Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Marko Vidoni

Napovedovanje obsega komentiranja spletnih novic z modeli strojnega učenja

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Blaž Zupan

Ljubljana 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V nalogi preučite uspešnost modelov, ki za dano spletno novico napovedo število spletnih komentarjev. V namene gradnje modela pridobite ustrezne podatke iz izbranega slovenskega spletnega novičarskega medija. Podatke z besedilom novice in spremljajočimi atributi ustrezno predobdelajte. Predlagajte in raziščite uspešnost metod strojnega učenja za ta regresijski problem.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Marko Vidoni sem avtor diplomskega dela z naslovom:

Napovedovanje obsega komentiranja spletnih novic z modeli strojnega učenja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal 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 na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 7. junija 2016 Podpis avtorja:

Najlepše se zahvaljujem svojemu mentorju prof. dr. Blažu Zupanu za vse strokovne nasvete in komentarje pri izdelavi diplomske naloge. Zahvalil bi se rad tudi svoji družini za vso podporo med celotnim študijem in pri pisanju tega diplomskega dela.

Kazalo

Povzetek

Svetovni splet nas spremlja na vsakem koraku in si življenja brez stalnega dostopa do spleta ne znamo več predstavljati. Na splet se je v skladu s tem preselila tudi objava ter branje dnevnih novic in drugih člankov. Pomembno lastnost, ki jo ima na spletu objavljena novica, in jo v preteklosti tiskani mediji niso imeli, je možnost izražanja mnenja bralcev s komentarji, spisanih na temo novice. Spletnim portalom je tako v interesu, da so objavljene novice čim bolj komentirane in s tem tudi bolj brane, to pa posledično pripelje do večjega obiska strani. V diplomski nalogi je bil razvit napovedni sistem, s katerim lahko na podlagi besedila novic in dodatnih meta podatkov napovemo število komentarjev, ki jih bo prejela spletna novica v slovenščini. V našem primeru se je najbolje odrezala uporaba odebeljenega dela besedila novice, dodatnih meta podatkov in nadgradnja modela gradientnega boostinga regresijskih dreves, ki je gradila ločene modele za vsako kategorijo novic. Z analizo smo dokazali, da je s pravo predpripravo podatkov in uporabo naprednejših tehnik strojnega učenja mogoče uspešno napovedati število komentarjev novic. Poleg modelov samih in njihovega testiranja je bil proučen tudi vpliv značilk na napoved količine komentarjev in lastnosti novic z dobrimi in slabimi napovedmi.

Ključne besede: tekstovno rudarjenje, regresijski model, spletne novice.

Abstract

World Wide Web accompanies us on every step of our lives and we cannot anymore imagine our lives without the constant access to the internet. Publishing industry has also moved online where news articles are now published and read. The important novelty of online articles is the ability of readers to express their opinions about articles' topics in a form of comments. It is in the best interest of web portals that the published web news are frequently commented and read, driving up the web portal visitors traffic. In this thesis a prediction system has been developed to predict the number of comments a news article in Slovene language will generate based on news text content and metadata. We got the best prediction results from bold parts of the text, coupled with metadata attributes and the extended gradient boosted regression trees model which builds separate models for each article category. Our analysis has proven that with proper data preprocessing and use of machine learning techniques it is possible to successfully predict the number of comments an article gets. In addition we also studied an influence of features on the prediction and properties of articles with good and bad prediction results.

Keywords: text mining, regression model, web news articles.

Poglavje 1

Uvod

V današnjem času se vedno več novic namesto po utečenih kanalih, kot so recimo časopisi in revije, objavlja tudi na spletu. Vedno več ljudi se prav tako poslužuje spletnih portalov z novicami za svoje informiranje o dogajanju po svetu. Tako bralci vedno več svojega branja po novem opravljajo na spletu. Spletna ponudba vsebin nam poleg prikladnosti omogoča tudi še zmožnost medsebojnega komuniciranja in deljenja mnenj s strani bralcev, recimo o tematiki neke novice. Danes je na spletnih portalih skoraj pravilo, da je poleg osnovne vsebine omogočena tudi izmenjava mnenj bralcev v obliki njihovih komentarjev.

Zelo hitro lahko ugotovimo, da nekatere novice večje število ljudi pripravijo do tega, da s svetom delijo svoja mnenja o neki tematiki. Število komentarjev je tako lahko tudi nekakšen pokazatelj, kakšno zanimanje obstaja za neko dotično novico in posledično za tematiko, na katero se le-ta nanaša. Verjetno je namreč nek obiskovalec portala, ki se je odločil, da bo komentiral dotično novico, le-to tudi vsaj do neke mere prebral, tako da ima vsaj neko osnovno predstavo, kaj je vsebina novice, da lahko nato na podlagi le-te izoblikuje in deli svoje mnenje.

Tako se nam postavi smiselno vprašanje, ali bi se dalo glede na podatke, ki opisujejo novice, zanje napovedati število komentarjev, ki jih bo posamezna novica pridobila. Poleg raznih standardnih podatkov, kot so recimo dan in

čas objave, so nam v primeru novic na voljo tudi besedila le-teh. Tekstovni podatki pa niso v svoji osnovni obliki primerni za uporabo, temveč jih je treba vnaprej predobdelati. Za napoved v tem primeru uporabimo različne regresijske tehnike, s katerimi napovemo število komentarjev.

Z napovedjo na podlagi tekstovnih podatkov in z regresijskimi tehnikami so se v različnih problemskih domenah ukvarjali že mnogi avtorji. Napisanih je bilo kar nekaj člankov, ki na podlagi raznih finančnih poročil in drugih besedil, povezanih z gospodarstvom, poskušajo napovedovati na primer gibanje raznih ekonomskih kazalcev. V članku [6] so tako avtorji na podlagi poročil ocenjevali ceno japonskih državnih obveznic. Avtorji članka [14] so na podlagi tekstovnih podatkov, pridobljenih iz novic, v kombinaciji z regresijskimi tehnikami napovedovali ceno nafte. Reševanje takšnega problema je lahko zelo uporabno, saj zaradi velikega nihanja cene nafte lahko pride tudi do kritičnega gibanja cen, ki pa bi jih z uporabo takšnega sistema lahko vnaprej predvideli in primerno odreagirali. Članek [8] v domeni delniškega trgovanja na podlagi teksta napoveduje volatilnost, ki se uporablja za merjenje negotovosti.

Z nekoliko drugačno domeno regresijskega tekstovnega rudarjenja so se ukvarjali avtorji članka [11], kjer so na podlagi teksta napovedovali starost osebe, ki je bila avtor le-tega. Poleg tega so tudi z dodatno analizo ugotavljali način izražanja različno starih oseb.

Še najbolj se naši problemski domeni približajo naslednji članki, ki napovedujejo število komentarjev in popularnost člankov. Tako članek [12] napoveduje z uporabo klasifikacije in tako tekstovnih podatkov, kot tudi raznih dodatnih podatkov, ali bo neka novica dobila veliko ali pa malo komentarjev. Prav tako so se, sicer nekoliko drugače, lotili klasifikacijske napovedi števila komentarjev glede na besedilo v članku [15]. Napovedovali so namreč, če bo neka novica dobila nadpovprečno ali podpovprečno število komentarjev. V članku [13] pa so avtorji napovedovali popularnost člankov na podlagi samo netekstovnih podatkov, kot so datum in ura objave, število obiskovalcev določene spletne strani z novico in število pojavitev spletne povezave do

novice na različnih družbenih omrežjih.

V diplomskem delu smo se osredotočili na uporabo tako tekstovnih atributov, kot tudi raznih dodatnih pomožnih atributov, na podlagi katerih smo nato s pomočjo različnih regresijskih tehnik skušali napovedati število komentarjev, ki jih posamezna novica pridobi. Poleg tega pa je pomembna razlika napram omenjeni literaturi ta, da smo v našem primeru analizo izvajali na besedilih, zapisanih v slovenskem jeziku in ne v angleškem, kot je prevladujoča navada v literaturi.

V nadaljevanju bomo najprej v drugem poglavju predstavili različne tehnike predobdelave in priprave predvsem tekstovnih podatkov, poleg tega pa bomo predstavili teoretično ozadje regresijskih tehnik, ki smo jih uporabili, ter opisali metode ocenjevanja uspešnosti napovednih modelov. V tretjem poglavju predstavimo podatke, ki smo jih uporabili. Sledi četrto poglavje, v katerem predstavimo razne poskuse in ocene uspešnosti napovedovanja in v njih uporabljene metode. Nato predstavimo in komentiramo pridobljene rezultate. Za konec pa še povzamemo naše pomembnejše ugotovitve diplomskega dela ter predstavimo potencialne možnosti za izboljšave in razširitev.

Poglavje 2

Tehnike strojnega učenja za analizo besedil

V tem poglavju najprej s teoretičnega stališča predstavimo metode predobdelave tekstovnih podatkov, saj, kakor bo razvidno v naslednjem poglavju, ki je namenjeno opisu naših podatkov, osnovna oblika besedil oziroma vsebine novic ni primerna za našo analizo. Nadalje predstavimo tudi regresijske algoritme, ki smo jih uporabili in testirali na naših podatkih ter z njimi skušali dobiti čim boljše natančnosti napovedi števila komentarjev.

2.1 Predobdelava podatkov

Za gradnjo napovednih modelov s tehnikami strojnega učenja potrebujemo vhodne, tudi besedilne podatke, predstaviti v atributni obliki, to je z matriko, kjer vrstice predstavljajo posamezne primere, pri nas so to posamezne novice, stolpci pa so različni atributi, ki opisujejo naše primere. Elementi takšne matrike so števila.

2.1.1 Lematizacija besedila

Ko imamo opravka z besedili v osnovni neobdelani obliki, nastane problem, ko se v besedilu pojavljajo besede z istim ali podobnim pomenom, ki pa

so različno zapisane. Ljudem v takem primeru ugotavljanje podobnosti ponavadi ne povzroča večjih težav. Edino mogoče v primeru, ko so besede dvoumne in imajo isti zapisi več pomenov, pa še takrat si pogosto pomagamo s kontekstom in si tako razjasnimo pomen. Drugače pa je v primeru računalnikov. Za le-te je namreč posamezna beseda le niz črk, ki so v tem primeru, ko gre za različno napisane besede, ki pa v končnem nosijo isti pomen, drugače zapisane, saj imajo recimo različno končnico. Računalnik namreč v osnovi z uporabo postopkov predstavitve besedil, ki so opisani kasneje, tretira vsako različno zapisano besedo kot različno.

Ta problem je še bolj pereč v slovenskem jeziku kot v angleškem jeziku. V slovenskem jeziku imajo posamezne osnovne oblike besed še več možnih izpeljank, saj poleg množine poznamo še različne oblike za različne sklone in spregane oblike besed [9].

Zato je pogosto priporočljivo za nadaljnjo strojno analizo besedil najprej le-ta predobdelati, tako da vse besede spravimo v njihove leme oziroma normalizirane oblike v katerih se recimo besede nahajajo tudi v slovarju kot slovarska gesla [7]. Kot je prej omenjeno, to za računalnike ni ravno trivialna naloga, še več, njena težavnost je tudi odvisna od jezika, v katerem je besedilo, za katerega želimo ugotoviti lematizirane oblike besed. Pristop k temu problemu ponavadi vsebuje uporabo raznih pravil in slovarjev, pri tem pa, enako kot lahko tudi pri ljudeh, povzroča probleme dvoumnost besed [9]. Pri tem si lahko pomagamo s poznavanjem besedne vrste in vloge dotične besede v stavku, vendar pa je s primernimi tehnikami [7] mogoče dobiti dobre in natančne rezultate tudi brez poznavanja prej omenjenih besednih vrst in vlog posameznih besed. Dobra stran takšnih tehnik, ki ne potrebujejo določenih besednih vrst besed je tudi ta, da jih lahko uporabljamo na posameznih skupkih besed in tako ne rabimo celotnih stavkov.

V naši diplomski nalogi smo uporabili orodje LemmaGen [7], ki nam je omogočilo obdelavo slovenskega jezika. Orodje deluje z uporabo pravil, ki so podana v drevesni obliki - torej imamo neke vrste odločitveno drevo. Baza pravil, ki jih orodje potrebuje za uspešno procesiranje besed, je že

predpripravljena s strani avtorjev programa in je vključena v programsko orodje. Orodje deluje tako, da se premika globlje po vozliščih drevesa, kjer je vsako vozlišče eno pravilo. Ko doseže končno vozlišče na podlagi obiskanih vozlišč oziroma pravil, preoblikuje dano besedo v njeno lematizirano obliko.

Nadalje si poglejmo zgoraj predstavljeni bolj teoretični opis delovanja lematizacije na realnem primeru. Za neobdelano obliko besedila si vzemimo naslednjo poved: *Ker danes sije sonce, bom vzel kolo in se odpeljal na vrh bližnjega hriba.* Z izvedbo postopka lematizacije pa se nam poved nekoliko spremeni.

Osnovna oblika Ker danes sije sonce, bom vzel kolo in se odpeljal na vrh bližnjega hriba.

Lematizirana oblika Ker danes sijati sonce, biti vzeti kolo in se odpeljati na vrh bližnji hrib.

Kot lahko vidimo, se je nekaj besed spremenilo, saj so sedaj besede v svojih lematiziranih oblikah. Tukaj lahko vidimo že prej omenjeno moč tega postopka, saj se nam bi recimo zadnji del povedi tudi, če bi bil v katerem drugem sklonu, še vedno po izvedbi postopka spremenil v isto obliko, in sicer *bližnji hrib*. Tako dobimo iz različno zapisanih besed in za računalnik v osnovi popolnoma različne besede, ki pa imajo v resnici isti ali pa podoben pomen in osnovo, po izvedbi postopka lematizacije iste besede.

2.1.2 Število pojavitev besedilnih elementov

Tekstovno informacijo moramo preoblikovati tako, da dobimo številski zapis, ki ga lahko spravimo v prej opisano matrično obliko. To lahko v problemski domeni, kot je naša, kjer imamo opravka z besedili, storimo tako, da štejemo število pojavitev različnih tekstovnih elementov. Med metode sodi tako štetje pojavitev besed v celotnem besedilu, ta postopek se v angleščini imenuje tudi *BOW* (bag of words)[10] oziroma *term frequency* (označeno tudi z *TF*).

Za ponazoritev postopka smo si izbrali dve povedi:

- Ker danes sije toplo sonce je zunaj toplo
- Ker je danes zunaj toplo bom zunaj igral košarko

Za lažje razumevanje in skladnost z uporabljenim postopkom predpostavimo, da so iz besedila odstranjena vsa ločila in so vse črke male tiskane. Najprej pridobimo unijo vseh besed, ki se pojavljajo v obeh besedilih, torej se v množici vse prisotne besede pojavijo le enkrat brez ponovitev, ki bi se sicer lahko dogodile v posameznih besedilih. V našem primeru je to množica: *ker, danes, sije, sonce, je, zunaj, toplo, bom, igral, košarko.* Ta množica nam tako sestavlja naš vektor vseh besed, za katere nas zanima število pojavitev v naših besedilih.

V spodnji tabeli 2.1 vidimo primer štetja pojavitev posameznih besed v obeh povedih iz našega primera.

Tabela 2.1: Primer štetja pojavitev besed v besedilu.

	ker	danes	sije	sonce	je	zunaj	toplo	bom	igral	košarko
B_1	1	1	1	1	1	1	2	0	0	0
B_2	1	1	0	0	1	2	1	1	1	1

V končnem bi v našem primeru ustvarili matriko M (enačba 2.1), kjer vsaka vrstica predstavlja eno besedilo, stolpci pa posamezne besede iz našega vektorja besed. Na poziciji $M_{v,s}$ se v matriki M nahaja število pojavitev besede, ki je na s-ti poziciji v vektorju vseh besed in v v-tem besedilu po vrsti. Recimo vrednost $M_{1,4}$ je enaka 1, kar nam pove, da je v prvem besedilu ena pojavitev četrte besede iz vektorja vseh besed, kar je beseda *sonce.*

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 2 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.1}$$

Isti pristop lahko nekoliko razširimo, in sicer tako, da namesto posameznih besed raje štejemo število pojavitev večjega skupka zaporednih besed,

imenovanega *terka*. Tako lahko pridobimo tudi nekoliko znanja o celotni strukturi stavkov in ne le samo o številu besed, ki so v besedilu, ne zanima pa nas njihova pozicija in povezava z drugimi besedami, kot je to v prvem primeru.

Ta pristop si oglejmo na istem primeru dveh besedil, ki smo ju imeli prej in določimo, da bomo gledali in šteli pojavitve za dve besedi, ki v besedilu stojita skupaj. Naš vektor elementov, za katere želimo imeti število pojavitev, ne vsebuje več posameznih besed tako kot prej, ampak pare zaporednih besed. Vektor elementov je tako: *(ker, danes), (danes, sije), (sije, toplo), (toplo, sonce), (sonce, je), (je, zunaj), (zunaj, toplo), (ker, je), (je, danes), (danes, zunaj), (toplo, bom), (bom, zunaj), (zunaj, igral), (igral, košarko).* Ko imamo vektor elementov, lahko na isti način kot prej zgradimo matriko M le, da sedaj posamezen stolpec matrike predstavlja število določenega para zaporednih besed. V enačbi 2.2 je predstavljena naša nova matrika M, zgrajena s pristopom štetja pojavitev dveh zaporednih besed. V tej matriki je recimo vrednost $M_{1,4}$ je enaka 1, kar nam tokrat pove, da je v prvem besedilu ena pojavitev četrtega elementa vektorja zaporednih besed, kar je v tem primeru *(sonce, je)*.

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.2}$$

2.1.3 IDF

Poleg zgoraj opisanega postopka ugotavljanja števila pojavitev (frekvenc) elementov v posameznih besedilih, obstaja tudi ugotavljanje inverznega števila besedil, v katerih se nek element pojavi. Prej opisana metoda se ne meni za pogostost nekega elementa v različnih besedilih. Edino, kar ji je pomembno, je sama frekvenca pojavitev znotraj dotičnega besedila. Ta problem odpravlja postopek inverznega števila besedil, ki ravno ugotavlja, kako razširjen je nek element ne znotraj nekega besedila, temveč v celotnem naboru besedil (korpusu)[10].

Za vsak element e se vrednost *IDF* izračuna po enačbi 2.3, kjer nam N predstavlja število vseh besedil v naši zbirki (korpusu), b_e pa število besedil iz naše zbirke, ki vsebujejo vsaj eno pojavitev elementa e.

$$idf_e = log(\frac{N}{b_e}) \tag{2.3}$$

2.1.4 TF-IDF

Če metodo štetja pojavitev elementov združimo z inverznim številom dokumentov dobimo združeni postopek, ki se imenuje *TF-IDF*, kar je kratica za *term frequency–inverse document frequency*. Nasprotno kot pri metodi štetja pojavitev elementov v posameznem besedilu, tukaj odpravimo problem, da so vsi elementi enakovredni, saj dobijo večjo težo tisti elementi, ki se velikokrat pojavljajo v manjšem številu besedil in ne tisti, ki se pojavljajo v velikem številu besedil [10].

Na ta način upoštevamo dodatno tudi dejstvo, da je nek element (recimo beseda) značilen samo za neko besedilo, v katerem se velikokrat pojavi, v preostalih besedilih pa se ta element ne pojavlja. Nasprotno pa je tudi primer, ko se nek element pojavlja recimo v vseh besedilih.

Elementi, ki so pogosti za vsa besedila, nam namreč pri naši napovedi ne pomagajo preveč, saj nam ne pripomorejo pri razločitvi med besedili. Po drugi strani pa nam pri razločitvi med različnimi besedili pomagajo elementi, ki so značilni samo za nekatera besedila iz naše množice besedil. Zato tudi ta metoda da takšnim elementom večjo težo.

Formalno se po opisani metodi za vsak element vrednost izračuna po enačbi 2.4, kjer pomnožimo število pojavitev elementa e v dotičnem besedilu b z inverznim številom dokumentov (idf_e), v katerih se pojavi vsaj enkrat naš trenutni element e.

$$tf - idf_{e,b} = tf_{e,b} \times idf_e \tag{2.4}$$

V naši analizi smo uporabili implementacijo iz knjižnice *scikit-learn*, kjer

pa je, kakor je zapisano v dokumentaciji[1], izračun nekoliko drugačen od splošnega, in sicer enak enačbi 2.5. S prištevanjem enice k idf_e delu dosežemo, da element, ki se pojavi v čisto vseh besedilih, ni popolnoma razvrednoten, vendar ima še vedno neko majhno težo.

$$tf - idf_{e,b} = tf_{e,b} \times (idf_e + 1) \tag{2.5}$$

2.2 Regresijski modeli

Pri našem napovednem problemu skušamo čim bolj točno napovedati število komentarjev, ki jih dobi posamezna novica. Tega problema smo se lotili z uporabo regresijskih modelov. V nadaljevanju so opisane osnovne oblike algoritmov, ki smo jih uporabili. Kot bo razvidno v naslednjem poglavju, pa smo glede na specifiko podatkov naše problemske domene, v nekaterih testih tudi nekoliko priredili osnovne algoritme in tako dobili še nekoliko boljše rezultate.

2.2.1 Regresijska drevesa

Algoritem regresijskih dreves je osnova, na kateri gradijo naši na koncu uporabljeni algoritmi naključnih gozdov in gradientnega boostinga regresijskih dreves. Le-ti namreč v osnovi uporabljajo bolj ali manj normalna regresijska drevesa, jih pa seveda uporabijo na nekoliko bolj napredne načine in v ansamblu, kjer je za končno napoved odgovornih več dreves. Podrobnosti algoritmov so opisane v naslednjih podpoglavjih, v tem pa se osredotočimo na osnovo, ki jo predstavljajo regresijska drevesa.

Regresijska drevesa delujejo tako, da atributni prostor razdelijo na regije R, znotraj katerih nato napovedo neko konstantno vrednost [5]. Ta konstantna vrednost je v primeru regresijskih dreves kar povprečje iskane spremenljivke vseh učnih primerov, ki padejo pod nek dotični list drevesa

[1] `http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html`

(končno vozlišče, ki nima potomcev), oziroma regijo prostora. Formalno zapisan postopek napovedi za nek primer vidimo v enačbi 2.6, kjer je c_m označena konstantna povprečna vrednost, ki pripada m-temu listu drevesa. $I(x \in R_m)$ nam predstavlja indikatorsko funkcijo, ki ima vrednost 1 takrat, ko x je element neke zahtevane množice R_m in 0 takrat, ko le-ta ni element zahtevane množice. Tako z uporabo indikatorske funkcije, v končnem glede na enačbo 2.6 za nek dotični testni primer x pogledamo, v katero regijo spada in mu za vrednost iskane spremenljivke napovemo vrednost c_m, ki pripada ugotovljeni m-ti regiji.

$$F(x) = \sum_{m=1}^{M} c_m I(x \in R_m) \tag{2.6}$$

Sedaj pa se še posvetimo nekoliko postopku gradnje samega regresijskega drevesa. V vsakem vozlišču, ki se ga odločimo razvejiti v dva njegova otroka, je potrebno ugotoviti dva podatka, in sicer atribut, glede na katerega bomo razvejili drevo v dva potomca in prag oziroma vrednost izbranega atributa, ki bo predstavljala mejo med levim in desnim vozliščem, ki je potomec trenutno razvijajočega vozlišča. V vsakem vozlišču se odločimo za tisto kombinacijo atributa in njegove razmejitvene vrednosti, ki nam minimizira neko izbrano vrednost nečistoče. Za vse atribute pregledamo vse možne razmejitve glede na možne vrednosti dotičnih atributov v učni množici [5]. Glede na dokumentacijo[2] se kot merilo nečistoče v uporabljeni knjižnici za programski jezik Python *scikit-learn* uporablja *srednja kvadratna napaka* (enačba 2.7). Glede na omenjeno enačbo v vozlišču za vsak primer i primerjamo njegovo vrednost iskane spremenljivke (označena z y_i) s povprečno vrednostjo iskane spremenljivke vseh primerov, ki spadajo v dotično vozlišče.

$$H(X_m) = \frac{1}{N_m} \sum_{i=1}^{N_m} \left(y_i - \left(\frac{1}{N_m} \sum_{i2=1}^{N_m} y_{i2} \right) \right)^2 \tag{2.7}$$

V končnem v nekem vozlišču, ki ga želimo razvejiti, izberemo tisti atribut in prag, ki nam minimizira vsoto nečistoč v levem in v desnem vozlišču, ki sta

[2] `http://scikit-learn.org/stable/modules/tree.html`

potomca in nastaneta z razvejitvijo. Glede na dokumentacijo[3] je ta vsota še nekoliko obtežena z deležem primerov, ki pripadejo vsakemu od otrok (enačba 2.8). Z X_l in X_d so označeni primeri, ki glede na njihovo vrednost izbranega delitvenega atributa pripadejo, prvi levemu otroku in drugi desnemu otroku trenutnega vozlišča. Z n_l in n_d pa je označeno število primerov, ki padejo pod levega in desnega otroka.

$$\frac{n_l}{N_m}H(X_l) + \frac{n_d}{N_m}H(X_d) \tag{2.8}$$

Primere nato porazdelimo med nastala otroka glede na njihovo vrednost izbranega atributa, in sicer, če je za nek dotični primer vrednost izbranega atributa manjša ali enaka vrednosti določene razmejitvene vrednosti, potem gre primer v levega otroka, sicer pa gre v desnega.

Opisani postopek gradnje drevesa se rekurzivno nadaljuje dokler ne pridemo do ustavitvenega pogoja. Le-ta je lahko v skrajnem primeru ta, da imamo v listu samo še en sam primer. Obstajajo pa še tudi drugi ustavitveni pogoji, kot je recimo omejitev na maksimalno globino drevesa ali pa omejitev minimalnega števila primerov, ki jih še neko vozlišče lahko ima, da ga razvejimo naprej na njegove potomce in se ne ustavimo pri njem in ga razglasimo za list drevesa.

2.2.2 Naključni gozdovi

Metoda naključnih gozdov spada v skupino ansambelskih metod. Zanjo je značilno, kot je opisano v [2], da deluje na ideji, da ločeno zgradimo večjo množico dreves (v našem primeru regresijskih dreves) in nato za končno napoved uporabimo povprečje napovedi množice dreves.

Ideja algoritma, kot je opisano v [5] je, da se v vsaki iteraciji zanke nauči novo, v našem primeru regresijsko drevo. Že prva posebnost algoritma je ta, da je vsako tako drevo naučeno na nekoliko drugačnih podatkih kot prejšnje. Algoritem namreč v vsaki iteraciji iz celotne osnovne učne množice za tre-

[3] `http://scikit-learn.org/stable/modules/tree.html`

nutno učno množico naključno z vračanjem izbere podmnožico primerov. Takemu načinu zbiranja oziroma vzorčenja se reče tudi *bootstrap sampling*. Na tako izbrani učni množici nato naučimo naše trenutno regresijsko drevo. Tukaj pa je zopet pomembna razlika, saj postopek gradnje drevesa ni popolnoma standarden. Ko gradimo drevo v vsakem vozlišču, ki ga nameravamo razširiti v njegova otroka, za določitev atributa, na podlagi katerega razdelimo primere, ne izbiramo med vsemi atributi, ampak samo med naključno izbrano podmnožico vseh atributov. V vsakem vozlišču tako vsakič znova naključno izberemo podmnožico vseh atributov in glede na le-to nato izberemo najbolj primeren atribut in razmejitveno mejo, ki trenutno vozlišče razveji najbolje na dve vozlišči, ki sta njegova otroka. V končnih listih dreves se napovedane vrednosti dobijo ponavadi kar po standardnem postopku, ki velja za navadna regresijska drevesa, in sicer se vzame kar povprečne vrednosti iskane spremenljivke učnih primerov, ki glede na strukturo drevesa pripadejo dotičnemu končnemu listu. Opisani postopek je predstavljen spodaj v algoritmu 1.

Algoritem 1 Naključni gozdovi

1: **for** d = 1 : D **do**
2: Izmed vseh učnih primerov izberemo vzorec z vračanjem. To je trenutna učna množica.
3: Zgradimo regresijsko $drevo_d$: v vsakem vozlišču izberemo podmnožico vseh atributov in glede na le-to izberemo najbolj primeren atribut za delitev.
4: Shranimo si $drevo_d$.
5: **end for**

Ko imamo pridobljena vsa drevesa za testno množico v našem regresijskem primeru, iskano spremenljivko napovemo tako, da za vsak testni primer vzamemo povprečje napovedanih vrednosti, ki jih dobimo iz vsakega izmed dreves.

2.2.3 Gradientni boosting regresijskih dreves

Metodo gradientnega boostinga regresijskih dreves je *Jerome H. Friedman* opisal v člankih [4, 3]. Metoda je pripadnik ansambelskih metod, kjer je končni model sestavljen iz večjega števila osnovnih modelov.

Algoritem se izvaja določeno število iteracij. V vsaki iteraciji se doda novo regresijsko drevo, ki skuša čim bolje popraviti napake prejšnjih že obstoječih dreves. Za metodo je značilno, da skuša optimizirati neko izbrano funkcijo napak $L(y, F(x))$. Sem sodita recimo *kvadratna napaka* $(y - F(x))^2$ in *absolutna napaka* $|y - F(x)|$.

V osnovi si želimo dobiti tak končni model $F^*(x)$, ki glede na učno množico x čim bolj minimizira izbrano funkcijo napak $L(y, F(x))$, torej čim bolj točno napove vrednost y. Želimo se kar najbolj približati funkciji $F^*(x)$ v enačbi 2.9.

$$F^*(x) = arg\min_{F(x)} E_{y,x} L(y, F(x)) \tag{2.9}$$

To dosežemo z združevanjem več osnovnih modelov v en končni model, kot je prikazano v enačbi 2.10. V spodnji enačbi je z M označeno število osnovnih modelov (v našem primeru so to regresijska drevesa), ki sestavljajo naš končni napovedni model. Z drugimi besedami je M tudi število iteracij, skozi katere je šel algoritem, saj le-ta v vsaki iteraciji pridobi nov model. β_m nam predstavlja parameter širjenja oziroma hitrost učenja, torej, kako velik korak bomo naredili v vsaki iteraciji. Kot zadnje pa nam ostane a_m, ki so parametri trenutnega modela. V našem primeru, ko imamo regresijska drevesa, so parametri recimo spremenljivke v vejiščih, na katerih delimo primere, meje razmejitev in končne vrednosti v listih drevesa. Torej kot vidimo, je v našem primeru $h(x; a_m)$ regresijsko drevo, ki ga v vsaki od M iteracij zgradimo.

$$F(x) = \sum_{m=1}^{M} \beta_m h(x; a_m) \tag{2.10}$$

Parameter a_m izračunamo po enačbi 2.11 za različne funkcije napak.

Spremenljivka $\widetilde{y}_{im}$ nam predstavlja najboljšo smer gradientnega spusta in je predstavljen kot nasprotna smer gradienta 2.12. V primeru, ko imamo za funkcijo napake kvadratno napako, je enačba po odvajanju naslednja: $\widetilde{y}_{im} = y_i - F_{m-1}(x_i)$, kjer y predstavlja resnične vrednosti spremenljivke, ki jo napovedujemo. V primeru, ko pa imamo opravka s funkcijo absolutne napake, je enačba sledeča: $\widetilde{y}_{im} = sign(y_i - F_{m-1}(x_i))$, kjer je $sign(x)$ funkcija, ki v primeru, da je $x < 0$, vrne vrednost -1, v primeru, da je $x = 0$, vrne vrednost 0, drugače pa v primeru, da je $x > 0$, vrne vrednost 1.

$$a_m = arg\min_{a,\rho} \sum_{i=1}^{N} [\widetilde{y}_{im} - \rho h(x_i, a)]^2 \tag{2.11}$$

$$\widetilde{y}_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right] \tag{2.12}$$

Parameter β_m pa izračunamo po enačbi 2.13 potem, ko imamo a_m.

$$\beta_m = arg\min_{\beta} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta h(x_i, a_m)) \tag{2.13}$$

Tako se v končnem v skladu z enačbo 2.14 model gradi s prištevanjem glede na model iz prejšnje iteracije. $F_m(x)$ nam v m-ti iteraciji algoritma predstavlja trenutno aproksimacijo želenega modela.

$$F_m(x) = F_{m-1}(x) + \beta_m h(x; a_m) \tag{2.14}$$

Za konec se osredotočimo še nekoliko bolj na konkreten primer uporabe algoritma gradientnega boostinga z regresijskimi drevesi, ki smo jih uporabljali v analizi.

Najprej si poglejmo algoritem na osnovnem primeru, kjer za funkcijo napake uporabljamo kvadratno napako. V tem primeru začnemo tako, da kot osnovno napoved vzamemo povprečno vrednost spremenljivke y. Nato sledi faza gradnje novih regresijskih dreves, kjer v vsaki iteraciji najprej izračunamo $\widetilde{y}_i$, torej v tem primeru, ko imamo opravka s kvadratno napako, razliko med resničnimi vrednostmi spremenljivke y in tisto napovedano vrednostjo, ki se je do trenutne iteracije akumulirala v F, v kateri na začetku

shranimo model povprečnih napovedi. Nato obstoječim modelom v vsaki iteraciji prištejemo nov model, katerega zgradimo v trenutni iteraciji in napoveduje trenutno napako skupnega modela do prejšnje iteracije napram resnični vrednosti napovedovane spremenljivke. Nad to izračunano vrednostjo $\widetilde{y}_i$ v trenutni iteraciji izvedemo učenje trenutnega regresijskega drevesa. Le-tega dodamo v F in nato se izvajanje ponovi v zanki od začetka, kjer ponovno izračunamo po formuli kvadratne napake $\widetilde{y}_i$ glede na do sedaj akumilirani F. Spodaj sledi psevdokoda (algoritem 2) zgoraj opisanega postopka. Z ν je označen parameter, s katerim določamo hitrost učenja.

Algoritem 2 Gradientni boosting regresijskih dreves - kvadratna napaka

1: $F_0(x) = \bar{y}$
2: **for** m = 1 : M **do**
3: $\quad \widetilde{y} = y - F_{m-1}(x)$
4: $\quad$ Zgradimo regresijsko drevo $h(x, a_m)$, ki napoveduje $\widetilde{y}$
5: $\quad F_m(x) = F_{m-1}(x) + \nu \cdot h(x, a_m)$
6: **end for**

Sedaj si poglejmo še algoritem 3, kjer za funkcijo napake uporabimo absolutno napako. Tukaj je algoritem nekoliko drugačen, kot je bil prej. Že na začetku se namreč razlikuje v začetnem napovednem modelu, ki je tukaj mediana in ne več povprečje. V skladu s spremenjeno funkcijo napake je nato tudi nekoliko drugačen izračun $\widetilde{y}$. Najbolj očitna razlika pa se pojavi v 5. vrstici algoritma, kjer regresijsko drevo, zgrajeno na podlagi napovedi vrednosti $\widetilde{y}$, spremenimo tako, da ima vsak list tega drevesa namesto prvotne vrednosti, ki jo v njem drevo napoveduje, raje vrednost mediane rezultatov, katere dobimo, če izračunamo $y - F_{m-1}(x)$ za vse učne primere, ki so glede na prej zgrajeno strukturo drevesa pripadli dotičnemu listu. Tako popravljeno drevo nato uporabimo v 6. vrstici algoritma za nov, popravljen izračun $F_m(x)$.

V končnem, ko je faza učenja končana in imamo vse modele zgrajene, v skladu z enačbo 2.15 samo iteriramo čez modele, ki jih imamo, in njihove napovedi seštevamo. Končna napoved je tako seštevek vseh napovedi, ki nam jih dajo naši osnovni modeli.

Algoritem 3 Gradientni boosting regresijskih dreves - absolutna napaka

1: $F_0(x) = meadiana(y)$
2: **for** m = 1 : M **do**
3: $\widetilde{y} = sign(y - F_{m-1}(x))$
4: Zgradimo regresijsko drevo $h(x, a_m)$, ki napoveduje $\widetilde{y}$
5: Popravimo zgrajeno regresijsko drevo tako, da za vsak list glede na primere, ki pripadajo le-temu, spremenimo vrednost na: $mediana(y - F_{m-1}(x))$
6: $F_m(x) = F_{m-1}(x) + \nu \cdot h(x, a_m)$
7: **end for**

$$y_{napoved} = \sum_{m=0}^{M} F_m(x) \tag{2.15}$$

2.3 Merjenje uspešnosti napovedi

Da lahko ugotovimo, kako uspešni smo bili v naših napovedih, potrebujemo metrike, s katerimi izmerimo uspešnost napovedi. V našem primeru, kjer imamo regresijski problem, potrebujemo mero, ki je primerna za regresijske napovedi.

2.3.1 Koren srednje kvardatne napake

Mera, ki smo jo uporabili za ugotavljanje uspešnosti naših različnih postopkov, je koren srednje kvadratne napake[4] (enačba 2.16), oziroma po angleško *root mean squared error* (RMSE). V enačbi je z y predstavljena resnična vrednost iskane spremenljivke in z $\hat{y}$ napovedana vrednost iskane spremenljivke, ki jo napove naš model. Poleg tega nam ostane še N, ki predstavlja število testnih primerov.

[4] `https://en.wikipedia.org/wiki/Root-mean-square_deviation`

$$RMSE(y, \hat{y}) = \sqrt{\frac{\sum_{p=1}^{N}(y_p - \hat{y}_p)^2}{N}} \tag{2.16}$$

2.3.2 Statistični t-test

To je statistični test, s katerim preverimo, ali se povprečji dveh vzorcev značilno razlikujeta [1]. V našem primeru bi radi preverili, ali smo z našimi postopki značilno presegli uspešnost napovedi, ki nam jih vrača uporaba nekega osnovnega postopka za napoved iskane spremenljivke, kot je recimo napoved povprečne vrednosti učnih primerov.

Ničelna hipoteza je pri tem testu ta, da sta povprečji enaki. V našem primeru si tako želimo zavreči to hipotezo. To lahko storimo, če dobimo p-vrednost dovolj majhno, recimo manjšo od $0,01$, $0,05$ ali pa $0,10$, kar je nekako ustaljena praksa v statistiki. P-vrednost nam predstavlja verjetnost, da naša testirana hipoteza velja. V primeru, da je p-vrednost visoka, bi za nas to pomenilo, da z našim postopkom napovedi nismo dosegli nobene značilne izboljšave. V nasportnem primeru, ko imamo majhno p-vrednost, pa lahko trdimo, da smo z našim postopkom dosegli značilno boljši rezultat od nekega osnovnega postopka.

Za začetek moramo izračunati vrednost statistike t:

$$t = \frac{\text{povprečje razlik posameznih rezultatov}}{\frac{\text{standardna deviacija razlik rezultatov}}{\sqrt{n}}} \tag{2.17}$$

V zgornji enačbi 2.17 je z n pa označeno število elementov v populaciji, torej, za koliko rezultatov primerjamo algoritma.

Nadalje glede na vrednost t in prostostno stopnjo $N - 1$ ugotovimo še potrebno p-vrednost, na podlagi katere lahko v primeru, da je le-ta dovolj nizka, ovržemo ničelno hipotezo. To lahko storimo z uporabo računalnika in katerega od statističnih programov, ki nam za vnešeno vrednost t in prostostno stopnjo zelo natančno izračunajo pripadajočo p-vrednost. Drugače pa lahko p-vrednost ugotovimo tudi sami z uporabo posebne tabele, ki ima že

poračunane pogoste vrednosti. V primeru, da naša vrednost t pade med dve vrednosti, ki sta prisotni v tabeli, to rešimo tako, da vzamemo povprečje.

Poglavje 3

Podatki

Podatke, na katerih smo nato izvajali naše preučevanje števila komentarjev, ki jih sproži posamezna novica, smo pridobili s spletne strani RTV Slovenija[1]. Za potrebe pridobivanja podatkov smo v programskem jeziku Python spisali program, ki je s spleta v določenih časovnih intervalih pridobival HTML vsebino strani. Za omejitev hitrosti izvajanja in pridobivanja vsebin smo se odločili, zaradi preprečevanja preobremenitve strežnika z našimi poizvedbami. Tako smo lahko simulirali obnašanje in hitrost normalnega uporabnika, ki je človek in ne program. Posamezne članke smo nato v našemu računalniku shranili kot tekstovni dokument. Za potrebe naše raziskave smo se nadalje v osnovi osredotočili na novice iz kategorij Slovenija, Svet, EU in Gospodarstvo, saj so bile le-te komentatorsko najbolj zanimive. V končnem smo tako uporabili podatke iz 892 različnih novic, objavljenih v obdobju med 13. februarjem 2015 in 11. majem 2015.

Ko smo pridobili posamezne novičarske članke s spleta, so bile le-te v obliki dokumenta HTML. Takšna oblika je bila neprimerna za našo analizo in uporabo z napovednimi modeli. Zato smo najprej iz vsakega dokumenta HTML, ki je predstavljal posamezno spletno stran oziroma novico, izluščili potrebne podatke. To smo storili z uporabo Python knjižnice *Be-*

[1] `http://rtvslo.si`

autiful Soup[2]. S tem smo se znebili vseh nepotrebnih elementov vsake posamezne spletne strani in značk HTML, ki sestavljajo spletne strani. Tukaj nam je obilico težav povzročal nestandardni format dokumentov HTML, ki so predstavljali posamezne novice. Veliko preglavic nam je recimo povzročal primer, ko je v dokumentu HTML bila prisotna začetna značka naslednjega bloka pred končno značko prejšnjega bloka. Vse takšne nestandardne strukture so povzročale težave uporabljeni knjižnici *Beautiful Soup*, zato smo se v teh primerih morali zateči k naši lastni specializirani kodi, ki je razrešila bolj problematične primere. Vse novice namreč niso imele iste HTML strukture, zato je bilo potrebno podrobno pregledati in upoštevati vse možne strukture, ko smo iz dokumenta HTML z našim programom luščili posamezne elemente, ki so nas zanimali.

Tako smo iz obširnih dokumentov HTML posameznih novic izluščili le pomembne in uporabne podatke, ki smo jih nato uporabili za nadaljnjo predprocesiranje in v končnem uporabo v analizi.

Končni podatki so bili sestavljeni iz dela, pridobljenega iz besedila posameznih novic in meta-podatkov, ki predstavljajo razne dodatne informacije, ki med drugim recimo vključujejo kategorije novic, število avtorjev, uro ter datum objave.

3.1 Tekstovni podatki

Iz vsake novice smo pridobili več različnih vrst tekstovnih podatkov. Prva vrsta tekstovnega podatka je celotna tekstovna vsebina posamezne novice. V to se šteje celotno besedilo novice, vendar brez raznih dodatnih tekstovnih podatkov, ki jih vidimo na spletni strani, nimajo pa neke velike povezave z novico in njeno vsebino. Med slednje sodijo recimo čas in datum objave novice, čas in datum zadnje spremembe novice in podobno. Tako smo kot tekstovne podatke vzeli le strogo tekstovno vsebinsko jedro novičarske strani in ne še vsega ostalega okoliškega teksta, ki se pojavlja na spletni strani.

[2] `https://pypi.python.org/pypi/beautifulsoup4`

Poleg vsebine novic smo za vsako novico izluščili tudi na podlagi značk HTML ves odebeljeni tekst, ki se nahaja v posamezni novici. To smo storili z iskanjem teksta, ki je v dokumentu HTML opremljen z začetno značko $< strong >$ in končno značko $< /strong >$. Tekst, ki se nahaja med omenjenima značkama HTML, se pri ogledu dokumenta spletne strani v spletnem brskalniku bralcu prikaže odebeljen. Tako se ponavadi označuje razne bolj pomembne dele besedila, za katere avtorji menijo, da nosijo večji pomen kot okoliško besedilo in bi morali biti bralci nanj še toliko bolj pozorni.

3.2 Dodatni podatki

Z namenom čim boljšega opisa in poznavanja zbranih novic smo iz novic pridobili tudi razne dodatne podatke, ki niso strogo tekstovne narave, kot tisti opisani zgoraj. V nadaljevanju so tako predstavljeni posamezni dodatni podatki, ki smo jih pridobili iz novic in uporabili za napovedi:

- Osnovni podatek, ki smo ga potrebovali, je bilo *število komentarjev*, ki jih je prejela vsaka posamezna novica. Ta podatek je bil tudi v končnem spremenljivka, ki smo jo z našimi regresijskimi modeli skušali čimbolj natančno napovedati.

- Informacija, v katero *kategorijo* sodi posamezna novica, kjer smo imeli na izbiro, kot je bilo že pri pridobivanju podatkov omenjeno, štiri osnovne kategorije, iz katerih so bile novice, ki smo jih pridobili. To so bile: Slovenija, Svet, EU (Evropska unija) in Gospodarstvo. Vendar pa se je občasno pri nekaterih novicah dogodilo, da so le-te tudi pripadale kateri izmed drugih kategorij, ki jih nismo omenili. V tem primeru smo novici za kategorijo pripisali oznako tipa "ostalo".

- *Število avtorjev* posamezne novice, saj je nekatere novice napisal en sam avtor, nekatere pa je spisala tudi recimo večja skupina avtorjev.

- Povprečne *ocene novic* in *število ocen novic*, ki so jih imele novice, spisane s strani avtorjev trenutne novice, v preteklosti.

– Povprečno število komentarjev v *preteklih podobnih novicah* glede na medsebojno kosinusno podobnost [10], ki smo jo izračunali po enačbi:

$$podobnost = \frac{A \cdot B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} \tag{3.1}$$

Za dve dotični novici nam A in B predstavljata vektorja, ki v našem primeru vsebujeta na istih pozicijah število pojavitev iste besede v posamezni novici. Na ta način smo za vse, ki so bile objavljene dovolj pred trenutno novico, poračunali kosinusno podobnost. Nato smo izbrali določeno število novic, ki so imele največjo podobnost in trenutni novici za iskani podatek pripisali njihovo povprečno število komentarjev.

– Kateri *dan v tednu* in ob kateri *uri*, je bila objavljena novica.

– Pridobili še podatka o *številu slik* in *videov*, ki se pojavijo v posamezni novici.

Poglavje 4

Eksperimenti in rezultati

V tem poglavju bomo predstavili, kako smo v naši analizi pripravili podatke s prej opisanimi metodami predobdelave podatkov in na kakšne načine smo uporabili opisane algoritme za regresijsko napoved števila komentarjev, ki jih dobi posamezna novica.

4.1 Uporabljena predobdelava podatkov

Preizkusili smo več različnih načinov predobdelave podatkov. Vsem besedilom smo najprej odstranili razne moteče elemente, kot so ločila, in vsa besedila pretvorili v sezname posameznih besed, čemur se tudi reče *tokenizacija*. Nato smo različne kratice, ki jih prepoznamo predvsem po tem, da so zapisane z velikimi tiskanimi črkami, v celotnih besedilih zamenjali z enotno besedo, ki je označevala vse kratice. Kratice, kot so: *BBC, AI, STA, MZZ,...* smo tako zamenjali z enotno oznako *ABBREVIATION*. Temu je sledila še zamenjava vseh številk z enotno besedo *NUMBER*, ki označuje številke. Nato je sledila pretvorba vseh preostalih besed v zapise, sestavljene iz malih tiskanih črk. Za konec smo še odstranili zelo pogoste besede, ki nam ne prinesejo nobenega dodatnega znanja pri napovednih modelih, kot so: *bi, bodo, če, ga, ki, saj, se, so,...* Seznam tovrstnih uporabljenih slovenskih besed smo pri-

dobili s spleta[1]. Preostanek besed smo nato lematizirali z uporabo knjižnice LemmaGen[2].

Predobdelane podatke smo nato predstavili z atributnim zapisom. Kot osnovo smo vzeli samo tekstovne podatke, ter te predstavili s številom pojavitev posameznih besed v novici. Naslednja možnost predobdelave samih besedilnih podatkov, ki smo jo pregledali, je štetje pojavitev več zaporednih besed. V našem primeru so bile to tri zaporedne besede. Kot tretji način predstavitve besedila pa je bila metoda štetja pojavitev posameznih odebeljeno zapisanih besed. Nad vsemi tremi opisanimi metodami štetja različnih pojavitev smo v končnem še izvedli postopek TF-IDF.

Opisanim podatkom smo opcijsko dodali še atribute z meta podatki, ki so opisani v poglavju 3.2. Večina teh je v osnovni številski obliki, edino podatek o kategoriji je predstavljen v binarni obliki.

Za dodatni eksperiment smo skupaj združili še vse tri vrste predstavitev tekstovnih podatkov in dodali atribute, dobljene iz dodatnih podatkov.

4.2 Uporabljeni algoritmi

Za začetek smo uporabili regresijska algoritma naključnih gozdov in gradientni boosting regresijskih dreves v osnovni obliki. Poleg tega pa smo oba algoritma uporabili še na nekoliko drugačen način, bolj prilagojen na našo problemsko domeno in podatke, kjer imamo tudi znanje o kategoriji, v kateri je bila novica objavljena. Le-to znanje o pripadajočih kategorijah novic smo izrabili tako, da smo v fazi učenja za vsako kategorijo naučili ločeni regresijski model. Novice iz različnih kategorij imajo namreč lahko dokaj različna števila komentarjev, kar lahko tudi vidimo iz slike 4.1. To smo dosegli tako, da smo iz učnih podatkov za vsako kategorijo zbrali njene pripadajoče novice in na njih izvedli učenje osnovnih prej omenjenih algoritmov. Poleg le-teh pa smo za pridobitev osnovne ravni uspešnosti modelov v naši problemski

[1] `https://github.com/jdf/cue.language`

[2] `https://pypi.python.org/pypi/Lemmagen`

domeni zgradili tudi regresor, ki vsem testnim primerom napove povprečno število komentarjev iz učne množice.

Ustrezne parametre učnih algoritmov smo izbrali s prečnim preverjanjem. Pri algoritmu naključnih gozdov[3] smo nastavili število regresijskih dreves, ki jih zgradi ta ansambelski model. Vsakemu posameznemu drevesu smo tudi omejili globino, do katere se lahko le-to med gradnjo maksimalno razširi. S tem smo preprečili prekomerno prilagajanje učnim podatkom in tako zagotovili potrebno čim večjo splošnost modela. Pri naključnih gozdovih smo v končnem v naših modelih uporabili 500 dreves ter maksimalno globino drevesa nastavili na 25 vozlišč. Pri algoritmu gradientnega boostinga regresijskih dreves[4] smo zopet nastavljali število regresijskih dreves, ki jih zgradi naš ansambelski model. Prav tako smo nastavili maksimalno globino, do katere dopustimo, da se razraste posamezno drevo. Poleg tega smo nastavili še dodaten parameter, in sicer stopnjo učenja, ki predstavlja velikost premika pri optimizacijskem procesu gradientnega spusta. Število regresijskih dreves v modelu in stopnja učenja sta povezana parametra, saj je recimo v primeru majhne stopnje učenja potrebno zgraditi več regresijskih dreves. Glede na [5] je v želji po čim manjši napaki priporočljivo izbrati manjšo vrednost stopnje učenja in nato poiskati primerno število regresijskih dreves, ki jih zgradi model. To pa ima tudi nekoliko manj všečno posledico, saj pomeni gradnja večih regresijskih dreves daljši čas izvajanja programa. V končnem smo za ta algoritem uporabili 1200 dreves ter globino le-teh omejili na 18 vozlišč. Stopnjo učenja pa smo nastavili na 0,006.

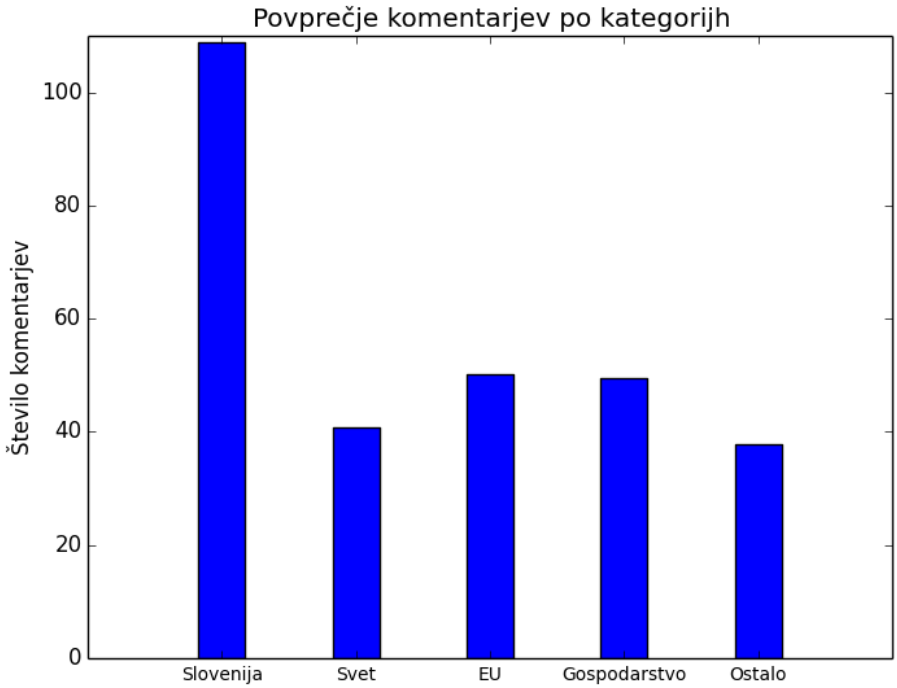
Napovedne točnosti zgornjih modelov smo ocenili na ločeni testni množici.

4.3 Rezultati

Tu so predstavljeni rezultati vrednostenja napovedne točnosti, ki smo jih opravili na naših podatkih z različnimi metodami napovedi. V tabelah z

[3] `http://scikit-learn.org/stable/modules/ensemble.html#forest`

[4] `http://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting`



Slika 4.1: Povprečno število komentarjev novic v izbrani kategoriji.

rezultati uporabljamo oznako *RF* za označevanje naključnih gozdov, oznako *GB* za gradientni boosting regresijskih dreves, *BOW* za štetje pojavitev besed in *n-gram* za štetje pojavitev treh zaporednih besed.

4.3.1 Razdelitev na učno in testno množico

Podatke smo najprej razvrstili kronološko glede na datum objave in za učno množico vzeli prvi del podatkov, za testno množico pa drugi del podatkov. Torej novice, ki so v učni množici, so objavljene pred novicami v testni množici. V učni množici je tako 738 primerov, kar je 82% celotnih podatkov, v testni množici pa 154 primerov novic, torej preostalih 18% podatkov.

Za osnovno uspešnost, katero smo želeli preseči z našimi napovednimi modeli (RF in GBRT), smo si zadali uspešnost napovedi regresorja, ki vsem

testnim primerom napove povprečno število komentarjev iz učne množice. Ocena uspešnosti korena srednje kvadratne napake je v primeru uporabe naše testne množice znašala 113,79.

Rezultate RMSE povzema tabela 4.1. V zgornjem delu tabele so rezultati v primeru, ko smo uporabili le posamezne predstavitve teksta. V drugem delu, kjer je poleg tehnike obdelave še "+", so predstavljeni rezultati, ko smo tekstovnim podatkom dodali še dodatne atribute iz netekstovnih podatkov. V zadnjem delu, to je v primeru *Združeno*, pa so rezultati kombinacije vseh prej uporabljenih podatkov.

Na teh vrstah podatkov smo nato testirali štiri napovedne modele. Prva dva sta osnovni verziji algoritmov naključni gozdovi in gradientnega boostinga regresijskih dreves, druga dva algoritma pa sta naši razširitvi osnovnih algoritmov, tako da uporabljamo za vsako kategorijo svoj regresijski model izbranega algoritma.

Tabela 4.1: Napovedne točnosti (RMSE) različnih regresijskih tehnik (RF in GB) pri različni predobdelavi podatkov s kronološko razdelitvijo na učno in testno množico.

Vrsta predobdelave	Enotni regresor		Za vsako kategorijo posebej	
	RF	GB	RF	GB
BOW	100.33	102.83	102.52	100.44
n-gram	122.47	117.38	123.22	125.64
Odebeljene besede	108.12	107.33	109.08	107.79
BOW+	97.13	98.79	97.04	95.85
n-gram+	107.80	102.84	102.64	98.90
Odebeljene besede+	100.49	92.15	93.24	86.62
Združeno	97.08	97.07	94.31	93.39

Kot je iz tabele 4.1 razvidno, so se bolje od regresorja, ki napoveduje povprečja, odrezali skoraj vsi napovedni modeli. Slabši rezultat smo dobili le v primeru, ko smo uporabljali zgolj štetje *n-gramov* in nobenih dodatnih meta

podatkov. Prav tako je razširjeni postopek, ki gradi regresorje za vsako kategorijo posebej, večinoma dajal boljše rezultate, kot v primeru, ko smo uporabili le osnovno različico posameznega algoritma. Če pa se osredotočimo še na same algoritme, smo večinoma dobili z metodo gradientnega boostinga regresijskih dreves boljše rezultate kot z naključnimi gozdovi. Nekoliko drugačna situacija je v primeru, ko smo uporabili le atribute, dobljene iz teksta in brez dodatnih podatkovnih atributov, saj so tukaj osnovne različice regresijskih algoritmov dobivale nekoliko boljše rezultate, kot tiste, ki so gradile regresorje za vsako kategorijo posebej.

4.3.2 Vrednotenje s prečnim preverjanjem

Za dodatno vrednotenje smo iste vrste podatkov in algoritme kot prej preizkusili z uporabo prečnega preverjanja. Metoda prečnega preverjanja deluje tako, da celotni nabor vseh podatkovnih primerov naključno premešamo in nato razdelimo na neko določeno število čim bolj enako velikih delov. To določeno število delov označimo recimo s K. Nato pa v K iteracijah vsakokrat vzamemo drugo od K-tih podatkovnih skupin in jo označimo za testno množico, vseh ostalih $K - 1$ podatkovnih skupin pa vzamemo za učno množico. Nato se na trenutni učni možici podatkov naučimo naš model in ga testiramo na trenutni testni množici. Končni rezultat je predstavljen kot povprečje uspešnosti korena srednje kvadratne napake na podlagi na vseh K iteracij.

V naši analizi uspešnosti delovanja uporabljenih metod in modelov smo se odločili za vrednost parametra K enako 10. Torej smo naš celoten nabor podatkov razdelili na 10 čim bolj enakih množic in nato vsako iz med njih enkrat vzeli za testno množico, preostale množice pa smo vzeli za učno množico. V našem primeru je bila pri tovrstnem načinu testiranja osnovna uspešnost, ki jo je dosegel regresor, ki je vsem testnim primerom vsakokrat napovedal povprečno vrednost iskane spremenljivke v učni množici okoli 125, merjena kot koren srednje kvadratne napake.

Tudi pri tem načinu testiranja smo dobili rezultate, ki nam večinoma

Tabela 4.2: Napovedne točnosti (RMSE) različnih regresijskih tehnik (RF in GB) pri različni predobdelavi podatkov z uporabo 10-kratnega prečnega preverjanja.

Vrsta predobdelave	Enotni regresor		Za vsako kategorijo posebej	
	RF	GB	RF	GB
BOW	104.60	101.44	106.61	102.92
n-gram	109.21	111.27	109.54	108.76
Odebeljene besede	118.17	119.70	118.55	114.37
BOW+	101.73	99.11	100.77	98.21
n-gram+	99.48	99.93	101.36	100.04
Odebeljene besede+	110.29	107.68	109.78	106.02
Združeno	99.70	98.72	102.10	98.20

kažejo na to, da je naš spremenjeni postopek uporabe algoritmov, ki izkoristi znanje o kategoriji člankov, bolje napovedoval kot pa posamezna osnovna različica algoritma. Poleg tega se je nekoliko spremenilo tudi stanje uspešnosti glede na vrsto uporabljene predobdelave podatkov. V tem primeru smo namreč dobili najboljši rezultat, ko smo uporabili metodo predobdelave, ki je skupaj združila vse vrste predobdelav.

Kot je že bilo omenjeno, smo izvedli prečno preverjanje z razdelitvijo podatkov na 10 čim bolj enako velikih podatkovnih množic. Tako smo tekom testiranja za vsako kombinacijo tehnike predobdelave in uporabljenega regresijskega algoritma dobili 10 rezultatov korena srednje kvadratne napake. Le-te smo nato vsakokrat primerjali z rezultati, ki jih vrača povprečni regresor, to je model, ki napoveduje povprečne vrednosti učnih podatkov. Da za osnovo vzamemo model, ki napoveduje povprečja, smo se odločili zaradi njegove preprostosti in hitrega izvajanja. Ostali modeli so namreč časovno veliko bolj zamudni in bi izračuni vseh kombinacij na našem lokalnem računalniku z omejenimi zmogljivostmi trajal zelo dolgo. Primerjavo med vsakim trenutnim modelom in povprečnim regresorjem smo podprli z izračunom stati-

stičnega t-testa, s katerim smo dokazovali, da sta si uspešnosti obeh modelov res različni. Rezultati posameznih t-testov, ki smo jih dobili, so v obliki p-vrednosti uporabljenega statističnega testa. Prikazani so v tabeli 4.3. Kot je že bilo rečeno pri teoretičnem opisu postopka t-testa, si želimo, da so p-vrednosti nižje od neke meje, ki si jo postavimo. Ponavadi je ta meja recimo 0,01 ali pa 0,05, v skrajnem primeru pa 0,1. Kot je razvidno iz omenjene tabele, so naše p-vrednosti v veliki večini primerov vseskozi celo pod najbolj ostro mejo 0,01. Tako lahko za večino testov trdimo, da smo z našimi metodami značilno presegli uspešnosti osnovnih napovedi s povprečji.

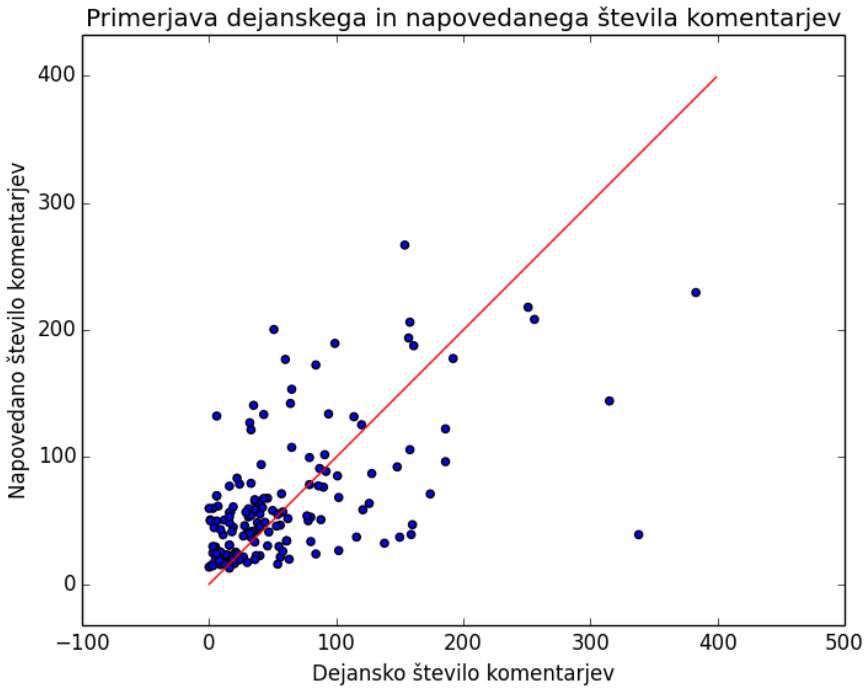
Tabela 4.3: p-vrednosti t-testa napram povprečni orientacijski napovedi.

Vrsta predobdelave	Enotni regresor		Za vsako kategorijo posebej	
	RF	GB	RF	GB
BOW	0.00039	0.00012	0.00620	0.00015
n-gram	0.00270	0.01950	0.00580	0.00050
Odebeljene besede	0.11516	0.01260	0.13514	0.01297
BOW+	0.00010	0.00010	0.00200	0.00015
n-gram+	0.00700	0.00020	0.00110	0.00030
Odebeljene besede+	0.00050	0.00001	0.00500	0.00001
Združeno	0.00027	0.00002	0.00062	0.00011

4.4 Pregled novic z dobrimi in slabimi rezultati

Nadalje smo se poglobili v ugotavljanje, katere novice dobijo dobre napovedi in katere slabe, pri uporabi našega najboljšega regresorja. Napako napovedi smo izračunali kot absolutno razliko med napovedjo števila komentarjev in resničnim številom komentarjev. Na sliki 4.2 so prikazane napovedane vrednosti števila komentarjev posameznih novic glede na njihovo dejansko

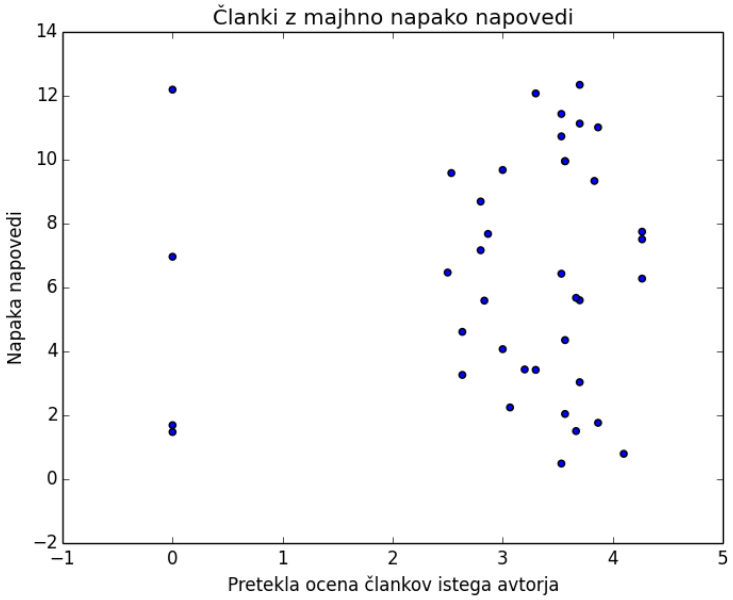
število komentarjev. Idealna napoved se nahaja na premici, ki predstavlja enako vrednost napovedi in dejanskega števila komentarjev. Bolj, ko so točke oddaljene od premice, večjo napako imamo pri naši napovedi. Iz slike lahko tako opazimo, da so za manjše število komentarjev napake manjše, saj so točke veliko manj razpršene in so bolj skoncentrirane blizu idealne diagonalne premice. Z rastjo števila komentarjev pa opazimo, da so točke vedno bolj razpršene in oddaljene od idealne pozicije, torej imamo večje napake napovedi. Za mejo majhne napake smo vzeli prvi kvartil, kar je v našem primeru bila razlika manjša od 12,74543. Za mejo velike napake pa smo vzeli vrednosti, ki so v zadnjem, četrtem kvartilu, v našem primeru je to pomenilo razliko večjo od 61,35747.



Slika 4.2: Napovedano število komentarjev spletnih novic v odvisnosti od dejanskega števila komentarjev, ki ga je dobila posamezna novica.

Če primerjamo napako napovedi v povezavi s preteklo povprečno oceno istega avtorja, lahko vidimo, da je v primeru člankov z boljšimi napovedmi,

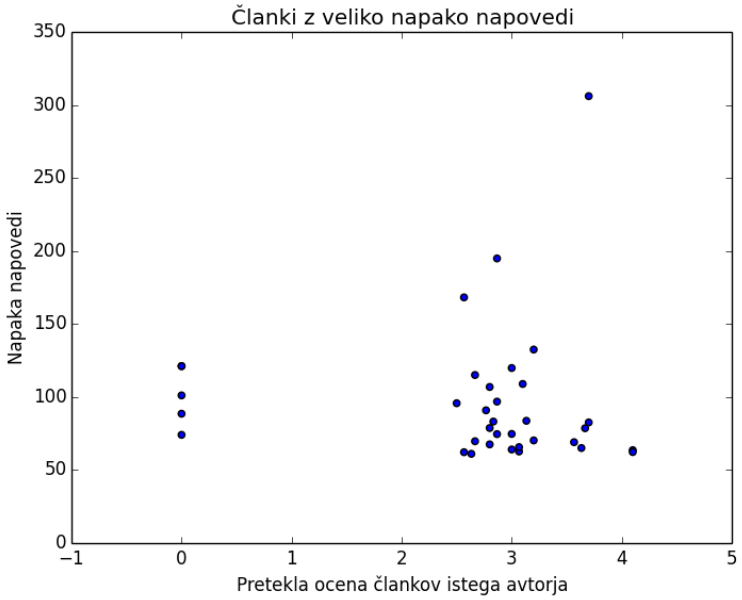
torej manjšimi napakami (na sliki 4.3), glavnina novic takšnih, ki imajo oceno večjo od 3, celo bližje oceni 4. Ko pa se osredotočimo na članke z velikimi napakami (na sliki 4.4), lahko vidimo, da so ocene v tem primeru nižje, in sicer je glavnina novic z ocenami nižjimi od 3. Tako lahko opazimo, da imajo novice, kjer smo imeli dobre napovedi števila komentarjev, tudi večinoma višje povprečne ocene, ki so jih dobili avtorji v preteklih člankih.



Slika 4.3: Majhna napaka napovedi v odvisnosti od pretekle povprečne ocene istega avtorja.

Ko primerjamo napako napovedi s povprečnim številom komentarjev, ki so jih dobili pretekli članki istega avtorja, lahko v sliki 4.5 vidimo, da pri člankih z manjšimi napakami, preteklo število komentarjev večinoma ostane pod mejo 100. Isto pa ne velja za članke z velikimi napakami napovedi, kar je vidno v sliki 4.6. Tukaj se razpon števila komentarjev razteza daleč čez prej omenjeno mejo 100 komentarjev.

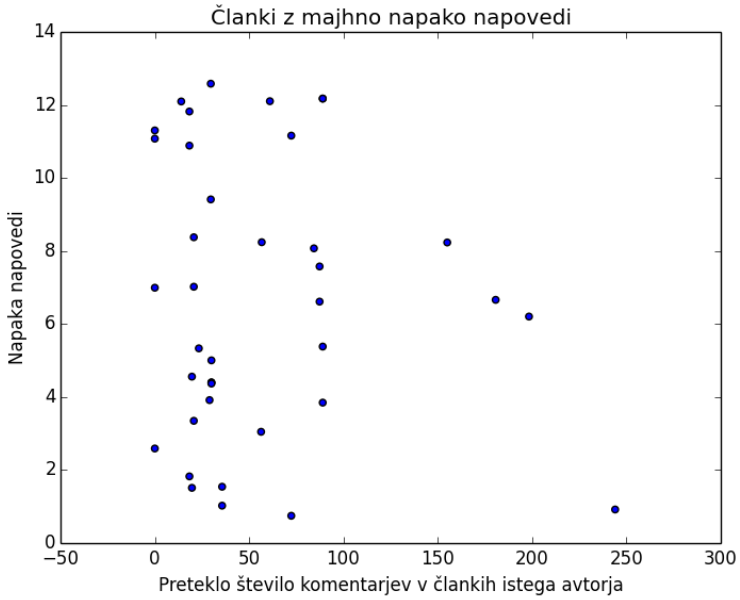
Pri pregledu napake glede na število slik v članku smo ugotovili, da imajo članki, za katere dosežemo dobre napovedi (slika 4.7), večinoma manjše število slik, to je, pod 5 slik. Za članke s slabšimi napovedmi (slika 4.8) pa



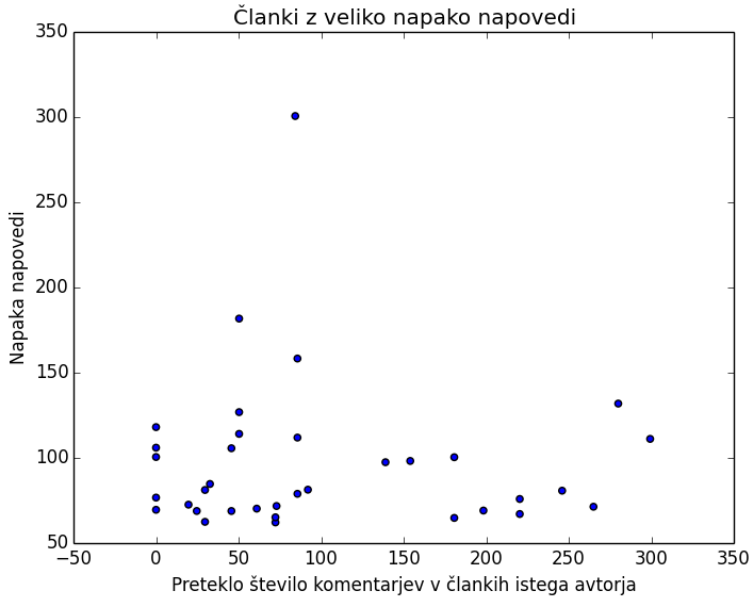
Slika 4.4: Velika napaka napovedi v odvisnosti od pretekle povprečne ocene istega avtorja.

opazimo veliko večji razpon možnega števila slik. Ko gledamo članke z velikimi napakami napovedi, lahko tudi opazimo, da se, razen nekaj osamelcev, napaka napovedi ne viša z večanjem števila slik v članku.

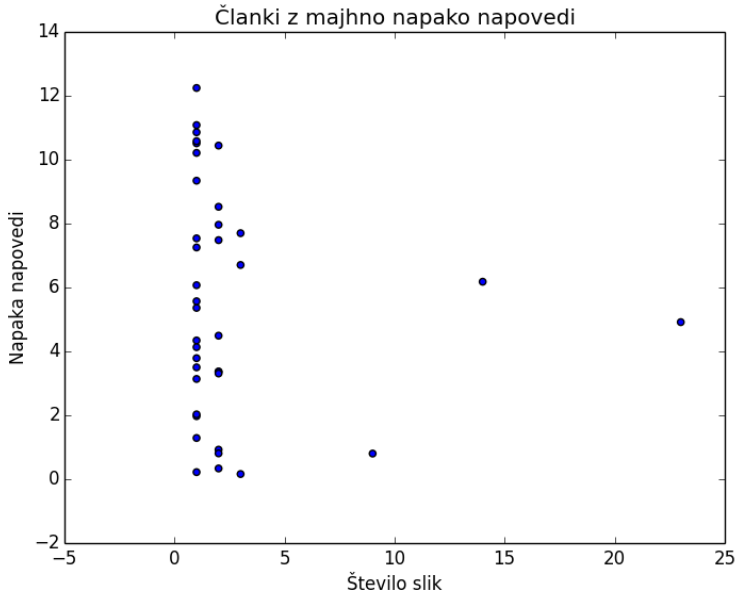
Za konec je še zanimivo pogledati gibanje povprečne napake napovedi glede na čas objave novice. Iz slike 4.9 je razvidno, da imamo največje napake predvsem v jutranjih urah. Poleg tega pa je večinoma tudi razvidno, kasneje v dnevu ko je novica objavljena, manjše napake napovedi imamo.



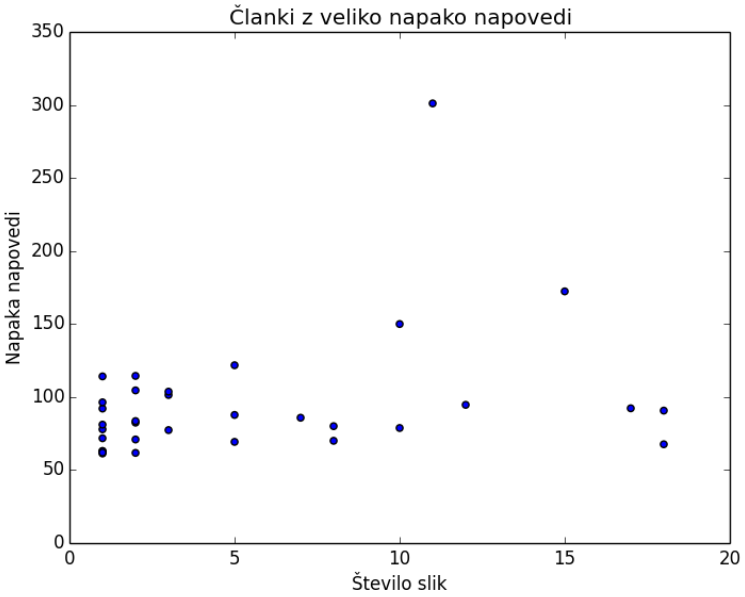
Slika 4.5: Majhna napaka napovedi v odvisnosti od preteklega povprečnega števila komentarjev.



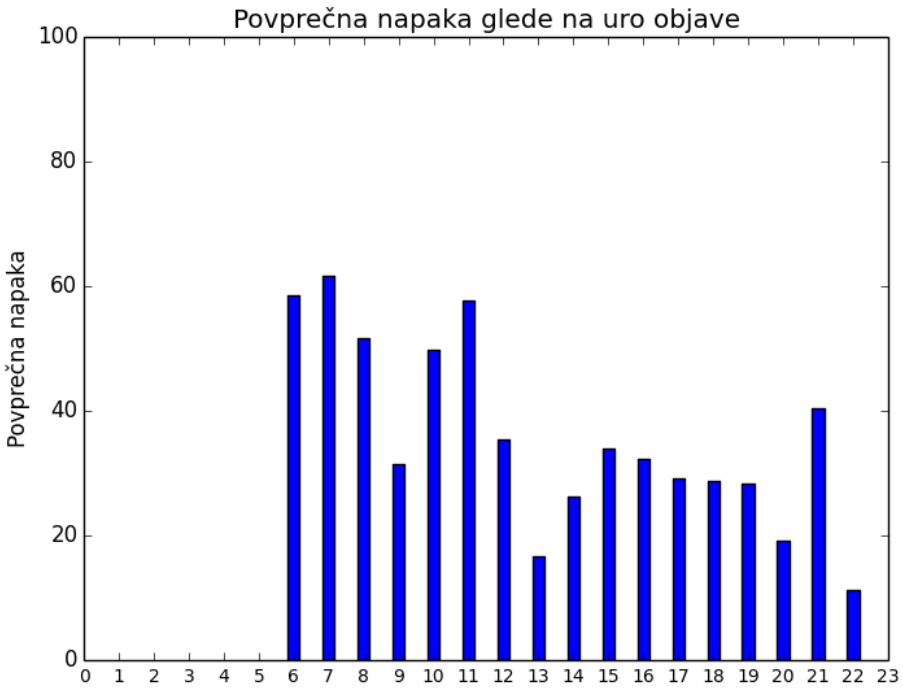
Slika 4.6: Velika napaka napovedi v odvisnosti od preteklega povprečnega števila komentarjev.



Slika 4.7: Majhna napaka napovedi v odvisnosti od števila slik.



Slika 4.8: Velika napaka napovedi v odvisnosti od števila slik.



Slika 4.9: Povprečna napaka napovedi novic, objavljenih ob posamezni uri.

4.5 Pregled vpliva značilk

Pregledali smo, katere značilke je algoritem s tem, ko se je naučil na učni množici, identificiral kot najbolj pomembne za napoved števila komentarjev, ki jih prejme neka novica. V tabelah 4.4 in 4.5 je predstavljenih 20 značilk, in njim pripisanih uteži, ki jih je identificiral algoritem. Za napovedi smo uporabili algoritme, ki spadajo v skupino ansambelskih metod, natančneje - uporabljajo množico regresijskih dreves. V primeru posameznih regresijskih dreves se pri gradnji le-teh v vsakem notranjem vozlišču izračuna pomembnost značilk[5]. Za razmejitveni kriterij v trenutnem vozlišču se izbere značilko z najvišjo vrednostjo pomembnosti. Ko želimo ugotoviti pomen značilk glede na končno celotno regresijsko drevo, pregledamo vsa vozlišča in pomembnosti izbranih razmejitvenih značilk v le-teh. Ob tem izračunamo skupno vsoto pomembnosti vsake značilke glede na naše drevo [5]. Ponavadi te vsote še skaliramo na nek želen interval, ali pa dosežemo vsoto vpliva vseh značilk enako 1. Zaradi več modelov, ki šele skupaj tvorijo celoto in s tem naš končni model, se v primeru ansambelskih metod ne moremo poslužiti zgolj osnovne metode za ugotavljanje pomembnosti posameznih značilk, ki jo uporabimo za regresijska drevesa. Kot je opisano v dokumentaciji knjižnice *scikit-learn*[5], ki smo jo uporabljali za gradnjo modelov, se v tem primeru pomembnost spremenljivk ugotovi s povprečenjem pomembnosti značilk iz vseh modelov, ki so v ansamblu. V našem primeru so bile značilke pridobljene iz razširjenega postopka gradientnega boostinga regresijskih dreves, ki je uporabljal znanje o kategorijah člankov in za vsako kategorijo gradil nov napovedni model glede na učno množico. Nabor podatkovnih atributov je bil v tem primeru enak tistemu iz tabele 4.1, kjer smo vzeli odebeljene besede in dodali razne uporabne meta podatke, ki niso neposredno izvirali iz teksta. Tako vsak stolpec tabele predstavlja najpomembnejše značilke za vsak model oziroma kategorijo posebej.

Zaradi pomanjkanja prostora smo v tabelah uporabili naslednje kratice:

[5] `http://scikit-learn.org/stable/modules/ensemble.html`

ŠOPČ Število ocen podobnega članka

ŠOIA Število ocen istega avtorja

OIA Ocena istega avtorja

ŠKIA Število komentarjev istega avtorja

ŠAČ Število avtorjev članka

Kot je razvidno iz tabel 4.4 in 4.5, so vsi glede na kategorijo ločeni modeli največjo težo pripisali nekaterim atributom, pridobljenim iz dodatnih podatkov, ki niso v osnovi pridobljeni iz samega teksta novic.

Če pa se nekoliko bolj osredotočimo na same strogo tekstovne atribute, lahko vidimo, da je model, ki je namenjen napovedi števila novic v kategoriji *Slovenija*, dal veliko težo atributom, ki imajo pretežno povezavo z raznimi vsakodnevnimi splošnimi novicami o dogajanju v državi Sloveniji in novicami o političnem dogajanju.

Model, ki je napovedoval za kategorijo *Svet*, če se zopet osredotočimo le na tekstovne atribute, je kot pomembne identificiral tiste, ki se nanašajo na svetovno pomembne države in politike.

Tretji model v tabeli je napovedoval za kategorijo *EU*, oziroma Evropske unije. Kot pomembne tekstovne atribute je označil v novicah pojavljajoče besede, ki se nanašajo na razne, za celotno Evropsko unijo pomembne tematike, v času zajema podatkov. To je recimo tematika beguncev, ki prihajajo na ozemlje Evropske unije. Poleg tega je pomembna tematika Grčija in verjetno njihovo trenutno pogajanje z Evropo glede njihovega dolga. Prav tako je model tudi velik pomen pripisal atributom, ki se nanašajo na razne novice v povezavi s tematiko gospodarstva, denarja, zaposlitve in mladimi. Mladi so verjetno velikokrat omenjeni ravno v povezavi z njihovimi zaposlitvami, saj se je kot posledica krize, v mnogih državah otežila zaposlitev mladih.

V tabeli je četrti model namenjen napovedim za kategorijo *Gospodarstvo*. Tukaj so, če se zopet osredotočimo na strogo tekstovne atribute, največjo težo

dobile besede, ki se večinoma nanašajo na gospodarstvo oziroma besede, ki so povezane z denarjem.

Kot zadnji pa so predstavljeni še pomembni atributi, ki jih je identificiral model za napovedi na vseh kategorijah, ki ne spadajo pod prej opisane štiri, in je tako označen z *Ostalo*. Tukaj je zaradi mešane narave novic težje razločiti neko točno določeno tematiko, na katero bi večinoma kazale besede, ki imajo največjo težo pri napovedi števila komentarjev.

Tabela 4.4: Prvi del seznama značilk z največjo težo za napovedi glede na ločeni model za kategorije. Podana je tudi informativnost atributa, kot jo izračuna model gradientnega boostinga regresijskih dreves.

Slovenija		Svet		EU	
ŠOPČ	0.05895	ŠOPČ	0.18353	ŠOPČ	0.16608
OIA	0.03853	OIA	0.05549	ŠOIA	0.07174
ŠKIA	0.03635	ŠOIA	0.05014	OIA	0.06096
večina	0.03530	# slik	0.04035	ŠKIA	0.05082
# slik	0.03367	ŠKIA	0.03722	begunec	0.03420
ŠOIA	0.03140	let	0.03181	aleksis	0.03330
cerar	0.03121	Številka	0.02983	# slik	0.02931
# videov	0.02241	vladimir	0.02926	EU	0.02788
Številka	0.02026	ati	0.02618	lani	0.02460
janša	0.01836	# videov	0.02000	objava dopoldne	0.02281
črn	0.01815	von	0.01923	mlad	0.02197
prebrati	0.01656	imeti	0.01638	zaposlen	0.02177
miro	0.01445	prebrati	0.01344	objava v torek	0.01976
referendum	0.01289	mesto	0.01326	čas	0.01737
ŠAČ	0.01196	izrael	0.01064	objava v petek	0.01591
imeti	0.01041	objava v petek	0.01061	grčija	0.01582
objava zvečer	0.00971	še	0.01058	denar	0.01269
veber	0.00843	ZDA	0.01030	# videov	0.01189
dušan	0.00808	martin	0.00961	slovenija	0.01143
velik	0.00793	predsednik	0.00944	objava zvečer	0.01109

Tabela 4.5: Drugi del seznama značilk z največjo težo za napovedi glede na ločeni model za kategorije. Zopet je tudi podana informativnost atributa, kot jo izračuna model gradientnega boostinga regresijskih dreves.

Gospodarstvo		Ostalo	
OIA	0.10739	OIA	0.12389
ŠOPČ	0.10628	# slik	0.10995
ŠKIA	0.07113	ŠOPČ	0.06824
ŠOIA	0.06965	objava zvečer	0.05755
objava v torek	0.04913	objava popoldne	0.05040
# videov	0.03616	objava v sredo	0.04931
objava popoldne	0.02694	ampak	0.03975
# slik	0.02566	ŠKIA	0.02868
Številka	0.02444	maribor	0.02513
plača	0.02358	ŠOIA	0.02283
prodaja	0.02259	točka	0.02027
objava v soboto	0.02160	večina	0.01925
nov	0.02081	objava v nedeljo	0.01891
velik	0.02016	Številka	0.01822
ever	0.01738	center	0.01774
ŠAČ	0.01554	andrej	0.01701
objava zvečer	0.01379	vojen	0.01681
rast	0.01331	mlad	0.01603
padec	0.01141	studio	0.01563
rešitev	0.00948	dejan	0.01489

Poleg prej omenjenega razširjenega algoritma smo pregledali pomembnosti atributov tudi za osnovni algoritem, in sicer smo tokrat na podatkih uporabili osnovno verzijo algoritma gradientnega boostinga regresijskih dreves, ki ni gradila novega modela z vsako kategorijo novic, temveč je bil uporabljen en enotni model nad vsemi kategorijami. Najbolj pomembnih 20 atributov

je predstavljenih v tabeli 4.6.

Tukaj so zopet med bolj pomembnimi atributi tisti, ki izvirajo iz dodatnih podatkov, ki niso strogo pridobljeni na podlagi tekstovne vsebine novic. Če se osredotočimo na tekstovne atribute, vidimo, da so večjo težo dobili atributi, ki se predvsem, še posebej, če jih primerjamo z razporeditvijo po kategorijah iz tabel 4.4 in 4.5, nanašajo večinoma na lokalno slovensko dogajanje v novicah, poleg tega pa so še tudi elementi tematik iz Evropske unije in gospodarstva.

Sedaj pa si poglejmo še pomembnost značilk po posameznih kategorijah novic, ki smo jih dobili v primeru, da nismo uporabljali dodatnih podatkov in s tem atributov, torej le atribute, pridobljene iz besedila novic. V tem primeru se je najbolje odrezala podatkovna predstavitev štetja posameznih besed, oziroma *BOW*, kakor je označen v tabelah z rezultati natančnosti napovedi. Kot algoritem pa se je kot najboljši izkazal razširjeni gradientni boosting regresijskih dreves, kjer smo za vsako kategorijo posebej zgradili svoj model. Najbolj pomembne besede, identificirane s strani modelov za vsako posamezno kategorijo, so predstavljene v tabeli 4.7.

V tabeli vidimo, da je model za novice iz kategorije *Slovenija* kot najbolj pomembne atribute označil predvsem besede, katere se uporabljajo v povezavi z raznimi novicami, ki se nanašajo na vsakodnevno dogajanje v državi, recimo na politiko.

Model za napoved na kategoriji *Svet* je kot pomembne izpostavil besede, ki nisto tako jasne kot v prejšnjem primeru, ko smo uporabili tudi dodatne podatke. So pa besede seveda takšne, ki kažejo na razne novice, ki poročajo o svetovnem dogajanju, še posebej o svetovnemu političnemu dogajanju.

Model za kategorijo *EU* kot pomembne označi besede, ki se predvsem nanašajo na valuto Evro in na splošno Evropsko unijo. Poleg tega so tudi izpostavljene besede, ki so v povezavi z gospodarstvom.

Model za kategorijo *Gospodarstvo* predvsem izpostavi besede, ki se nanašajo na ekonomijo in na splošno na tematike v povezavi z denarjem. Prav tako je tudi edini od petih modelov, ki je izpostavil številke v besedilu kot po-

Tabela 4.6: Seznam značilk z največjo težo za napovedi glede na enotni model. Podana je tudi informativnost atributa, kot jo izračuna model gradientnega boostinga regresijskih dreves.

Značilke	Teža značilke
ŠOPČ	0.05319
ŠOIA	0.03352
ŠKIA	0.02933
OIA	0.02924
cerar	0.02719
ura	0.02512
donalda	0.02389
# videov	0.02128
večina	0.02098
pomemben	0.01969
reforma	0.01868
jož a	0.01801
# slik	0.01709
veber	0.01528
mir	0.01392
begunec	0.01321
sprejeti	0.01277
prodaja	0.01130
Številka	0.01108
izrael	0.01076

membne. To je v skladu s tem, da so pomembne besede, ki se nanašajo na tematike o denarju, saj so ponavadi poleg takšnih besed tudi denarni zneski v številčni predstavitvi.

Zadnji je model za *ostale* kategorije, kjer je stanje zopet podobno kot prej, saj je zaradi bolj mešanih novic težje izluščiti neko določeno tematiko,

ki povezuje izpostavljene besede.

Tabela 4.7: Seznam značilk z največjo težo za napovedi glede na ločeni model za kategorije.

Slovenija	Svet	EU	Gospodarstvo	Ostalo
referendum	zgoditi	ever	povečati	pričakovati
skupen	država	banka	plača	kljub
ta	policija	evropski	precej	področje
zavod	april	glede	trg	partner
let	njegov	gospodarstvo	Številka	financiranje
imeti	vlada	okvir	ever	zdeti
ves	večina	družina	cena	ustava
beseda	sistem	sredstvo	ponudba	svetoven
stranka	še	kazati	leten	proračun
komisija	mora	zahteva	mora	reden
želeti	pot	prihodnji	predlog	vir
dejati	del	lani	dober	imeti
janša	javen	mnenje	ljubljanski	leten
stališče	zaradi	področje	svet	mlad
znova	kitajski	poročati	sindikat	stanje
slovenski	sporočiti	zunanji	navedba	Kratica
podatek	znova	podjetje	določen	ljubljana
država	dogovor	let	mena	določen
zaradi	reforma	svoj	finančen	še
videti	življenje	mlad	velik	drug

Za konec so v tabeli 4.8 predstavljene še identificirane najbolj pomembne besede za napoved s strani najboljšega osnovnega algoritma, tokrat naključnih gozdov, ki ni gradil posebnega modela za vsako kategorijo posebej. Tukaj zopet nekoliko prednjačijo besede, ki se navezujejo na tematiko lokalnih slovenskih novic.

Tabela 4.8: Seznam značilk z največjo težo za napovedi glede na enotni model, torej ta, ki ne upošteva kategorije novice.

Značilke
referendum
pravica
janša
zakonski
veber
ustaven
odstop
zdeti
ta
zveza
sprejeti
odstotek
fakulteta
sprejet
let
cerar
vojen
izmed
družina
podpis

Poglavje 5

Sklepne ugotovitve

V diplomski nalogi smo za besedila v slovenskem jeziku pregledali področje regresijske napovedi. Po pridobivanju podatkov in primerni predobdelavi teksta smo za regresijsko napoved uporabili štiri postopke, in sicer osnovni različici algoritmov naključnih gozdov in gradientnega boostinga regresijskih dreves ter razširjeni različici omenjenih algoritmov, ki sta uporabljali informacijo o kategoriji novice. Nato smo postopke napovedi testirali z razdelitvijo na učno in testno množico ter z desetkratnim prečnim preverjanjem. Pri slednjem smo s statističnim t-testom preverili, ali so naše napovedi značilno boljše od osnovne napovedi. Za konec pa smo še pregledali, katere značilke nosijo največjo težo za napoved števila komentarjev ter za kakšne tipe novic uspemo napovedati število komentarjev z majhno napako in za kakšne z veliko.

Zastavljeni cilji so bili izpolnjeni, saj smo uspeli z našimi postopki napovedati število komentarjev novic. Napovedne točnosti so še posebej v primeru naših najboljših razširjenih modelov, ki uporabljajo informacijo o kategoriji novice, občutno presegle uspešnost osnovnega napovednega modela. Prav tako smo tudi uspešno analizirali ozadje napovedi in predstavili razne lastnosti novic, ki igrajo vlogo pri uspešni napovedi količine komentarjev neke novice.

Za konec pa posvetimo nekaj pozornosti možnim izboljšavam našega sis-

tema napovedi. Prvo priložnost vidimo v še boljšem sistemu predobdelave besedil in lematizacije le-teh, saj je ta postopek občasno za nekatere besede vračal nekoliko neobičajne in čudne lematizirane oblike besed. Prav tako bi celotno raziskavo lahko še dodatno razširili, tako da bi pridobili in upoštevali poleg samega besedila novic tudi tekst posameznih komentarjev. Tako bi lahko še dodatno v večjo podrobnost proučili, kako neka novica vpliva na komentatorske navade spletnih uporabnikov. To je, kakšne vrste komentarjev spodbudi neka novica, se bodo bralci na novico odzvali negativno ali pozitivno. Poleg tega pa bi lahko nato še preverjali, kakšna medsebojna komunikacija se razvije med komentatorji, ki komentirajo neko novico. Bo novica sprožila v končnem val prerekanja med komentatorji, ali pa se bo razvila neka konstruktivna debata o vsebini dotične novice.

Literatura

[1] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[2] Misha Denil, David Matheson, and Nando de Freitas. Narrowing the gap: Random forests in theory and in practice. In *International Conference on Machine Learning (ICML)*, pages 665–673, 2014.

[3] Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38:367–378, 1999.

[4] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[5] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction.* Springer, second edition, 2009.

[6] Kiyoshi Izumi, Takashi Goto, and Tohgoroh Matsui. Trading tests of long-term market forecast by text mining. In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, ICDMW '10, pages 935–942, Washington, DC, USA, 2010. IEEE Computer Society.

[7] Matjaz Juršic, Igor Mozetic, Tomaz Erjavec, and Nada Lavrač. Lemmagen: Multilingual lemmatisation with induced ripple-down rules. *Journal of Universal Computer Science*, 16(9):1190–1214, 2010.

[8] Shimon Kogan, Dimitry Levin, Bryan R. Routledge, Jacob S. Sagi, and Noah A. Smith. Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, (AACL '09), pages 272–280, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[9] Igor Kononenko and Marko R. Šikonja. *Inteligentni sistemi*. Založba FE in FRI, 2010.

[10] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[11] Dong Nguyen, Noah A. Smith, and Carolyn P. Rosé. Author age prediction from text using linear regression. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, (LaTeCH '11), pages 115–123, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[12] Manos Tsagkias, Wouter Weerkamp, and Maarten de Rijke. Predicting the volume of comments on online news stories. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1765–1768, New York, NY, USA, 2009. ACM.

[13] Gitte Vanwinckelen and Wannes Meert. Predicting the popularity of online articles with random forests. In *ECML/PKDD Workshop on Predictive Web Analytics, Nancy, France, 19 September 2014*, pages 1–6, 2014.

[14] Felix Wex, Natascha Widder, Michael Liebmann, and Dirk Neumann. Early warning of impending oil crises using the predictive power of online news stories. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 1512–1521, 2013.

[15] Tae Yano and Noah A. Smith. What's worthy of comment? content and comment volume in political blogs. In *Proceedings of the Fourth International Conference on Weblogs and Social Media (ICWSM)*, 2010.