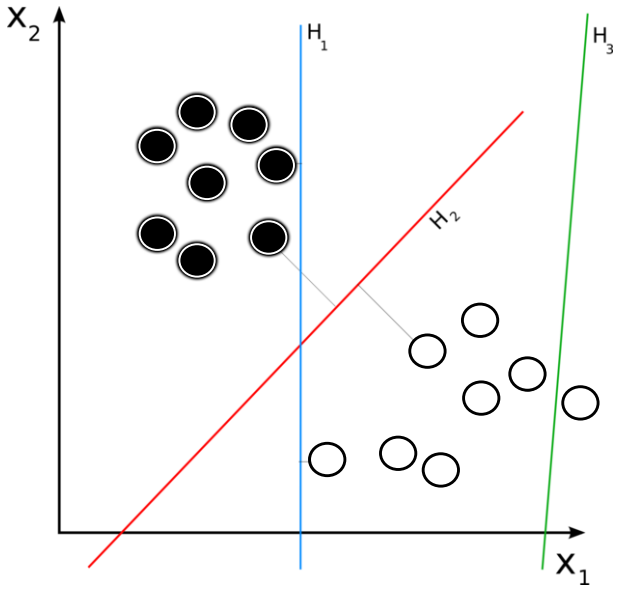
- malenkost bolj zapletena kot nekateri drugi postopki,
- časovna zahtevnost je bližje kvadratni kot linearni v odvisnosti do števila učnih primerkov,
- modeli, ki jih dobimo s to metodo, so slabše razložljivi kot pri nekaterih drugih postopkih (npr. odločitvena drevesa).

Osnovna različica SVM-ja se ukvarja s klasifikacijskimi problemi med dvema razredoma. Vsak primerek je predstavljen s točko v nekem večrazsežnem realnem prostoru, učenje pa postane optimizacijski problem. Iščemo tako hiperravnino, ki bo razmejila predstavnike enega razreda od predstavnikov drugega tako, da bi ležali učni primerki na pravi strani ravnine in še čim dlje od nje. Ta optimizacijski problem lahko s pomočjo nekaj matematičnih prijemov preoblikujemo v dualno obliko, to pa rešujemo numerično. Metodo je mogoče elegantno razširiti, da namesto hiperravnine odkriva tudi drugačne razmejitvene ploskve. Obstajajo tudi razširitve za klasifikacijo v več razredov in za regresijo.

2.3.2 Izpeljava optimizacijskega problema

- Imamo množico primerov, ki so predstavljeni z vektorji v $x_i \in \mathbb{R}^d$
- Imamo dva razreda, pozitivnega in negativnega. Vsak primer spada v enega od teh dveh razredov: $y_i \in \{-1, 1\}$
- Podatki so predstavljeni kot (2.1):

$$D = \{(x_i, y_i) | x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}\}_{i=1}^n \tag{2.1}$$



Slika 2.1: Grafična predstavitev klasifikacijskega problema.

Slika (2.1) nam prikazuje preprost binarni klasifikacijski problem, s katerim želimo, čim bolje ločiti razreda med seboj. Ravnina H_3 sploh ne ločuje razreda, H_1 pa ju slabo loči, saj že v primeru majhnega odstopanja prihaja do napačne klasifikacije. Ravnina H_2 v tem primeru še najbolje ločuje oba razreda, saj je po celotni dolžini dovolj prostora za odstopanje.

Želimo določiti hiperravnino (2.2), ki kar se da dobro razločuje oba razreda. Enačbo ravnine zapišemo kot:

$$w^T x + b = 0 \tag{2.2}$$

Podporna vektorja sta definirana kot (2.4):

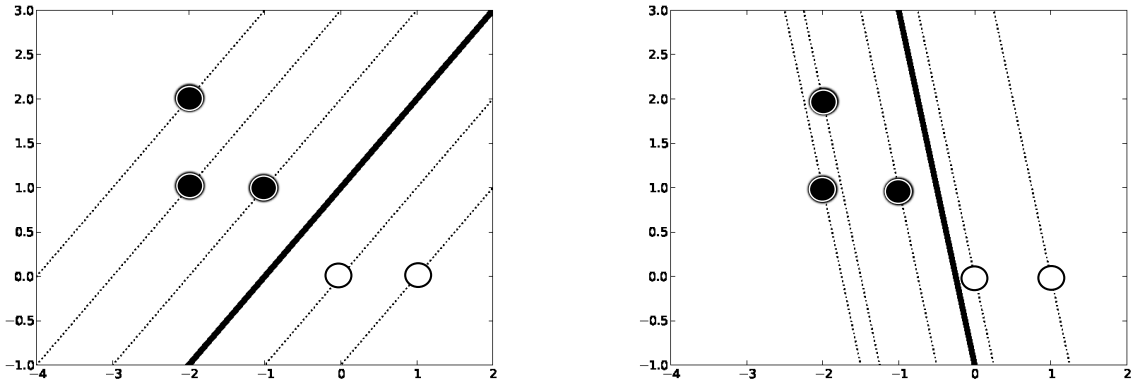
$$w^T x_i + b = 1 \text{ za pozitivni razred in} \tag{2.3}$$

$$w^T x_i + b = -1 \text{ za negativni razred} \tag{2.4}$$

Kjer je w normalni vektor ravnine, b pa konstanta .Hiperravnina je veljavna, kadar izpolnjuje naslednji pogoj:

$$y_i(w^T x_i + b) \geq 1; 1 \leq i \leq n \tag{2.5}$$

Hiperravnin, ki ustrezajo danemu pogoju je lahko več, zato potrebujemo dodatni kriterij, po katerem določimo najboljšo izmed njih. Ko računamo $w^T x + b = 0$ za različne primere x, se pri vsaki točki pozna le to, kako daleč je od razmejitvene hiperravnine in na kateri strani je. Ne želimo, da bi bili učni primeri preblizu hiperravnine, ker lahko postanejo preobčutljivi na majhne perturbacije (2.2).



Slika 2.2: Prikaz iskanja hipperravnine z najširšim robom.

Funkcijski in geometrijski rob

Funkcijski rob (2.6) učnega primera (x_i, y_i) glede na hiperravnino (w, b) je število γ.

$$\gamma(w, b, i) = y_i(w^T x_i + b) \tag{2.6}$$

Funkcijski rob hiperravnine(2.7) glede na učno množico pa je razdalja najbližje točke do hiperravnine.

$$\gamma(w,b) = min_{i=1,...,l}\gamma(w,b,i) \tag{2.7}$$

Geometrijski rob (2.8) učnega primera (x_i, y_i) definiramo kot:

$$\hat{\gamma}(w,b,i) = \gamma(\frac{w}{||w||}, \frac{b}{||w||}, i) \tag{2.8}$$

Med funkcijskim in geometrijskim robom velja naslednja zveza: $\hat{\gamma} = \frac{\gamma}{||w||}$.

Hiperravnina z najširšim robom

Najpogosteje iščemo hiperravnino z najširšim robom. Hiperravnina z najširšim robom ima največjo razdaljo od najbližje točke. Rob učne množice $S = (x_1, y_1), ..., (x_l, y_l)$ je definiran kot:

$$\gamma(S) = max_{w,b}\hat{\gamma}(w,b) \tag{2.9}$$

Hiperravnini (w^*, b^*), ki izpolnjuje pogoj $\gamma(S) = \hat{\gamma}(w^*, b^*)$, pravimo hipperavnina z najširšim robom (ang. maximal margin hyperplane).

Formulacija optimizacijskega problema

Naša želja je, da imamo hiperravnino z največjim geometrijskim robom. Če uporabimo zvezo $\hat{\gamma}(w,b) = \frac{\gamma(w,b)}{||w||}$ in fiksiramo $\gamma(w,b) = 1$, dobimo sledeči optimizacijski problem:

- minimiziraj $\frac{1}{2}||w||^2$
- pri pogojih: $\forall i = 1....l : y_i(w^T x)$

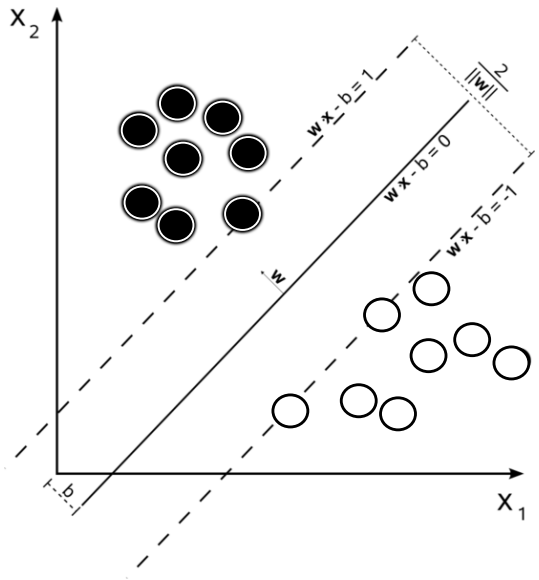
Razdalja med njima pa je enaka:

$$\frac{2}{||w||} \tag{2.10}$$

Vektor w predstavlja normalo za dani podporni vektor. Primere želimo razdeliti tako, da med obema podpornima vektorjema ni več nobenih primerov. Razdalja med podpornima vektorjema je odvisna od w (normale), zato je naš cilj dobiti minimalni w.

Splošna definicija dualnega optimizacijskega problema

Problem klasifikacije s tehniko podpornih vektorjev se tako prevede v optimizacijski problem, ki je odvisen od w:



Slika 2.3: Prikaz podpornih vektorjev in ločilne hiperravnine.

- minimiziraj $\frac{1}{2}||w||^2$
- pri pogojih: $y_i(w^T x_i + b) \geq 1, \forall i = 1....l$

Ekstreme funkcije, ki ima dane pogoje, lahko izračunamo z Lagrangeevijemi multiplikatorji. Tako dobimo dokončno izpeljavo dualnega optimizacijskega problema:

- maksimiraj $\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$
- pri pogojih: $\sum_{i=1}^{n} y_i \alpha_i = 0$

Optimizacijski problem rešujemo s kvadratnim programiranjem. $k(x_i, x_j) = x_i^T x_j$ predstavlja notranji produkt dveh primerov.

2.3.3 Trik z jedri

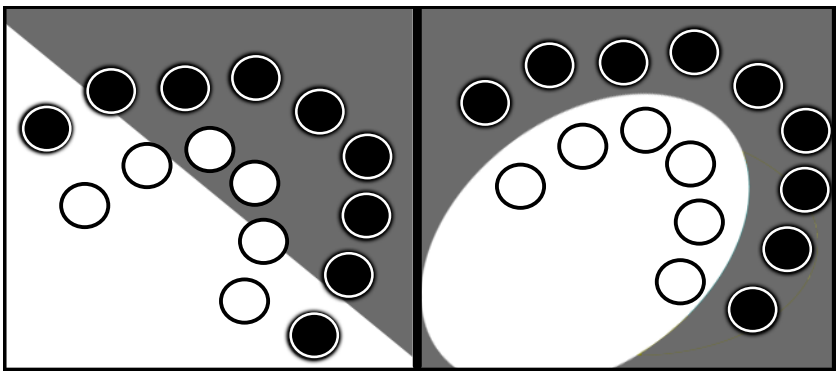
Tu pa nastopi tako imenovani trik jedra, kjer lahko notranji produkt zamenjamo z jedrno funkcijo K. Znotraj jedra lahko primera preslikamo v poljubno dimenzijo in izračunamo njuno podobnost. Ideja je bila predlagana leta 1992 z namenom oblikovanja nelinearnih klasifikatorjev.

Najbolj znana nelinearna jedra:

- RBF (radial basis function): $k(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), for : \gamma > 0$
- Polinomsko jedro: $k(x_i, x_j) = (x_i \cdot x_j)^d$
- Gaussovo jedro: $k(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$

2.3.4 Lastnosti SVM-ja

Splošno najbolj učinkovito jedro je RBF, saj ima v primerjavi z linearnim, sposobnost ustvarjanja ukrivljenih hiperravnin. Tako lahko veliko bolj učinkovito pri kompleksnih problemih in obravnavi šuma. Parameter γ določa, kako zelo lahko ukrivi hiperravnino pri ločevanju razredov.



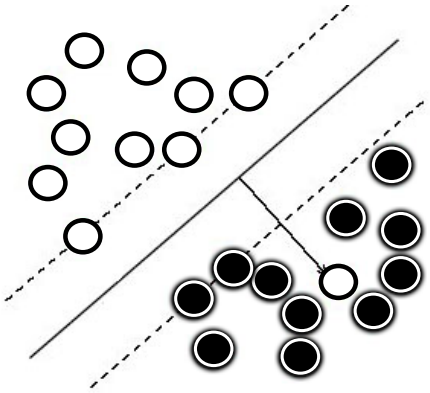
Slika 2.4: Način določevanja podpornih vektorjev. Levo je prikazano linearno, desno pa RBF jedro.

S slike (2.4) je lepo razvidno, da v določenih primerih linearno jedro ne more najti optimalne hiperravnine, medtem ko jedro RBF pravilno najde ukrivljeno hiperravnino. Moramo pa biti pazljivi, saj se ob velikih vrednostih parametra γ hiperravnina preveč prilagodi obliki podatkov in posledično onemogoči učinkovito obravnavo šuma.

V primeru, da pozitivnih in negativnih primerov ne moremo razmejiti z ravnino, vpeljemo kazenske spremenljivke ξ. Tako se pogoj razširimo v:

- minimiziraj $\frac{1}{2}||w||^2 + C\sum_{i=1}^{l}\xi_i$
- pri pogojih: $y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i = 1....l$

Konstanta C je parameter tehnike podpornih vektorjev, s katerim določamo obravnavo šum pri učenju (2.5). Z njo določimo, kako zelo naj dopušča



Slika 2.5: Kako s parametrom C obravnavamo šum.

odstopanja od osnovne skupine enega razreda. Za dobro klasifikacijo je potrebno izbrati pravilno vrednost konstante, s katero bo algoritem znal dovolj dobro obravnavati šum. Višji kot je, bolj je tehnika tolerantna do odstopanja. Biti pa moramo pazljivi, saj od neke vrednosti dalje višji C-ji ne prinašajo boljše klasifikacije, vendar le po nepotrebnem upočasnijo proces učenja.

Eden izmed pomembnih parametrov je tudi obteževanje razredov zaradi neuravnovešenih podatkov med razredi. V primeru, da ima eden od razredov znatno več primerov kot drugi, moramo nasprotnemu razredu podati sorazmerno večjo utež glede na velikost.

$$\frac{S_+}{S_-} = \frac{l_-}{l_+} \tag{2.11}$$

Utež določimo z enačbo (2.11), kjer S predstavlja utež razreda, l pa velikost razreda [6]. Tehnika podpornih vektorjev je v osnovi namenjen reševanju dualnih problemov (dva razreda). V primeru, da imamo več kot dva razreda, lahko uporabimo dva načina reševanja problema:

- eden proti enemu,
- eden proti vsem.

2.4 Vreča besed

Vreča besed (angl. bag of words ali BOW) je preprosta tehnika preoblikovanja besedila za potrebe klasifikacije. Besedilo se najprej normaliziramo (samo male

črke), odstranimo nepotrebne neinformativne besede in nato lematiziramo preostalo.

lastnik	4
delen	3
imeti	3
zgodba	3

Tabela 2.1: Primer oblike BOW vektorja.

Končna predstavitev nekega besedila je torej neurejen vektor besed ter njihova frekvenca v danem besedilu. Problem take predstavitve je, da izgubimo vso informacijo o zaporedju in sklanjatvi besed. Prednost BOW formulacije je hiter in preprost izračun podobnosti dveh primerov. Dani pristop se je v praksi pokazal kot zelo učinkovit pri filtriranju nezaželene pošte.

2.5 TF-IDF

TF-IDF je mera, ki se uporablja za določevanje pomembnosti besede v danem besedilu, glede na celotno zbirko besedil. Pomembnost besede je določena s frekvenco v danem besedilu, vendar je zmanjšana glede na velikost frekvence v celotni zbirki. Naj bo pogostost besede i definirana kot:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{2.12}$$

TF (2.12) je definirana kot frekvenca posamezne besede v danem besedilu j. Pri tem $n_{i,j}$ predstavlja frekvenco besede, deljeno s številom vseh besed v besedilu.

$$idf_i = \log(\frac{|D|}{|\{d : t_i \in d\}|}) \tag{2.13}$$

IDF (2.13) izražamo kot, razmerje med številom vsem besedil in številom besedil, kjer se nahaja dana beseda. Tako dobimo za besedo, ki se velikokrat pojavi visok imenovalec, ter posledično nizek IDF.

Normalizirana utež besede s TF-IDF se tako glasi:

$$(tf - idf)_{i,j} = tf_{i,j} * idf_i \tag{2.14}$$

Tak način normalizacije pomembnosti besed je nujen za dobro klasifikacijo, saj tako odpravimo vpliv besed, ki se zelo pogosto pojavljajo v celotni zbirki in s tem odvračajo pozornost od bolj redkih in bolj informativnih(ključnih) besed za dano besedilo. TF-IDF se uporablja predvsem pri BOW načinu obravnave besedila.

2.6 N-gram

N-gram je jezikovno neodvisen pristop predstavitve besedila. Besedila pretvorimo v vektor z visoko dimenzijo, kjer vsak element predstavlja zaporedni podniz dolžine n. Največja možna dimenzija za dani n je tako lahko $|A|^n$. A je definiran kot celotna abeceda znakov. Iz tega lahko vidimo, da je za visoke n-je lahko ogromna dimenzija.

Primer: Za dano besedilo d izpišemo vse možne 3-grame.
d: 'barbara'
3-grami: 'bar', 'arb', 'rba', 'bar', 'ara'

N-gram se uporablja za analizo besedila in primerjanja podobnosti. To storimo tako, da najdemo vse možne n-grame obeh primerov. Nato vsakemu določimo njegovo frekvenco v testu. Končni rezultat je notranji produkt obeh primerov, glede na enake n-grame in njihove frekvence.

2.7 Ostala znakovna jedra

2.7.1 Jedro podzaporedij

Ideja jedra podzaporedij(angl. substring sequence kernel ali SSK) je podobna jedru z n-grami [7]. Oba primerjata besedili na podlagi ujemajočih se podnizov. Posebnost jedra podzaporedij je v tem, da ni nujno, da so ti podnizi zaporedni (skupaj). Podniz se lahko nahaja v širšem delu besedila. S tem načinom, ko omogočimo preskakovanje, je veliko lažje premosti problem slovničnih napak. V praksi se zelo dobro obnese na nezanesljivih in kratkih podatkih.

$$w = \lambda^l, \lambda \in (0, 1), l = \text{širina razpona} \tag{2.15}$$

Utež ujemajočega podniza se določi glede na razpon, na katerem se nahaja (2.15). Širši kot je razpon (glede na λ), manj je podniz pomemben. Utež

imenujemo upadajoči faktor. Manjši kot je, bolj kaznujemo preskoke v podnizu.

Primer: podniz 'c-a-r' se nahaja v besedi '**car**d' in v besedi '**c**ust**ar**d'. Razlika pa je predvsem v razponu. Niz 'c-a-r' se v 'card' pojavi v razponu z dolžino 3. Tako je utež enaka λ^3. V 'custard' pa se nahaja v razponu 6, zato je utež enaka λ^6.

Končno ujemanje pa se izračuna kot notranji produkt med ujemajočimi se podnizi obeh besedil.

Problem takšnega pristop je predvsem zelo visoka časovna zahtevnost, ki je nepraktična tudi za krajša besedila. Ta problem se rešuje z dinamično implementacijo in učinkovitim približkom, glede na celotno zbirko. Približek išče podnize, ki so ortogonalni na celotno zbirko primerov. Proces izbora je sledeč:

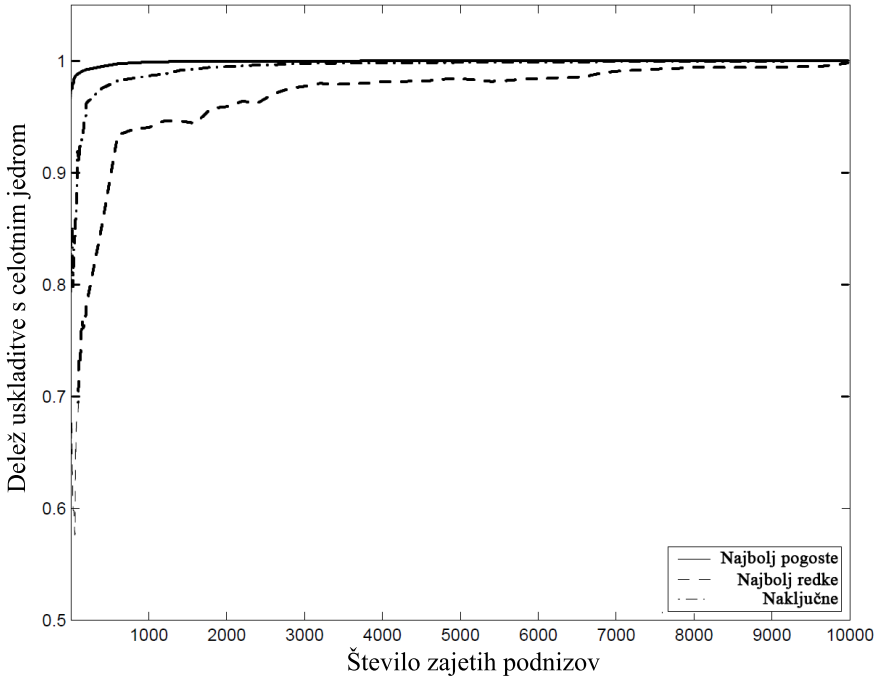
- zajamemo vse možne podnize iz zbirke besedil,
- izberemo podmnožico podnizov, ki se pojavijo najpogosteje v celotni zbirki.

Prednost danega pristopa je, da že z malo podmnožico dosežemo usklajenost s celotnim jedrom. Tako lahko za vsako besedilo ustvarimo že vnaprej pripravljen vektor in se ob vsakem klicu jedra izvede le notranji produkt med dvema primeroma.

Slika (2.6) prikazuje, kako hitro se določena podmnožica ujema s celotnim jedrom. Iz slike lahko razberemo, da se najhitreje uskladi podmnožica, ki vsebuje najbolj pogoste podnize, medtem ko se množica z najmanj pogostimi uskladi najpočasneje. Kot pričakovano se naključna množica po usklajenosti nahaja med obema množicama. Tako je potrebno zajeti le majhen delež podnizov, ki nam služi za pridobitev dobre uskladitve.

2.7.2 Jedro zaporedja besed

Jedro zaporedja besed (angl. Word sequence kernel ali W-S) je implementacija jedra podzaporedij, ki namesto z znaki operira z besedami [1]. Prehod na besede s seboj prinese zanimivo posledico. Časovna zahtevnost se zmanjša, saj je dolžina možnih ujemajočih se zaporedij toliko nižja, da je tak način bistveno manj zahteven od izvedbe celotnega jedra podzaporedij. V svoji originalni obliki se jedro podzaporedij nanaša le na štetje, kolikokrat se podnizi pojavijo v besedilu in ne vsebuje sheme za obteževanje elementov abecede.



Slika 2.6: Uskladitev približka s celotnim jedrom.

Zato jedro zaporedja besed vsebuje dve dodatni razširitvi. Prva omogoča določevanja uteži $\lambda \in (0, 1)$ za vsako besedo posebej in s tem uporabo uveljavljene metode tf-idf. Na primer: besedi 'bilo' določimo nizko utež ($\lambda_{bilo} = 0.01$), saj s seboj ne prinese bistvene informacije o vsebini besedila. Besedi 'umor' pa lahko določimo visoko utež, saj s seboj nosi visoko informativno težo ($\lambda_{umor} = 0.5$). Druga razširitev vpelje uporabo različnih upadajočih faktorjev za luknje v ujemajočem zaporedju.

2.8 Površina pod ROC krivuljo

Površina pod ROC krivuljo - (angl. Area under the Curve ali AUC) je statistična mera za primerjavo klasifikacijske natančnosti modelov [4]. Temelji na principu klasifikatorja, ki vrača verjetnost, da dani primer spada k pozitivnemu razredu. Mero lahko interpretiramo kot verjetnost, da bo klasifikator za naključno izbran pozitiven primer vrnil večjo verjetnost, kot za naključno izbran negativen primer.

Primer izračuna:

- najprej sortiramo rezultate padajoče po verjetnosti,
- AUC krivuljo narišemo tako, da sledimo rezultatom v tabeli,
- če je razred primera enak 1 potem se premaknemo za eno gor,
- sicer pa gremo v desno.

Korak	**Razred**	**Verjetnost**
1	1	0.90
2	1	0.80
3	1	0.70
4	-1	0.55
5	1	0.50
6	-1	0.40
7	-1	0.30
8	-1	0.20
9	-1	0.10

Tabela 2.2: Klasifikacijski podatki

	5	6	7	8	9
3	4				
2					
1					

Tabela 2.3: Prikaz korakov za risanje AUC grafa

Koraki so prikazani v tabeli (2.3). AUC površina je obarvana sivo. AUC meri kako dobro klasifikator pozitivnim primerom pripiše višjo verjetnost kot negativnim. AUC lahko zavzame vrednost med 0.5 in 1.

2.9 Krivulja učenja

Krivulja učenja je tehnika, ki prikazuje kako hitro se klasifikator nauči na istih podatkih. Naprimer, da imamo množico 1000 primerov, ki jo razdelimo

na manjše podmnožice [100, 200, 300, 600]. Nato s prečnim primerjanjem merimo vrednost AUC pri posameznih podmnožicah primerov. Rezultat je grafična predstavitev krivulje, ki za posamezne pomnožice prikazuje doseženo natančnost.

Poglavje 3

Razvoj sistema za napovedovanje najbolj branih novic

3.1 Razvoj baze podatkov

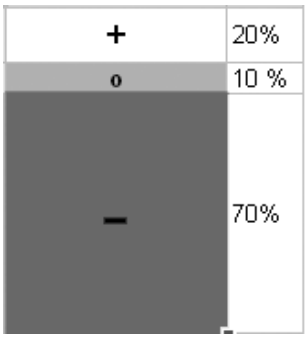
Za vsako iskanje zakonitosti v podatkih je pomembno imeti takšne podatke, ki bodo reprezentativno in nepristransko predstavljali probleme, katere bomo srečali v realnem svetu. Ne želimo, da bi podatki oblikovali rezultate. Želimo, da model na nepristranskih podatkih razvije pristop, ki bo kljuboval vsem realnim situacijam.

Področje diplomske naloge so novice, ki se nahajajo na internetnih portalih. Najprej smo morali izbrati portal, ki nam bo nudil vse potrebne informacije o novicah. Najbolj kritični dejavnik je bil seveda branost določene novice. Edini internetni portal, ki natančno nudi to informacijo je www.revija-reporter.si. Razvili smo plezalca, ki je prenesel vsebino novic in njihovo branost. Pri tem smo vzeli v obravnavo samo napisani članek brez komentarjev.

Uspelo nam je pridobiti okoli 2500 novic za obdobje 22. mesecev, ki so razmeroma dolge, v primerjavi z ostalimi internetnimi portali. To predstavlja približno 100 novic na mesec, 25 na teden in 4 na dan.

Za potrebe klasifikacije smo podatke sortirali po branosti in jih razdelili na pozitiven in negativen razred. Slika (3.1) nam prikazuje razporeditev podatkov. Iz nje je razvidno, da nam pozitivni razred predstavlja 20% najbolj branih novic. Ker pa nismo mogli definirati točne meje med najbolj brano in zelo brano novico smo se odločili, da ustvarimo med najbolj branimi in ostalimi novicami majhen prepad velikosti 10%. Ostalih 70% pa predstavlja negativni

razred, oziroma novice, ki niso najbolj brane.



Slika 3.1: Prikaz razdelitve podatkov v pozitivni in negativni razred.

Branost	**Razred**	
	Pozitivni	Negativni
Maksimum	18016	977
Minimum	1150	62
Povprečje	1993	584
Stand. odklon	1687	209

Tabela 3.1: Statistični podatki strukture podatkov glede na razred.

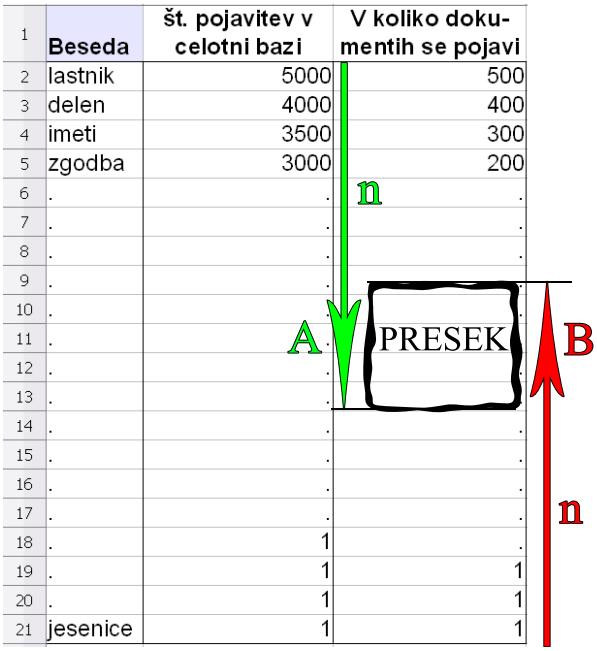
3.2 Znakovna jedra

3.2.1 BOW jedro

Ko smo imeli izdelano bazo novic, smo pričeli z razvojem jeder, ki bodo znala povedati, kako zelo sta si dve novici podobni. Odločili smo se za knjižnico libsvm^{TM} [2], ki vsak primer vidi kot vektor realnih števil. Zato mora znakovno jedro preoblikovati vsak primer v vektor enake dolžine, saj libsvm med njimi izvaja preprosti notranji produkt. Prednost danega pristopa je visoka hitrost, vendar je potrebno za doseg zahtevane oblike podatkov uporabiti približek.

Najprej smo se lotili izdelave BOW jedra, ki temelji na principu vreče besed. Za vsako novico smo izvedli normalizacijo in lematizacijo, ter odstranili nepotrebne besede (stop-words). Cilj je, da imamo besede v isti sklanjatvi, saj bomo izvajali primerjavo na nivoju besed. Skupno število edinstvenih besed po

vseh zgoraj navedenih postopkih se giba okoli 30.000 besed s povprečno dolžino osmih znakov ($\sigma = 2$). Problem, s katerim smo se srečali je, da preprosto nismo mogli obravnavati vseh besede, saj bi bil vektor za vsako novico dolg 30.000, kar je nepraktično v realnih situacijah. Prav tako je iz teoretičnega vidika nesmiselno uporabiti vse besede, saj veliko besed nastopa v eni ali parih novicah in s tem ne prinašajo večje natančnosti pri primerjanju dveh novic. Naša želja je imeti hitro jedro, zato se moramo omejiti na vektor dolžine nekaj tisoč besed. Tako smo se spoprijeli s problemom izbora pravih besed.



Slika 3.2: Prikaz izbora besede za BOW jedro.

Znan pristop (ki je prav tako prikazan na sliki 3.2) k omenjenemu približku je, da ustvarimo dve množici besed:

- Množica A, ki vsebuje seznam najbolj pogostih besed v celotni zbirki novic,
- Množica B, ki vsebuje seznam besed, ki se pojavijo v najmanj novicah.

Pristop temelji na tem, da omejimo obe množici za število n in nato n povečujemo toliko časa, dokler ne dobimo dovolj velik presek. Pri tem je tudi smiselno določiti prag, v vsaj koliko novicah se mora beseda pojaviti, saj nam besede, ki se pojavijo v samo eni novici, ne prinesejo nobene prednosti

pri izvajanju jedra. Problem izbora besed tako temelji na pravilnem izboru dveh spremenljivk, ki smo ga rešili s testiranjem klasifikacije na različnih parih spremenljivk.

Prišli smo do rezultata, da je za klasifikacijo najbolj primerno izbrati besede, ki se pojavijo v vsaj 13 novicah. Količino besed pa smo omejili na 4000, saj nadaljnje povišanje ni prineslo boljših rezultatov.

Ko smo imeli izbrane skupne besede, smo s tem lahko določili vektor besed za posamezno novico. Pri tem smo se želeli izogniti vplivu besed, ki se pojavijo v skoraj vseh novicah, zato smo nad podatki uporabili tf-idf pristop in skaliranje.

prvotno	1	4	4	7	3	=	19
skalirano	0.05	0.21	0.21	0.37	0.16	=	1

Tabela 3.2: Prikaz delovanja skaliranja.

Skaliranje je eden izmed priporočenih pristopov [5], saj se z njim izognemo numeričnim problemom znotraj tehnike podpornih vektorjev. Temelji na tem, da frekvenco besed posamezne novice normaliziramo tako, da je njihov seštevek enak 1. Vsi ti dodatni procesi znatno izboljšajo natančnost klasifikacije BOW jedra.

3.2.2 Jedro z n-grami

Za učinkovito klasifikacijo jedra n-gram je zelo pomembno izbrati pravilen parameter k, ki določa dolžino primerjanih podnizov. Za besedilo je najbolj primeren k, ki se nahaja v razponu od 4 pa do 14. Prednost pri večji k-jih je, da kljub svojemu znakovnemu pristopu zajamejo semantične in lingvistične informacije posameznega besedila. Moramo pa biti konzervativni pri izboru k-ja in dati prednost nižjim k-jem.

V tabeli (3.3) so prikazani rezultati merjenja AUC za posamezne k-je na 400 primerih z uporabo jedra RBF. Iz tabele je lepo razvidno kako AUC narašča z vse večjimi k-ji, vendar se hitro ustavi pri prevelikih k-jih. Največjo natančnost smo dosegli pri k je enako 8. Pri tem je zanimivo, da je 8 tudi povprečna dolžina besede v naši celotni zbirki.

Kljub temu, da n-gram temelji na nivoju znakov, obstaja zelo preprosta in učinkovita implementacija, ki zajame vse podnize obeh besedil. Ker pa smo morali pripraviti podatke primerne za libsvm, moramo uporabiti približek, ki bo omogočil, da podamo podatke v obliki vektorja realnih številk.

	AUC
4	0.811
6	0.854
8	**0.887**
10	0.871
12	0.867
14	0.758

Tabela 3.3: Vrednost AUC-ja za posamezne k-je.

Naš približek je temeljil na pristopu, ki se uporablja pri SSK jedru. V našem primeru smo se odločili, da množico omejimo na 6000 najpogostejših edinstvenih podnizov, predvsem zaradi uporabe razmeroma visokih k-jev.

Ko najdemo primerno podmnožico za vsako novico, izračunamo frekvenco posameznih podnizov in rezultat shranimo kot vektor realnih števil. Pomembno je, da ne pozabimo skalirati dani vektor, kjer je vsota celotnega vektorja enaka 1.

3.3 Implementacija s SVM-jem

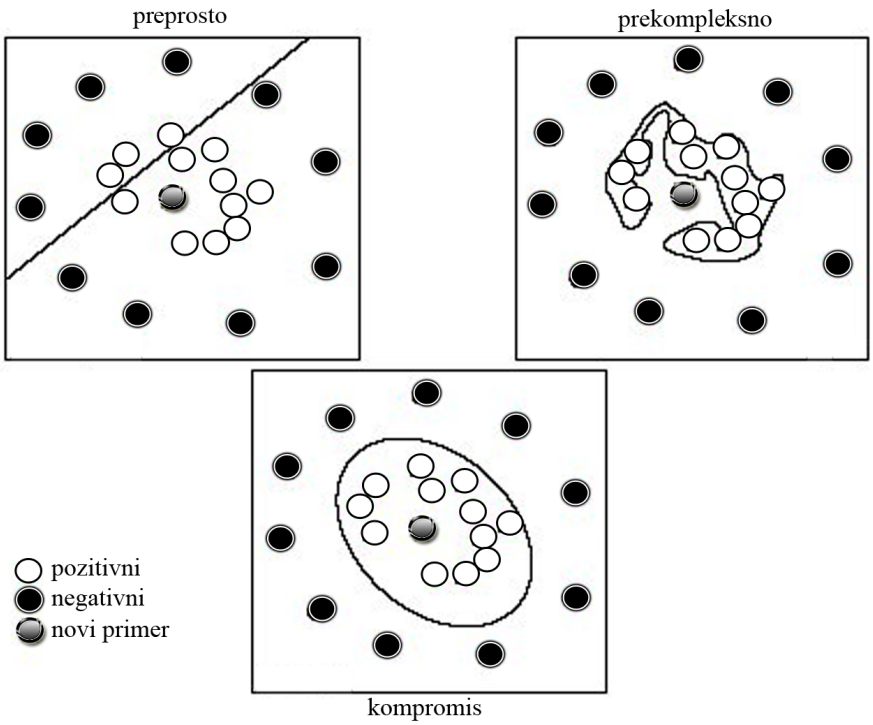
Po izgradnji znakovnih jeder, smo se lotili njihove implementacije s SVM-jem. Pri našem danem problemu smo se usmerili k obravnavanju linearnega in jedra RBF, saj polinomsko jedro ni pokazalo praktično zadovoljivih rezultatov za nadaljnjo obravnavo. V splošnem sta linearno in RBF jedro najbolj primerni za klasifikacijo besedila.

Pri tem smo morali poiskati vrednosti za vse dane parametre SVM-ja. Natančnost SVM modela je močno odvisna od optimalne določitve parametrov, kot so uteži razreda, C in γ. Pri tem moramo biti pazljivi, da najdemo konzervativne vrednosti parametrov.

Želimo se izogniti situaciji, kjer se model premalo prilagodi podatkom. Primer je viden na sliki (3.3), kjer linearno jedro išče hiperravnino v prostoru primerov, ki je prekompleksen za preprosto linearno rešitev. Hkrati pa želimo biti pazljivi, saj se lahko pri visoki vrednostih parametrov model prekomerno prilagodi podatkom. Primer prekomerne prilagoditve je lepo viden z uporabo jedra RBF, ki ima previsoki vrednosti parametrov C in γ.

Uteži smo določili s splošno formulo, ki je navedena zgoraj.

Formulo za izračun uteži smo praktično preizkusili in dokazali, da je na-



Slika 3.3: Prikaz problema premajhnega in prekomernega prilagajanja modela.

jboljša glede na ostale možne kombinacije. Izračuna parametra C, ki je namenjen obravnavanju šuma v zbirki novic, smo se lotili kot obravnave linearnega problema. Parameter C smo v kombinaciji s preprosto linearno klasifikacijo povečevali toliko časa, dokler je z vsakim povečanjem posledično sledila večja vrednost AUC mere. Želja je, da imamo skromen C, saj ne želimo, da se SVM model prekomerno prilagodi učnim podatkom. Nadaljnje povečanje C-ja od določene vrednosti naprej, ne prinaša večje natančnosti modela, ampak le po nepotrebnem upočasni hitrost zgradbe modela.

Razred	**splošno**	**velikost**	**skrajšano**
S+	l-	1752	4
S-	l+	500	1

Tabela 3.4: Izračunanje uteži za posamezne razrede

C	**AUC**
1	0.71
2	0.85
3	0.87
4	0.89
5	**0.90**
6	0.89
7	0.89
8	0.88
9	0.88

Tabela 3.5: Iskanje optimalnega C-ja za SVM model.

Prišli smo do rezultata, da AUC doseže vrhunec pri $C = 5$ (3.5). Od tu naprej se AUC samo še zmanjšuje, saj model z vse večjimi C-ji ne zna več tako dobro obravnavati šuma v podatkih.

Za jedro RBF smo poleg C-ja potrebovali izračunati še parameter γ. Parameter C smo vzeli iz prejšnjega sklepa. Opravili smo preprost način iskanja, kjer smo s $C = 5$, parameter γ zviševali toliko časa, dokler AUC ni začel vidno upadati.

γ	Meritev 1	Meritev 2	Meritev 3	Meritev 4	**Povprečje**
14	**0.86**	**0.86**	**0.83**	**0.81**	**0.840**
12	0.86	0.86	0.83	0.8	0.839
16	0.86	0.86	0.83	0.81	0.839
18	0.86	0.86	0.83	0.8	0.838
22	0.85	0.86	0.84	0.8	0.838
26	0.85	0.87	0.84	0.79	0.838
10	0.86	0.86	0.83	0.8	0.837
20	0.86	0.86	0.84	0.8	0.837
28	0.85	0.87	0.84	0.79	0.837
24	0.85	0.86	0.84	0.79	0.837
8	0.86	0.86	0.83	0.79	0.834
6	0.86	0.85	0.82	0.79	0.830
4	0.85	0.84	0.82	0.78	0.824
0	0.83	0.46	0.48	0.76	0.633
2	0.65	0.34	0.65	0.78	0.605

Tabela 3.6: Uspešnost klasifikacije (AUC) pri različnih vrednostih parametra γ.

Poglavje 4

Eksperimentalno ovrednotenje

4.1 Podatki

Naši podatki so vsebovali 2500 novic, za obdobje 22. mesecev z internetne strani www.revija-reporter.si. Za vsako novico smo imeli točen podatek kolikokrat je bila posamezna novica prebrana. V pozitiven razred smo izbrali 20% najbolj branih novic, v negativni pa 70% najmanj branih novic.

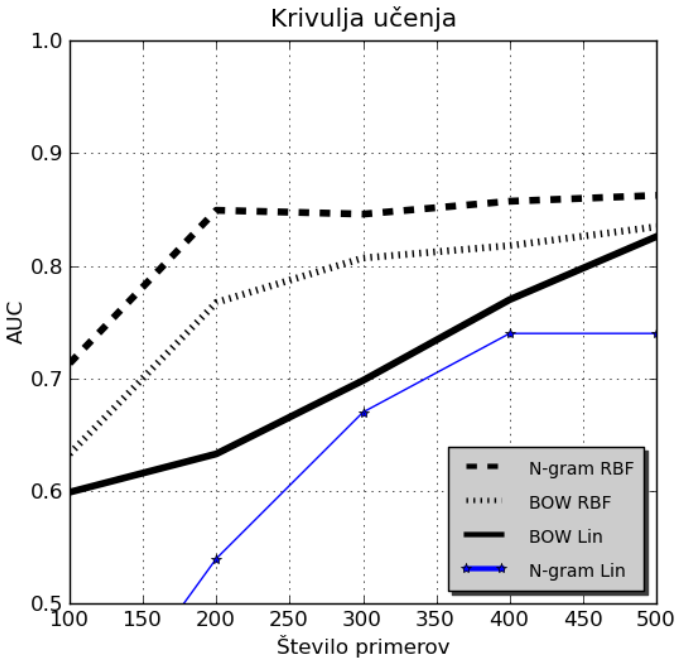
4.2 Metodologija vrednotenja

Naša metoda vrednotenja klasifikacijskega modela se imenuje krivulja učenja. Krivulja učenja nam pove, kako hitro se dani klasifikator nauči dobre klasifikacije in katera je njegova najvišja dosežena vrednost. Naš cilj je, da smo sposobni že na majhni množici učni primerov napovedovati najbolj brane novice v bližnji prihodnosti. Za dosego našega cilja smo zato potrebovali model, katerega krivulja učenja bo že pri majhnih učnih množicah visoka. Lotili smo se iskanja krivulje učenja za vse 4 možne kombinacije jeder, kjer je vsako jedro dobilo 500 naključnih novic. Za vsako kombinacijo smo nato vzeli povprečno krivuljo po petih zagonih.

Na sliki (4.1) vidimo, kako hitro se naučijo posamezne kombinacije jeder.

4.3 Rezultati

Naslednji korak pri naših eksperimentih je bil preučiti, kako dobro jedra klasificirajo naš prvotno zastavljeni problem. Zastavili smo si sledečo obliko testiranja. Vzeli smo učno množico, ki vsebuje novice zadnjih dveh mesecev (200



Slika 4.1: Krivulja učenja za posamezne kombinacije SVM in znakovnih jeder.

novic) in pri tem upoštevali, da novice, stare le nekaj dni še niso dosegle svoje najvišje branosti. Zato smo se odločili, da v učno množico ne vključimo novic starih manj kot 4 dni. Testna množica pa vsebuje novice naslednjega meseca (100 novic). Novice so sortirane po datumu in za vsako naslednjo iteracijo se pomaknemo za 2 tedna naprej. Opisani test smo izvedli z vsemi kombinacijami jeder. Gibanja krivulj za posamezna jedra so vidna na sliki (4.1).

	AUC	
	Povprečje	Stand. odklon
N-gram RBF	**0.85**	**0.07**
BOW RBF	0.82	0.06
N-gram LIN	0.78	0.19
BOW LIN	0.75	0.18

Tabela 4.1: Vrednosti AUC za posamezna jedra.

Naš naslednji in najbolj zahtevan test klasifikatorja je bil, da smo videli, koliko je sposoben na podlagi poljubno velike učne množice napovedati, katere

izmed novic bodo v naslednjih dneh postale najbolj brane. Testna množica je tako velika le 10 novic in kot pri prejšnjem testu, smo tudi tu izključili novice stare manj kot 4 dni. Za ta test smo se odločili vzeti najboljšo kombinacijo jedra iz prejšnjega testa (N-gram RBF). Rezultati so prikazani v tabeli (4.2).

Velikost učne množice	**AUC**	
	Povprečje	**Stand. Odklon**
40	0.65	0.29
50	0.71	0.26
70	0.76	0.22
100	0.78	0.21
120	0.79	0.20
150	**0.80**	**0.19**
200	0.81	0.19
300	0.82	0.19
400	0.81	0.20
800	**0.84**	**0.19**

Tabela 4.2: Rezultati klasificiranja za naslednja 2 dni, za različno velike učne množice.

4.4 Diskusija

Krivulja učenja (4.1) nam lepo prikazuje prednosti jedra RBF pri klasifikaciji novic na majhni učni množici, saj je njegova hiperravnina veliko bolj prilagodljiva za naš kompleksen problem. Linearno jedro ima predvsem problem določiti hiperravnino, ki bo dovolj dobro razdelila oba razreda. Problem neprilagodljivosti se vidi predvsem v visokem standardnem odklonu, ta pa je visok kljub veliki učni množici.

Ko primerjamo znakovna jedra vidimo, da smo pri jedru z n-grami dobili veliko boljše rezultate, kot pri BOW jedru . Naši rezultati so tako potrdili dosedanje primerjave, kjer je jedro z n-gram kljub svojemu znakovnemu pristopu dosegel boljše rezultate kot jedro BOW. To prednost pripisujemo predvsem njegovi značilnosti, ki zajema semantične informacije besedila. Jedro BOW izgubi vso informacijo o zaporedju besed, ko zanj oblikujemo množico besed. TF-IDF je sicer zelo pripomogel k povečanju zmogljivosti jedra BOW,

tako da je izpostavil bolj relevantne besede, vendar še vedno premalo v primerjavi z jedrom z n-grami.

Iz krivulje učenja smo sklepali, da bo SVM z jedrom RBF in jedrom z n-grami jedro tudi v realnem testu dosegel najboljše rezultate. Naši sklepi so se potrdili in kombinacija obeh se je izkazal kot najboljši izbor za klasifikacijo novic. Iz tabele (4.2) lahko vidimo, da lahko s sorazmeroma malo novicami dosežemo dobro napoved. Pri učni množici, ki je velika 150 novic smo dobili $AUC = 0.8$, kar je zelo dobro za praktične napovedi.

Poglavje 5

Sklepne ugotovitve

V diplomski nalogi smo prikazali, kako ustvariti model podpornih vektorjev in znakovna jedra za napovedovanje najbolj branih novic. Našo pozornost smo namenili predvsem jedru BOW in jedru z n-grami, ki se razlikujeta glede na nivo obravnave besedila. Jedro BOW vidi besedilo kot množico besed, medtem ko jedro z n-grami vidi le zaporedje znakov.

Iz naših rezultatov vidimo, da smo z jedrom RBF in jedrom z n-grami sposobni zanesljivo napovedovati, katera izmed današnjih novic bo postala najbolj brana.

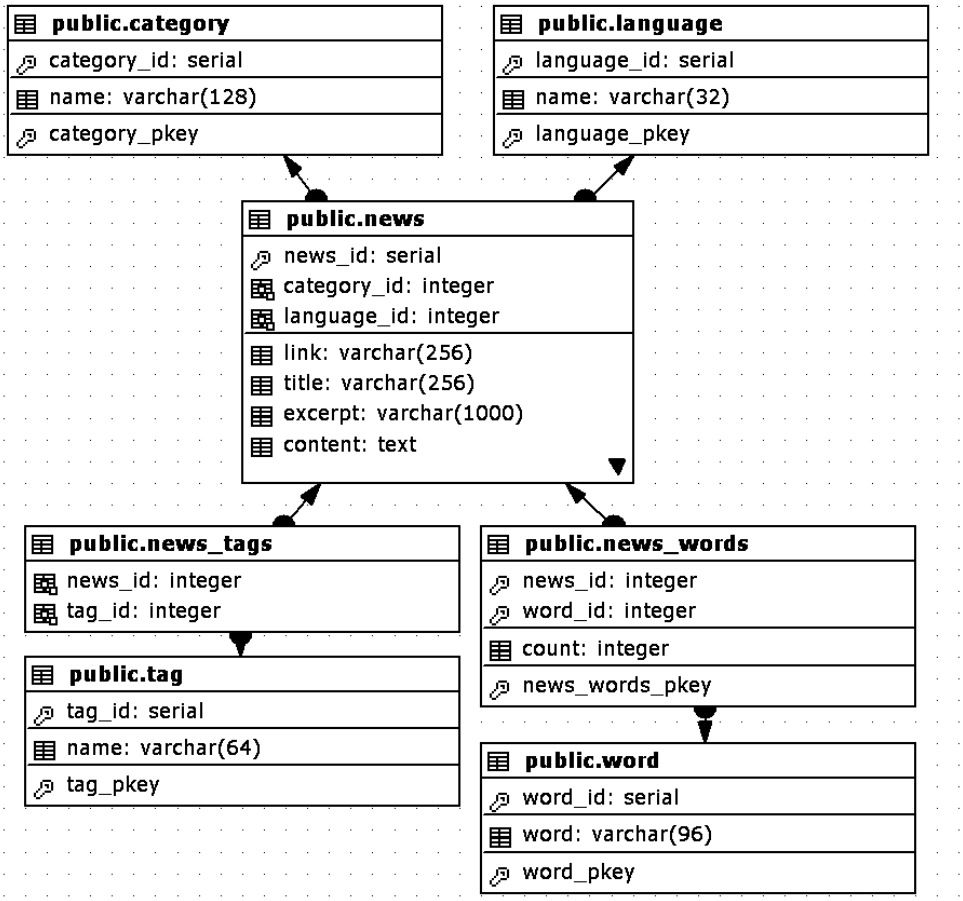
Izbrani pristop klasificiranja se lahko uporabi za iskanje zakonitosti v poljubnem besedilu, saj je naš pristop splošno uporaben za vse vrste klasifikacije besedila. Predstavlja zanesljivo izbiro tudi zato, ker so izbrane kombinacije jeder s tehniko podpornih vektorjev, dokazale svojo uporabnost tudi pri mnogih drugih problemih.

Znanje o tem katere novice bodo postale najbolj brane, je lahko uporabno za oglaševalce na internetnih portalih. Oglaševalec bi lahko zakupil oglasni prostor na novicah, ki bodo postale najbolj brane in se navezujejo na njegov produkt. Prav tako nudi posamezniku že danes znanje o tem, katere novice bodo vplivale na javno mnenje. Novinarjem nudi večdnevno prednost, da pripravijo novice, ki bodo v prihodnje zanimale širšo javnost.

Za nadaljnji razvoj sistema za napovedovanje novic, bi lahko poskusili še vpliv SSK in W-S znakovnih jeder. Oba obravnavata besedilo na bolj napreden in kompleksen način, ter sta opisana v tej diplomski nalogi. Mislim, da bi lahko predvsem z W-S jedrom lahko dosegli zanimive rezultate, saj zna zelo dobro zajeti semantične informacije besedila.

Dodatek A

Model podatkovne baze



Slika A.1: Shema baze novic.

Dodatek B

Izbor BOW besed

```
  count = kolikokrat vsaj se mora novici pojaviti v celotni zbirki
  limit = koliko sta lahko množici A in B veliki
 1 def getBow (count, limit):
 2     || A = Besede, ki se največkrat pojavijo v bazi
 3     most_freq_words ← db.execute ('''
 4               select word_id, sum(count) as s from   news_words
 5               group by word_id order by sum(count) desc
 6               limit %s''' % (limit)).fetchall ()

 8     || B = Besede, ki so v najmanj dokumentih
 9     low_documents_words ← db.execute ('''
10               select word_id, count(*) as c
11               from   news_words
12               group by word_id having count(*) > %s
13               order by count(*) asc limit %s''' %
           (count, limit)).fetchall ()
14     || Presek med A in B
15     most_freq_words ← [w for w, s ∈ most_freq_words]
16     return dict ([(w, c) for w, c ∈ low_documents_words if w ∈ most_freq_words])
```

```
 1
       def tf_idf (text_word_freq, text_word_count, documents_number, documents_word_count):
 2     return (text_word_freq/float (text_word_count)) ∗
           math.log (documents_number/(float (documents_word_count) + 1), 2)
```

Dodatek C

Izračun BOW matrike za libsvm

```
1 sample_length ← 400   # Število novic
2 count, limit ← 13, 4500

4 db ← dbConnect.getDBConnection ()
5 kernel_words ← getBow (count, limit)   # izbor besed opisan v točki: 3.2.1
6 kernel_words_length ← len (kernel_words)
7 all_news ← testData.getAllData (sample_length, random ← False)
8 news ← {}, svm_data ← [ ]
```

Za vsako novico izračunaj njen TF-IDF: točka 2.5

```
11 for n ∈ all_news:
12     news_words ← n.words   # bow novice
13     news_words ← dict ([(nw.word_id, nw.count) for nw ∈ news_words])
14     news_words_length ← len (news_words)

16     if news_words_length > 0:
17         ‖ Vector velikosti, ki ga vrne getBow s TF-IDF frek. novice
18         news_kernel_words ←
                   np.array ([tf_idf (news_words.get (w, 0), news_words_length,
18                   kernel_words_length, kernel_words[w]) for w ∈
                           kernel_words], dtype ← 'float32')
19         ‖ Skaliranje
20         news_kernel_words ← news_kernel_words/sum (news_kernel_words)
21         svm_data.append ((int (n.class_id), map (float, news_kernel_words)))
```

Dodatek D

Izračun N-gram matrike za libsvm

```
2 k ← 8   # dolžina n-grama
3 sample_length ← 300
4 db ← getDBConnection()
5 news ← testData.getAllData(sample_length, random ← False)
6 news_data ← {}
7 svm_data ← []
```

1. Izračun frekvence vseh n-gramov v zbirki in n-grami posamezne novice

```
10 all_ngrams ← {}

12 for n ∈ news:
13     || normaliziran, stop-words
14     text ← n.content_normalized
15     if len(text) > 300:
16         || N-grami novice
17         kmers ← [text[i: i + k] for i ∈ range(len(text) − k + 1)]
18         d ← dict([(i, 0) for i ∈ set(kmers)])
```

```
20          for i ∈ kmers:
21              d[i]+ ← 1
22              if all_ngrams.has_key (i):
23                  all_ngrams[i]+ ← 1
24              else:
25                  all_ngrams[i] ← 1

27          news_data[n.news_id] ← d
```

2. izračun približka opisan v točki 2.7.1

```
30 sgrams ← all_ngrams.items ()
31 sgrams.sort (lambda x, y: −cmp (x[1], y[1]))
32 sgrams ← sgrams[: 6000]
33 d ← dict ([(i[0], 0) for i ∈ sgrams])
```

3. Oblikovanje vektorja besed za posamezno novico

```
36 news_produkt ← {}

38 for n ∈ news:
39      if n.news_id ∈ news_data:
40          ngrams ← news_data[n.news_id]
41          ‖ Skaliranje
42          scaling_data ← [ngrams.get (i, 0) for i ∈ d]
43          suma ← float (sum (scaling_data))
44          news_produkt[n.news_id] ← [i/suma for i ∈ scaling_data]
45          svm_data.append ((n.class_id, news_produkt[n.news_id]))
```

Dodatek E

Krivulja učenja

```
2 numTests ← 2
3 folds ← 5   # Število foldov pri prečnem primerjanju
4 prop ← [0.1, 0.2, 0.3, 0.4, 0.6, 1.0]   # Deleži podatkov
5 C, gamma, weight ← 5, 14, [1, 4]   # SVM parametri

7 for p ∈ prop:
8     auc_orange_array ← np.array ([ ])
9     || razdeli proporcionale podatke na folde
10    prop_data ← svm_data[: int (len (svm_data) ∗ p)]
11    slices ← [prop_data[i : : folds] for i ∈ xrange (folds)]

13    for fold ∈ range (folds):
14        || pripravi učne in testne podatke za posamezni fold
15        test_data ← slices[fold]
16        train_data ← [i for slice ∈ slices if slice ≠ test_data for i ∈ slice]
17        train_labels ← [i[0] for i ∈ train_data]
18        train_samples ← [i[1] for i ∈ train_data]

20        orange_results ← [ ]
```

```
21          || IZBIRA JEDRA:
22          if numTests = 1:   # Linearno jedro
23              param ← svm_parameter (kernel_type ← LINEAR, C ←
                        C, nr_weight ← 2,
24                      weight_label ← [−1, 1], weight ← weight, probability ← 1)
25          elif numTests = 2:   # RBF jedro
26              param ← svm_parameter (kernel_type ← RBF, C ← C, gamma ←
                        gamma,
27                      nr_weight ← 2, weight_label ← [−1, 1], weight ←
                                weight, probability ← 1)

29          || Zgradba SVM modela
30          problem ← svm_problem (train_labels, train_samples)
31          model ← svm_model (problem, param)
32          ex ← orngTest.ExperimentResults (2, [' '], [1, −1], −1)

34          || Testiranje modela na testnih podatkih
35          for i ∈ test_data:
36              probability ← model.predict_probability (i[1])
37              || shrani predvidevanje
38              t ← orngTest.TestedExample (0, i[0])
39              t.probabilities ← [[probability[1][−1], probability[1][1]]]
40              ex.results.append (t)

42          auc_orange_array ← np.append (auc_orange_array, orngStat.AUC (ex)[0])

44      || Izpiši mere
45      results ← [round (i, 3) for i ∈ auc_orange_array.mean ()]
46      toPrint ← map (str, [p, len (svm_data) ∗ p, C, weight, gamma] + results)
```

Dodatek F

Realni primer

Velikost učnih in testnih podatkov, ter velikost koraka

```
3 train_size, test_size, step_size ← 200, 10, 10
4 gap ← 16   # kolikšen je prepad med testnimi in učnimi podatki

6 for index ∈ range (0, 3000, step_size):
7     if index + train_size + test_size < len (svm_data):
8         || Razdeli podatke na učne in testne
9         train_data ← svm_data[index: index + train_size]
10        test_data ← svm_data[index + train_size + gap: index + (train_size +
              test_size + gap)]
11        train_labels ← [i[0] for i ∈ train_data]
12        train_samples ← [i[1] for i ∈ train_data]

14        || IZBIRA JEDRA:
15        if numTests = 1:   # Linearno jedro
16            param ← svm_parameter (kernel_type ← LINEAR, C ←
                  C, nr_weight ← 2,
17                weight_label ← [−1, 1], weight ← weight, probability ← 1)
18        elif numTests = 2:   # RBF jedro
19            param ← svm_parameter (kernel_type ← RBF, C ← C, gamma ←
                  gamma,
20                nr_weight ← 2, weight_label ← [−1, 1], weight ←
                      weight, probability ← 1)
```

```
22      || Zgradba SVM modela
23      problem ← svm_problem (train_labels, train_samples)
24      model ← svm_model (problem, param)
25      ex ← orngTest.ExperimentResults (2, [' '], [1, −1], −1)

27      || Testiranje modela na testnih podatkih
28      for i ∈ test_data:
29          probability ← model.predict_probability (i[1])
30          || shrani predvidevanje
31          prediction_results.append ((i[0], probability))
32          t ← orngTest.TestedExample (0, i[0])
33          t.probabilities ← [[probability[1][−1], probability[1][1]]]
34          ex.results.append (t)

36      test_poz ← [i for i ∈ test_data if i[0] = 1]
37      test_neg ← [i for i ∈ test_data if i[0] = −1]

39      results ← [round (i, 3) for i ∈ orngStat.AUC (ex)[0]]
40      toPrint ← map (str, [index, index +
                train_size, len (test_poz), len (test_neg), C, weight] + results)

42      print toPrint
```

Literatura

[1] N. Cancedda, E. Gaussier, C. Goutte, J.-M. Renders, "Word-Sequence Kernels" v Journal of Machine Learning Research 3 (2003) 1059-1082.

[2] C. Chang, C. Lin, LIBSVM: knjižnica za tehniko podpornih vektorjev Dostopna na: http://www.csie.ntu.edu.tw/ cjlin/libsvm

[3] N. Cristianini, J. Shawe-Taylor, "Support Vector Machines" v Cambridge University Press, Cambridge, UK, 2000.

[4] T. Fawcett, "An introduction to ROC analysis" v Pattern Recognition Letters 27 (2006) 861–874.

[5] C. Hsu, C. Chang, C. Lin, "A Practical Guide to Support Vector Classification" Dostopno na:
http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf

[6] B. Li, J. Hu, K. Hirasawa, "Support Vector Machine Classifier with WHM Offset for Unbalanced Data" v Journal of Advanced Computational Intelligence 95 and Intelligent Informatics Vol.12 No.1, 2008.

[7] H. Lodhi, C. Saunders, J. S. Taylor, N. Cristianini, C. Watkins, "Text Classification using String Kernels" v Journal of Machine Learning Research 2 (2002) 419-444.