Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Klemen Kadunc

Določanje sentimenta slovenskim spletnim komentarjem s pomočjo strojnega učenja

diplomsko delo
na visokošolskem strokovnem študiju

Mentor: izr. prof. dr. Marko Robnik-Šikonja

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Analiza sentimenta besedil se ukvarja z odkrivanjem piščevega mnenja o predmetu pisanja. Večinoma se uporablja polarna analiza, torej določamo pozitivno, negativno in nevtralno mnenje. Naloga za avtomatske sisteme ni enostavna, saj je potrebno iz besedila izluščiti bistveno semantično informacijo, pri tem pa težavo predstavljajo (večkratno) zanikanje, sarkazem, dvoumnost besedila, kontekstna odvisnost rabe besed itd. Pravilno določanje sentimenta je koristno za številne namene, npr. za napovedovanje uspešnosti izdelkov, izidov volitev ali v sociooloških raziskavah.

Pripravite prototip sistema za sentimentno analizo, ki vsebuje leksikon sentimentnih besed za slovenski jezik, in ga v kombinaciji s strojnim učenjem ovrednotite na zbirki besedil, ki vsebuje spletne komentarje. Vašo rešitev kritično analizirajte.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Klemen Kadunc sem avtor diplomskega dela z naslovom:

Določanje sentimenta slovenskim spletnim komentarjem s pomočjo strojnega učenja (angl. *Using machine learning for sentiment analysis of Slovene web commentaries*)

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Marka Robnika-Šikonje,
- 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 8. marec 2016 Podpis avtorja:

Zahvaljujem se svojemu mentorju, ki mi je veliko pomagal z napotki glede problemske domene ter kako se spoprijeti z večjimi in manjšimi težavami, ki so se pojavile tekom izdelave dela. Hvala tudi za veliko mero potrpežljivosti, da ekspresno hitrih odgovorov na vprašanja niti ne omenjam.

Posebna zahvala gre moji ožji in širši družini, ki mi je tekom let stala ob strani in mi omogočila brezksrbno otroštvo ter prijetna mladostniška leta. Vse to so stvari, ki so neizmerljive.

Posebej moram omeniti še Katarino in Barbaro, ki sta mi pomagali pri označevanju besedila. Hvala vama, z vajino pomočjo je korpus označenih komentarjev zagotovo kvalitetnejši.

Moji mami Bernardi.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTML	hypertext markup language	jezik za označevanje spletnega besedila
AJAX	asynchronous JavaScript and XML	asinhroni JavaScript in XML
XML	extensible markup language	razširljiv označevalni jezik
XSD	XML schema definition	definicija sheme XML
DBMS	database management system	sistem za upravljanje podatkovnih baz
SQL	structured query language	strukturiran povpraševalni jezik
IDE	integrated development environment	integrirano razvojno okolje
API	application programming interface	programski vmesnik
UGC	user generated content	uporabniško generirane vsebine
NLTK	Natural Language Toolkit	zbir knjižnic za obdelavo naravnega jezika
CA	classification accuracy	klasifikacijska točnost
SVM	support vector machine	metoda podpornih vektorjev
NB	naive Bayes	naivni Bayes
TF	term frequency	pogostost izraza
TF-IDF	term frequency - inverse document frequency	pogostost izraza - obratna pogostost dokumentov

Povzetek

Cilj diplomske naloge je izdelava orodja za sentimentno analizo besedila, konkretneje uporabniških komentarjev. Preizkusili smo več metod strojnega učenja in več metod za predobdelavo besedil, še posebej tistih za spletna besedila. Kot najboljši klasifikator se je izkazal multinomski naivni Bayes. Za izboljšanje klasifikatorja smo pripravili slovenski slovar sentimenta - seznam besed in besednih zvez s pozitivno in negativno konotacijo. Za osnovo smo vzeli angleški slovar sentimentnih besed ter ga ročno prevedli v slovenščino. Analizo sentimenta smo izvajali na ročno označenem korpusu uporabniških komentarjev, ki smo jih izluščili iz nekaterih najbolj obiskanih slovenskih novičarskih portalov. Slovar ter označen korpus uporabniških komentarjev sta naša glavna prispevka k analizi sentimenta za slovenski jezik.

Ključne besede: analiza sentimenta, strojno učenje, rudarjenje mnenj, obdelava naravnega jezika, klasifikacija, označevanje besedil, slovar sentimenta, slovenski jezik, predobdelava besedila, uporabniško generirane vsebine.

Abstract

Title: Using machine learning for sentiment analysis of Slovene web commentaries

The purpose of this work is to develop a tool for sentiment analysis of user comments. Several machine learning classifiers were tested and multinomial naive Bayes turned out to be the best predictor. We tried several preprocessing techniques, especially those for web texts. The classifier was improved with a Slovene sentiment lexicon, which is a list of words and set phrases with a positive and a negative connotation. An English sentiment lexicon was manually translated into Slovene. The analysed corpus of user comments was manually annotated by three annotators; its entries were selected from some of the most visited Slovene news portals. Both the lexicon and the annotated corpus of user comments are the main contributions of this work.

Keywords: sentiment analysis, machine learning, opinion mining, natural language processing, classification, annotating text, opinion lexicon, Slovenian language, text preprocessing, user generated content.

Poglavje 1

Uvod

V diplomskem delu raziskujemo problematiko analize sentimenta uporabniških komentarjev v slovenskem jeziku. Analiza sentimenta oziroma razpoloženja je v zadnjem času še posebej priljubljena zaradi velikega potenciala za raziskovanje javnega mnenja. Z analizo sentimenta zaznavamo mnenje in ga skušamo opredeliti kot dobro (mnenje s pozitivno konotacijo) ali slabo (mnenje z negativno konotacijo) [1]. Tema nas je pritegnila zaradi aktualnosti in dejstva, da tovrstnih raziskav za slovenski jezik primanjkuje.

V tem poglavju najprej opredelimo pojem analize sentimenta, predstavimo nekaj osnovnih pristopov k reševanju problematike ter navedemo razloge za pomembnost analize uporabniško generiranih vsebin (denimo komentarjev uporabnikov) v današnjem času. Po opredelitvi pojmov sledi postavitev ciljev ter pregled obstoječih poizkusov analize sentimenta za slovenski jezik. Poglavje zaključimo s pregledom organizacije diplomske naloge.

1.1 Splošno o analizi sentimenta

Ljudje radi izražajo mnenje o različnih stvareh, prav tako radi prisluhnejo mnenju drugih, še posebej takrat, ko je v igri odločitev in s tem povezana dilema [2], pa naj si bo to odločitev o nakupu novega osebnega vozila ali zgolj izbira celovečerca v lokalni kinodvorani v soboto zvečer. Analizo sentimenta

lahko opredelimo kot raziskovanje posameznikovega mnenja, razpoloženja, emocij o nekem dogodku, produktu, organizaciji, temi ali osebi [3]. Cilj analize sentimenta je izdelava avtomatiziranega sistema za identifikacijo in opredelitev mnenja v besedilu. Z razvojem interneta in dostopa do velikanske količine besedil, ki izražajo mnenje, je analiza sentimenta postala zelo privlačno raziskovalno področje v domeni obdelave naravnega jezika (NLP). Besedilne informacije v grobem klasificiramo kot objektivne oziroma dejstva in subjektivna mnenja [2]. Za razliko od analize sentimenta se ostala NLP opravila, kot je denimo poizvedovanje po informacijah (angl. *information retrieval*), fokusirajo na objektivne informacije. V literaturi [1, 3] se poleg pojma analiza sentimenta uporabljajo še sinonimi, kot so analiza mnenj, klasifikacija razpoloženja ter rudarjenje mnenj (angl. *opinion mining*). Čeprav gre med njimi za (manjša) odstopanja in jih ne gre povsem enačiti, se raziskave osredotočajo pretežno na isto vsebino.

Bing Liu [4] je analizo sentimenta definiral kot *"Za dano množico besedilnih dokumentov D, ki vsebujejo mnenja (ali sentimente) o objektu, analiza sentimenta stremi k izluščevanju atributov in komponent komentiranega objekta v vsakem dokumentu $d \in D$ in določanju ali so komentarji pozitivni, negativni ali nevtralni."*. Oglejmo si še opredelitev nekaterih pojmov, ki se pri tem pojavijo. Objekt O je definiral kot *"entiteto, ki je lahko produkt, tema, oseba, dogodek ali organizacija"* na katero se mnenje nanaša, medtem ko je imetnik mnenja (angl. *opinion holder*) *"oseba ali organizacija, ki ima mnenje"*. V primeru uporabniških komentarjev je imetnik mnenja običajno avtor komentarja. Semantična usmerjenost mnenja označuje ali ima imetnik mnenja o objektu (ali njegovih atributih), na katerega se mnenje nanaša, pozitivno ali negativno občutenje. Za semantično usmerjenost se uporabljajo različne oznake. Poleg oznake pozitivno in negativno se v primeru, ko besedilo ne vsebuje subjektivnih elementov, lahko doda še oznaka nevtralno. Primeri besedil za vse tri oznake so v tabeli 1.1. Semantična usmerjenost je lahko predstavljena tudi z drugimi oznakami, denimo številskimi vrednostmi (5 - zelo pozitivno, 4 - pozitivno,..., -1 skrajno negativno) ipd [1].

sem. usmerjenost	besedilo
pozitivno	Hudooo. Ves večer smo kričali od navdušenja.
	Naši so jih res nadigrali.
nevtralno	Neodločen rezultat Slovenije in
	Makedonije na prvi tekmi.
negativno	Ne vem, zakaj sploh hodijo na tekme.
	Drugič naj bodo raje doma.

Tabela 1.1: Primeri semantične usmerjenosti besedila.

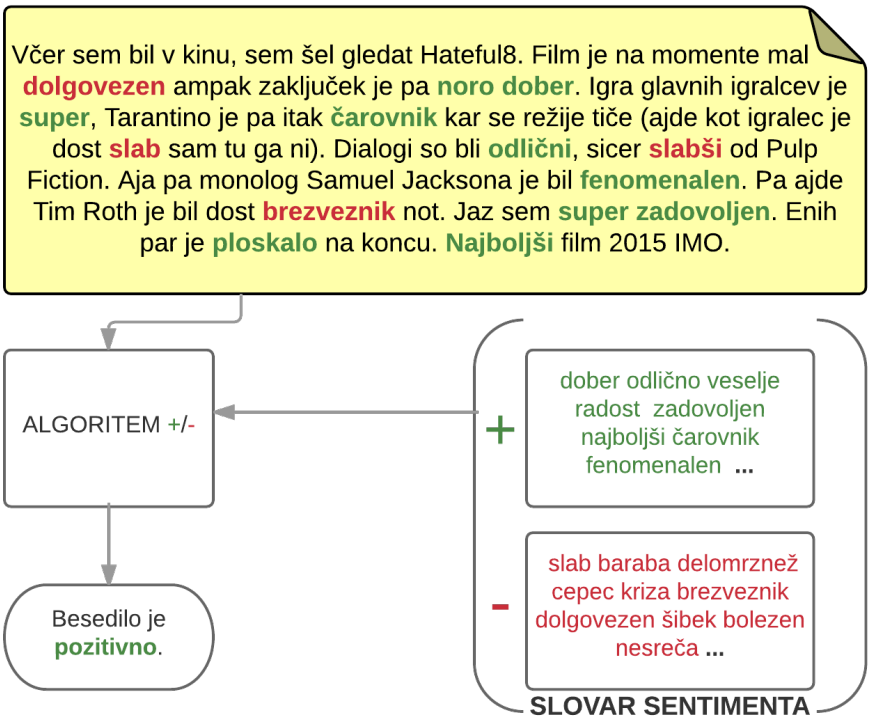
Analizo sentimenta lahko po Bingu Liu [3] razdelamo in ovrednotimo na različnih nivojih. Na **nivoju dokumenta** (angl. *document level*) se za dokument kot celoto (denimo komentar uporabnika na novico) ugotovi ali izraža pozitiven ali negativen sentiment. V našem delu se osredotočimo na ta nivo. Na **nivoju stavka** (angl. *sentence level*) se poizkuša ugotoviti sentiment posameznih stavkov. V kolikor stavek ne izraža mnenja, se ga označi kot objektiven (nevtralen). Na **nivoju entitete in vidika** (angl. *entity and aspect level*) gre za najpodrobnejšo razčlenitev. Težava pri prejšnjih dveh nivojih je v tem, da ni točno določeno, kaj je bilo nekomu všeč ali česa nekdo ne mara. Na tem nivoju gre za fino določanje sentimenta, kjer se ne osredotočamo na jezikovne konstrukte, kot je v primeru prvega nivoja dokument, ampak na mnenje samo. Mnenje sestoji iz sentimenta in tarče mnenja. Za primer lahko vzamemo besedilo *"Na mojem Ford Focusu mi je zelo všeč podvozje, zmogljivosti pa so na žalost povprečne. Tudi poraba goriva ni najboljša."*, kjer je mnenje izraženo s tremi vidiki, in sicer pozitivno glede podvozja in negativno glede zmogljivosti ter porabe goriva. Šele ta nivo analize sentimenta nam omogoča, da nestrukturirano besedilo predstavimo s strukturiranimi podatki. Na sliki 1.1 vidimo, kako bi lahko za zgornji primer strukturirano prikazali mnenja po entiteti (*avtomobil Ford Focus*) in posameznih vidikih. Ta nivo je daleč najzahtevnejši za implementacijo. Poleg same identifikacije in opredelitve mnenja je potrebno identificirati entitete in posamezne vidike, kar je zelo težka naloga že sama po sebi. Denimo eksplici-

tno izpostavljen vidik *zmogljivosti vozila* bi lahko v stavku nadomestili z *"... žal pa avto ni ravno hiter ..."*. Identifikacija vidika *zmogljivosti vozila* je v tem primeru izredno težavna naloga.

Slika 1.1: Strukturiran prikaz rezultatov analize sentimenta na nivoju entitete in posameznih vidikov.

Pri analizi sentimenta sta se uveljavila predvsem dva pristopa, leksikalni in pristop s strojnim učenjem. Pri **leksikalni metodi** (na sliki 1.2) potrebujemo enega ali več slovarjev sentimenta. Slednji vključujejo besede (lahko tudi fraze) s pozitivno in negativno konotacijo. Algoritem takšne besede in fraze v besedilu, ki ga želimo klasificirati, identificira. V kolikor prevladujejo besede z negativno konotacijo, takšno besedilo označi kot negativno. In obratno, v kolikor je identificiranih več besed s pozitivno konotacijo, besedilo izžareva pozitiven sentiment, zato je takšno besedilo klasificirano kot pozitivno. Osnovni problem pri tem pristopu je, da je potrebno ogromno vloženega dela za izdelavo slovarja. Izrazoslovje se s časom pogosto spreminja, zato je potrebno veliko sprotnega dela s prilagajanjem slovarja. Najboljše bi bilo, da bi nad takšnim slovarjem bedeli eksperti. Pri **pristopu s stroj-**

nim učenjem se preko (ročno) označenega besedila, kateremu se priredi ena od sentimentnih kategorij (pozitivno, negativno ipd.), tvori učna množica, na podlagi katere se zgradi statistični model za klasifikacijo. Nekaj najbolj popularnih metod strojnega učenja pri analizi sentimenta bo predstavljeno v poglavju 2. S tem pristopom so raziskovalci dosegli boljše rezultate kot z identifikacijo sentimentnih besed s slovarji. Za analizo sentimenta v slovenskem jeziku predstavlja težavo predvsem pomanjkanje dostopnih korpusov označenega besedila. Medtem ko je za angleščino veliko takšnih korpusov javno dostopnih, za slovenščino temu ni tako. Zato smo se odločili, da korpus izdelamo sami. Potek izdelave korpusa je predstavljen v poglavju 4. V našem delu smo se osredotočili na pristop z uporabo metod strojnega učenja. Leksikalne vire smo uporabili kot možne izboljšave statističnega modela (več v poglavju 5).



Slika 1.2: Leksikalna metoda za analizo sentimenta.

1.2 Analiza sentimenta uporabniško generiranih vsebin

S pojmom uporabniško generiranih vsebin (UGC) lahko označimo vsebine, ki so plod uporabnikov storitve ali sistema. Do teh vsebin lahko dostopajo drugi uporabniki storitve ali sistema. Pod UGC štejemo komentarje uporabnikov, objave na blogih, slike, video posnetke ipd. Za analizo sentimenta so zanimive predvsem tiste UGC, ki so v tekstovni, torej besedilni obliki.

Z razvojem spleta 2.0 (angl. *Web 2.0*) na začetku tega stoletja, ki temelji na "interaktivni" rabi spleta, se je oblika in količina UGC zelo povečala. Avtor vsebin na spletu je tako lahko postal vsakdo. Sodelovanje pri tvorjenju vsebin s strani uporabnikov interneta je postala značilnost spleta 2.0. Za razliko od spleta 1.0, pri katerem so uporabniki samo pregledovali vsebine, s spletom 2.0 vsebine tudi aktivno sooblikujejo. Tako so ljudje dobili možnost komentiranja različnih dogodkov na blogih, sodelovanja v družabnih omrežjih ipd. [5].

Raziskovalci iz različnih področij so s tem, ko lahko vsakdo izmed velike množice uporabnikov spleta prispeva svoje misli, videnja, odzive, dobili neusahljiv vir UGC, ki jih s pridom uporabijo v svojih raziskavah. Analiza sentimenta je s popularizacijo UGC doživela velikanski zalet. Z analiziranjem UGC se lahko pridobi neposreden vpogled v mišljenje ljudi. Politične stranke lahko preko komentarjev uporabnikov analizirajo odzive na posamezne akcije v predvolilni kampanji. Slednjo lahko sproti prilagajajo. Drage ankete, ki običajno zajamejo majhen odstotek volilcev, tako zamenjajo neposredna mnenja tisočih. Ljudje svoja mnenja delijo iz varnega zavetja domačega fotelja in (vsaj navidezne) anonimnosti, kar zagotavlja precejšnjo iskrenost takšnih mnenj. Podobno lahko trgovci z mnenji kupcev izmerijo stopnjo zadovoljstva z novimi artikli na policah, pripravljajo razne prodajne akcije, ki zajamejo čim širši krog potencialnih kupcev ipd. In nenazadnje, potrošniki lahko z analizo UGC pred nakupom določenega artikla pridobijo odzive obstoječih kupcev ali si pred obiskom kina preberejo mnenja gledalcev o filmu, ki si ga nameravajo ogledati. Kakšen vpliv imajo mnenja drugih, je razvidno

iz [6]:

- "81% uporabnikov interneta je že brskalo po spletu za različnimi izdelki";
- "med bralci spletnih opisov restavracij, hotelov in drugih storitev jih je med 73 in 87% priznalo, da so imeli opisi velik vpliv na njihovo odločitev o nakupu";
- "potrošniki so pripravljeni plačati od 20 do 99% več za izdelke, ki so ocenjeni s 5. zvedicami v primerjavi s tistimi, ki so ocenjeni s 4. zvezdicami";
- "32% jih je že ocenilo izdelek, storitev ali dalo oceno na delo kakšne osebe preko spletnih sistemov, ki omogočajo ocenjevanje";
- "30% jih je že komentiralo izdelek ali storitev".

Izmed UGC smo se v našem delu osredotočili na komentarje uporabnikov, konkretneje na komentarje uporabnikov slovenskih spletnih novičarskih portalov. Praktično vsi pomembnejši ponudniki novic (dnevniki, televizijske postaje ipd.) so zastopani na spletu. Večina svojim bralcem omogoča komentiranje pod novicami[1]. Analiza sentimenta že sama po sebi predstavlja velik izziv, pri analizi uporabniških komentarjev pa se pojavijo vsaj še naslednje težave [7, 8]:

- Besedilo pogosto vsebuje elemente ironije in sarkazma.
- Vsebina je pogostokrat slovnično nepravilna, napisana v neformalnem stilu ter vsebuje pravopisne napake.

[1]V zadnjem času se na žalost pojavlja trend, da nekateri uporabniške komentarje izključujejo. Takšen primer je spletna izdaja časnika Dnevnik, ki se nahaja na naslovu `http://www.dnevnik.si`. Spet drugi komentarje svojih bralcev hranijo le določeno časovno obdobje, denimo spletni portal Siol, ki se nahaja na naslovu `http://www.siol.net`.

- Besedilo je dodatno obogateno z emotikoni, povezavami, okrajšavami in slengom. Predobdelava besedila je zato bistveno bolj zahtevna in pomembna kot v primeru bolj formalnih besedil.
- Pri komentarjih gre ponavadi za krajša besedila. Po eni strani je to prednost, saj so avtorji ponavadi bolj jedrnati. Po drugi pa lahko izraznost mnenja predstavlja ena sama beseda. Beseda lahko ni zastopana v leksikalnem viru ali se ne pojavi v učni množici, kar lahko pripelje do tega, da se takšno mnenje izgubi.
- Izrazoslovje se s časom hitro spreminja.
- Komentarji so z ozirom na novico pogostokrat nerelevantni.

Nekatere izmed zgornjih točk so težko rešljive s trenutno tehnologijo in znanjem, denimo sarkazem. Slednje predstavlja velik izziv že za človeka, kaj šele za sistem za prepoznavo mnenja. Pri sarkazmu gre za popolnoma normalne stavke nasprotnega pomena. Za uspešno reševanje tega problema je pogosto potrebno poznati kontekst. V tem diplomskem delu se s prepoznavo sarkazma ne bomo ukvarjali.

1.3 Cilji

Glavni cilj dela je izdelava klasifikatorja ter orodja, ki klasifikator uporablja, za klasifikacijo uporabniških komentarjev. Za analizo sentimenta uporabniških komentarjev smo se odločili, ker:

- smo želeli analizirati neformalne oblike besedil,
- uporabniki skozi komentarje pogosto neposredno izražajo svoje mnenje in misli,
- obstaja dovolj slovenskih spletnih portalov, ki omogočajo komentiranje in preko katerih lahko izluščimo zadostno število komentarjev za analizo,

- velika množica komentarjev omogoča izvedbo zanimivih eksperimentov.

Klasifikator mora biti sposoben uporabniške komentarje razvrstiti v eno izmed treh kategorij, in sicer med pozitivne, negativne ali nevtralne.

Pri našem delu smo postavili nekaj hipotez:

- Na podlagi uporabniških komentarjev je mogoče razviti zadovoljujoč model za analizo sentimenta nekaterim težavnostim navkljub (sarkazem, ironija, izpostavljanje več vidikov v enem komentarju ipd.).
- Uporaba slovarja sentimenta in ostalih leksikalnih virov pripomore k boljšim rezultatom klasifikacije tudi neformalnih besedil, kar uporabniški komentarji so.
- Predobdelava besedila bistveno izboljša samo klasifikacijo. Sledimo predpostavki, da komentarji vsebujejo veliko šuma (neformalno besedilo, pravopisne napake ipd.), zato bi morala premišljena predobdelava besedila dati ustrezne rezultate.

Poleg glavnega cilja se pojavi potreba po izdelavi slovenskega slovarja sentimenta ter označenega korpusa slovenskih uporabniških komentarjev. Tako slovar kot korpus sta splošno uporabna in ju lahko pri svojem delu uporabijo vsi tisti, ki bi jih utegnila problematika analize sentimenta zanimati.

1.4 Obstoječa dela in primeri uporabe

Velika večina raziskav analize sentimenta je na voljo za angleški jezik. Tekom zadnjih let se je nabralo tudi nekaj poizkusov reševanja te problematike za slovenščino. V nadaljevanju podrobneje predstavimo obstoječe poskuse analize sentimenta za slovenski jezik.

Brina Škoda se v svoji diplomski nalogi [9] osredotoči na klasifikacijo komentarjev enega izmed treh najbolj priljubljenih novičarskih portalov v državi. Spletne komentarje so najprej ročno označili. Razvrstili so jih v tri razrede, in sicer med pozitivne, negativne in nevtralne. Komentarje sta

označila dva označevalca (angl. *annotator*). Ugotovili so, da so komentarji med razredi precej neenakomerno porazdeljeni, razen pri kategoriji *šport*. Pri učenju klasifikatorja in pri evalvaciji so tako upoštevali samo komentarje iz te kategorije, ostale so zanemarili. Z različnimi metodami strojnega učenja so zgradili več klasifikatorjev. Za klasifikacijo komentarjev so uporabili izključno metode strojnega učenja. Najboljši rezultat je dala metoda SVM.

Rok Martinc v svojem magistrskem delu [10] izdela orodje za analizo sentimenta na družbenem omrežju Twitter. Za razliko od prej opisanega pristopa, v tem delu za klasifikacijo niso uporabili metod strojnega učenja, ampak so uporabili leksikalno metodo. Na podlagi seznama AFINN-111[2] so sestavili seznam besed ter vsaki priredili vrednost z razponom od -5 (skrajno negativno) do 5 (skrajno pozitivno). Vrednost predstavlja sentimentno oceno posamezne besede. Za lažjo ponazoritev je v tabeli 1.2 prikazanih nekaj naključno izbranih besed s pripadajočimi sentimentnimi ocenami. Sentiment ugotovimo tako, da besede v besedilu primerjamo s seznamom besed, seštejemo sentimentne ocene teh besed in izračunamo njihovo srednjo vrednost. V kolikor je seštevek negativen, je besedilo klasificirano kot negativno; v kolikor je seštevek pozitiven, ima besedilo pozitivno konotacijo.

Medtem ko je pri prvih dveh omenjenih delih fokus na klasifikaciji neformalnih besedil, se je Mateja Volčanšek ukvarjala z analizo sentimenta bolj formalnih besedil - vsakodnevnih novic [11]. Tudi v tem delu so besedila klasificirali z leksikalno analizo. Primaren cilj je bil sestava kakovostnega slovenskega slovarja sentimenta. Za osnovno so vzeli angleški leksikon General Inquirer[3]. Iz dveh kategorij, *Positiv* in *Negativ*, so z uporabo avtomatskega prevajanja in ročnega preverjanja sestavili slovenski slovar sentimenta. Končna verzija slovarja šteje 1669 pozitivnih in 1912 negativnih besed. Za potrebe evalvacije modela so označili 5000 novic s spleta. Z rezultati klasifikacije niso bili povsem zadovoljni. Napram večinskemu klasifikatorju se je klasifikacija izboljšala za okrog 5%.

[2] `http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010`
[3] `http://www.wjh.harvard.edu/~inquirer/`

ocena	besede
5	hura
4	hud, mojstrovina, najboljši
3	drzen, fascinanten, hvali
2	ekskluziven, eleganten, gojiti
1	bog, diamant, dogovoriti
-1	ambivalenten, anti, bankir
-2	aretacija, aroganten, bes
-3	brezbrižen, depresiven, dolgočasen
-4	faker, jezni, kreten
-5	baraba, nepridiprav, svinja

Tabela 1.2: Primeri besed iz slovenske tabele AFINN-111 s pripadajočo sentimentno oceno.

Matej Martinc se v svojem diplomskem delu [12] primarno posveča primerjavi različnih knjižnic za obdelavo naravnega jezika za programski jezik Python, vendar ne izostane niti posplošen pregled različnih pristopov analize sentimenta. V delu je predstavljen tako leksikalni pristop (z uporabo slovenske tabele AFINN-111 [10]), kot tudi pristop z uporabo metod strojnega učenja. Pri slednjem je poudarek na zmožnostih klasifikacije posameznih knjižnic. Kot zanimivost lahko navedemo, da se je avtor lotil avtomatske izdelave korpusa označenih besedil za potrebe analize sentimenta. Preko Twitter API-ja je pridobil množico tvitov v slovenščini, ki so vsebovali pozitivne ali negativne emotikone. V razponu dveh tednov je z orodjem zbral dobrih 6000 tvitov s pozitivnimi emotikoni ter okoli 700 tvitov z negativnimi emotikoni. Avtorju je s tem pristopom uspelo zgraditi klasifikator, ki je bil za približno 2% boljši od večinskega. Slab rezultat utemelji z dokaj majhno učno množico ter verjetnim šumom v podatkih.

V doktorski disertaciji [13] se Jasmina Smailović sprašuje ali lahko z analiziranjem Twitter sporočil napovedujemo bodoča gibanja tečajev delnic pod-

jetij. Z uporabo strojnega učenja so prišli do zaključka, da bi lahko z ugotavljanjem sentimenta sporočil napovedovali vrednosti tečajev za nekaj dni vnaprej. Tudi v tem delu so preizkusili več metod strojnega učenja. Podobno kot v [9] pridejo do zaključka, da se pri analizi sentimenta najboljše obnese metoda podpornih vektorjev. Delo je zanimivo vsaj iz treh vidikov. Prvič, veliko pozornosti so posvetili predobdelavi besedila, v tem primeru tvitov. Analizirali so sestavo tvita in izvedli nekaj specifičnih predobdelav. Nekaj primerov predobdelave besedila je prikazanih v tabeli 1.3. Drugič, Twitter sporočila vedno ne nosijo elementov subjektivnosti. Za takšne primere uvedejo nevtralno cono. S klasifikatorjem SVM sporočila uvrstijo med pozitivne, negativne ali nevtralne. Posebnost je v tem, da je z uporabo nevtralne cone sporočila možno klasificirati kot nevtralna, kljub temu da v učni množici ni bilo tovrstnih primerov. Sporočila so klasificirana kot nevtralna, v kolikor so projecirana v območje blizu hiperravnine SVM (glej sliko 1.3). To območje so definirali kot nevtralno cono, ki je parametrizirana z vrednostjo t, kjer t predstavlja pozitivno in $-t$ negativno mejo nevtralne cone. Če je sporočilo x, z razdaljo $d(x)$ od hiperravnine, projecirano v to cono, se klasificira kot nevtralno. Tretjič, klasifikator se je sposoben, z uporabo aktivnega učenja, prilagajati novim primerom sporočil. Aktivno učenje se uporablja tako, da se za določene mejne primere izvede ročna označba človeških označevalcev. Takšni primeri se uporabijo za inkrementalno posodabljanje klasifikatorja.

predobdelava	**rezultat**
uporabniška imena	@JanezNovak → atttJanezNovak
simboli za delnice	$GOOG → stockGOOG
spletne povezave	www.fri.uni-lj.si → URL
hashtagi	#volitve2014 → hashvolitve2014
negacija	not, isn't, aren't → NEGATION

Tabela 1.3: Predobdelava Twitter sporočil.

Poglejmo še praktične primere uporabe analize sentimenta. Slovensko

Slika 1.3: Klasifikacija nevtralnih sporočil preko nevtralne cone z uporabo metode SVM.

podjetje Gama System d.o.o.[4] je v sodelovanju z največjo komercialno televizijsko postajo POP TV[5] spremljalo predsedniške volitve 2012. Aplikacija je v realnem času zbirala objave iz družbenih omrežij. Vse relevantne objave so analizirali in na nekaj minut preračunavali t.i. politični indeks. Politični indeks je bil izračunan na podlagi absolutne razlike med številom pozitivnih in negativnih sentimentov zaznanih v javnem mnenju [14]. Merjenje se je izkazalo za precej natančno. Kandidat z najvišjim indeksom je na koncu tudi dejansko zmagal. Glede na to, da so aplikacijo gledalci v živo spremljali v najbolj gledanem delu dneva (angl. *prime-time*), je to pomenilo velik uspeh za predstavljeno tehnologijo. Aplikacija (na sliki 1.4) je bila v času kampanje dosegljiva 24 ur dnevno preko spletne strani *www.predsedniskevolitve.si*.

[4] `http://www.gama-system.si/`
[5] `http://pro-plus.si/slo/pro_plus/televizija/`

Slika 1.4: Aplikacija, preko katere so gledalci spremljali t.i. politični indeks kandidatov v okviru predsedniške kampanje 2012.

Na podoben način so lahko gledalci spremljali tudi Evropsko prvenstvo v košarki 2013 v okviru televizijske postaje ŠPORT TV[6]. Analizirali so odzive na tekme, skupine in igralce.

Obe navedeni aplikaciji za spremljanje javnega mnenja preko družbenih omrežij sta temeljili na analitični platformi Gama System PerceptionAnalytics[7]. Platforma PerceptionAnalytics omogoča spremljanje javnega mnenja o določeni temi, podjetju, osebi, blagovni znamki, dogodku ipd. v realnem času preko družbenih omrežij [15]. Platforma za klasifikacijo objav iz družbenih omrežij uporablja metodologijo, ki je bila razvita v okviru doktroske disertacije Jasmine Smailović [13] in je bila v predhodnih odstavkih že predstavljena. Morda bi bilo potrebno na tem mestu dodati še, da je uporabljena metodologija analize sentimenta zelo prilagodljiva, saj so jo prilagodili

[6] `http://www.sport-tv.si`
[7] `http://www.perceptionanalytics.net`

za delovanje z različnimi jeziki: angleščina, slovenščina, španščina, nemščina ipd. (popoln seznam podprtih jezikov je objavljen v [13]).

1.5 Struktura diplomskega dela

Po uvodu, v katerem opredelimo osnovne pojme in bralcu predstavimo problematiko analize sentimenta, sledi obširno poglavje o klasifikaciji sentimenta. V njem spoznamo različne pristope pri reševanju obravnavane problematike, predstavimo nekaj najbolj pogosto uporabljanih metod strojnega učenja, izbiro značilk ter tehnike (pred)obdelave besedila za namen analize sentimenta.

V poglavju 3 predstavimo glavna orodja in tehnologije, ki smo jih uporabili pri razvoju orodja za klasifikacijo sentimenta. V prvem delu poglavja so predstavljeni programski jeziki in knjižnice, sledi predstavitev spletnih tehnologij ter platforme, s katero našo aplikacijo odpremo svetu.

Poglavje "Izdelava orodja za klasifikacijo uporabniških komentarjev"je namenjeno predstavitvi praktičnega dela diplomske naloge. Opis sledi posameznim sklopom orodja, in sicer je razdeljeno na slovar sentimenta, označen korpus uporabniških komentarjev ter sam klasifikator besedila. Proti koncu poglavja sledi še opis metod spletne storitve, preko katere imajo zunanji odjemalci možnost koriščenja funkcij našega sistema z izmenjavo sporočil XML.

Sledi poglavje z rezultati, ki jih naš klasifikator dosega na označenem korpusu uporabniških komentarjev nekaterih najbolj znanih slovenskih novičarskih portalov. Ocenimo uspešnost klasifikatorja ter rezultate tudi kritično ovrednotimo. Diplomsko delo zaključimo z odgovorom na vprašanje “Smo zastavljene cilje dosegli?”. Podamo nekaj možnosti za nadaljnje delo na razvitem orodju ter opredelimo nekaj točk, s katerimi bi klasifikacijo lahko potencialno še izboljšali.

Poglavje 2

Klasifikacija sentimenta

Osredotočili smo se na analizo sentimenta na nivoju celotnega dokumenta. Analizi sentimenta na tem nivoju pravimo tudi klasifikacija sentimenta (angl. *sentiment classification*) [3]. Za vhodno besedilo nas zanima splošno piščevo razpoloženje. Za primer lahko navedemo opis filma, ki izraža bodisi pozitivno ali negativno mnenje o filmu, na katerega se nanaša opis. Ne zanima nas, kaj si avtor misli o določenem igralcu ali posebnih efektih. Prav tako nas ne zanima morebitna sprememba mnenja med deli opisa. Zanima nas mnenje na nivoju celotnega opisa.

Klasifikacija sentimenta je le eno izmed opravil v domeni klasifikacije besedil. Klasifikacijo lahko opredelimo kot opravilo, s katerim za dan vhod izberemo pravilno oznako razreda. Množica oznak je običajno določena vnaprej. Oznaki razreda pravimo tudi kategorija, klasifikacijo pa lahko imenujemo tudi kategorizacija. V bolj formalizirani obliki lahko klasifikacijo dokumentov predstavimo z množico dokumentov D in množico (predefiniranih) kategorij C, s ciljem dodelitve logične vrednosti za vsak par $\langle d_i, c_j \rangle$, kjer je $d_i \in D$ in $c_j \in C$. Če je paru $\langle d_i, c_j \rangle$ dodeljena logična vrednost *true*, pomeni, da dokument d_i spada v kategorijo c_j [16]. V domeni klasifikacije sentimenta so običajno uporabljene oznake *pozitivno*, *negativno* ter *nevtralno*. Za druge primere klasifikacije besedila lahko navedemo:

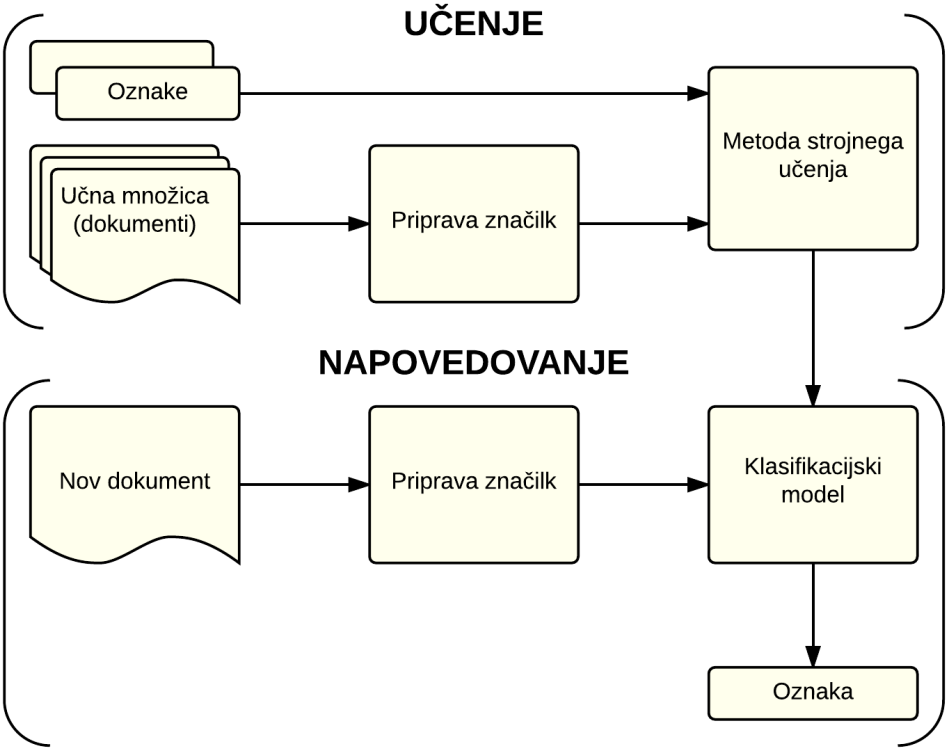
- filtriranje nezaželene elektronske pošte,

- določanje teme novicam (šport, gospodarstvo, politika ipd.),
- prepoznava jezika v besedilu,
- organizacija dokumentov,
- preverjanje plagiatorstva.

Klasifikacija sentimenta velja za eno težjih nalog v domeni klasifikacije besedila. Kljub temu, da pri klasifikaciji sentimenta besedilo običajno kategoriziramo samo z dvema ali tremi oznakami in bi na prvi pogled pričakovali zelo dobre rezultate, temu pogosto ni tako. Pri nekaterih drugih opravilih je denimo število oznak precej večje. Za kategorizacijo novic je lahko oznak deset in več (šport, gospodarstvo, politika, svet, zabava ipd.) in kljub temu klasifikatorji dosegajo zelo dobre rezultate. Medtem ko pri določanju tem v dokumentu zadostujejo že ključne besede (za kategorijo *šport* je terminologija bistveno drugačna od denimo kategorije *politika*), pa je lahko sentiment izražen na precej bolj subtilen način [17]. Stavek *"A pa ta film je kdo resnično gledal do konca?"* ne vsebuje niti besede, ki bi nakazovala negativno percepcijo avtorja do filma, pa kljub temu vemo, da avtor o filmu ne misli nič dobrega. Boljši avtomatizirani sistemi za klasifikacijo sentimenta naj bi tako dosegali klasifikacijsko točnost okoli 80%, kar je povsem dovolj za spremljanje trendov (denimo priljubljenost neke politične stranke skozi čas) vendar premalo za bolj poglobljene raziskave. Za razliko od analize sentimenta, je točnost kategorizacije tem v dokumentih skoraj enaka človeškim ocenjevalcem [18].

V splošnem klasifikacijo razdelimo na **nadzorovano** (angl. *supervised*) in **nenadzorovano** (angl. *unsupervised*). O nenadzorovani klasifikaciji dokumentov govorimo takrat, ko učna množica in predefinirane kategorije ne obstajajo. Dokumentom se dodeljuje oznake izključno na podlagi vsebine. V primeru nadzorovane klasifikacije obstaja učna množica in znana množica oznak razredov [19]. V tej nalogi se v celoti osredotočimo na klasifikacijo sentimenta z uporabo nadzorovanih metod strojnega učenja. Na sliki 2.1

vidimo osnovni diagram nadzorovane klasifikacije. V fazi učenja najprej pripravimo značilke. Z množicami značilk in predefiniranih oznak se z metodo (algoritmom) strojnega učenja zgradi klasifikacijski model. V fazi napovedovanja nov dokument prav tako pretvorimo v množico značilk. Množico značilk pošljemo v klasifikacijski model, ki vrne oznako [20].



Slika 2.1: Nadzorovana klasifikacija.

Pogoja za uspešno nadzorovano klasifikacijo sentimenta z metodami strojnega učenja je v izbiri učinkovitega algoritma za strojno učenje in v izbiri in uteževanju ustreznih značilk. Zato v nadaljevanju poglavja sledi predstavitev nekaj bolj priljubljenih metod strojnega učenja za klasifikacijo sentimenta. Posebno sekcijo posvečamo predobdelavi besedila, ki je posebej pomembna pri klasifikaciji sentimenta neformalnih besedil, kar uporabniški komentarji so. Poglavje zaključujemo s pregledom tehnik pri izbiri in izločevanju značilk.

2.1 Nadzorovane metode za klasifikacijo sentimenta

Večina raziskav analize sentimenta uporablja za klasifikacijo metode strojnega učenja. V tem razdelku predstavljamo nekaj metod, ki so se uveljavile pri klasifikaciji sentimenta.

2.1.1 Naivni Bayes

Prva izmed metod temelji na Bayesovem teoremu, ki ga predstavimo z enačbo:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)} \tag{2.1}$$

P(c|d) je aposteriorna verjetnost, ki je izračunana iz apriornih verjetnosti P(c), P(d|c) in P(d). c predstavlja hipotezo, v našem primeru oznako razreda oziroma eno izmed kategorij *pozitivno*, *negativno* ali *nevtralno*. d predstavlja dokument, primer iz učne množice. Z izrazom P(c|d) nas torej zanima, kakšna je verjetnost, da je dokument d uvrščen v kategorijo c. Na izbiro kategorije c P(d) nima vpliva, zato ga lahko eliminiramo. Za izračun P(d|c) naivni Bayes (NB) "naivno" predpostavlja, da so značilke pogojno neodvisne. Kot primer lahko navedemo, da je dokument lahko kategoriziran kot *pozitiven*, če vsebuje besedo *super*, podpis in pozitiven emotikon. NB predpostavlja, da omenjene značilke neodvisno prispevajo k temu, da je dokument klasificiran kot *pozitiven*, povsem neodvisno od prisotnosti ali odsotnosti drugih značilk [21]. Navkljub pričakovanju, da pogojna neodvisnost predstavlja težavo, se NB odreže dobro pri klasifikaciji besedil [17]. NB izračuna verjetnost za vsako izmed kategorij. Dokument uvrsti v kategorijo z največjo verjetnostjo. Maksimalno aposteriorno oceno (MAP) izrazimo kot:

$$c_{MAP} = argmax_{c \in C} P(c)P(d|c) \tag{2.2}$$

Poznamo več modelov NB, ki se med seboj razlikujejo po tem, kako se izračuna verjetnost P(d|c). V domeni klasifikacije besedil sta se uveljavila

dva. **Multivariatni Bernoullijev naivni Bayes (BNB)** za vsebovanost značilke (denimo besede) v dokumentu uporablja binarno informacijo. Upošteva se, ali beseda v dokumentu obstaja (vrednost 1) ali ne (vrednost 0). **Multinomski naivni Bayes (MNB)** za razliko od BNB upošteva tudi število pojavitev besede v dokumentu. McCallum in Nigam [22] sta podala lep primer (na sliki 2.2), ki opisuje to razliko pri klasifikaciji. Vzela sta primer pojavljanja števil v člankih. Običajno je vsak članek datiran zato vsebuje vsaj eno pojavitev števila. Za nekatere članke je značilno, da vsebujejo veliko števil v besedilu (denimo novice o borznem trgovanju). V tem primeru bi informacija o številu pojavitev te značilke lahko vplivala na boljšo klasifikacijo. V primeru BNB bi bila značilka, ki bi vsebovala informacijo o vsebovanosti števil povsem neinformativna. In sedaj pridemo še do druge večje razlike. Medtem ko v primeru BNB (ne)prisotnost značilke v dokumentu vpliva na klasifikacijo, pa pri MNB v primeru, da značilka v dokumentu ni prisotna, nima vpliva na klasifikacijo.

Borza | Skladi | Valute

Sklad	%	VEP	Datum
Triglav vzajemni skladi - Delniški Triglav Enegija	3,24	8,1200	21.1.16
NLB Skladi - Naravni viri delniški	2,97	2,4244	21.1.16
ILIRIKA Energija delniški	2,91	2,8432	21.1.16
KD Latinska Amerika, delniški	-0,85	0,8400	21.1.16
Perspektiva: EurAsia Flexible, mešani podsklad	-2,18	6,8083	21.1.16
Delniški podsklad ALTA ASIA	-1,37	5,7679	21.1.16

Vir: **Vzajemci.com**

Slika 2.2: Merjenje pogostosti pojavitve števil v dokumentu lahko pripomore k boljši klasifikaciji. Po izseku iz članka lahko sklepamo, da gre za članek o borznem dogajanju. Za takšne primere je izbira klasifikatorja BNB neprimerna.

2.1.2 Logistična regresija

Alternativa NB je logistična regresija (LR), ki se pri klasifikaciji besedila lahko odreže tudi bolje od prejšnje metode [17]. Klasifikator LR za razliko od NB ne predpostavlja pogojnih neodvisnosti med značilkami, kar je v domeni klasifikacije besedil pravilna predpostavka, saj so med značilkami tudi besede, ki pogosto niso neodvisne ena od druge. Ko želimo zajeti širši kontekst, bi lahko uporaba LR bila prednost. Slabost klasifikatorja LR je večja časovna zahtevnost v fazi učenja. Po izgradnji modela je s stališča porabe pomnilnika in procesorja primerljiv z drugimi klasifikatorji [23].

Verjetnost, da dokument spada v določeno kategorijo se oceni z naslednjo eksponentno formulo:

$$P_{LR}(c|d) = \frac{1}{Z(d)} exp\left(\sum_i \lambda_{i,c} Fi, c(d,c)\right), \tag{2.3}$$

kjer je Z(d) normalizacijska funkcija, $F_{i,c}$ predstavlja funkcijo značilka-kategorija za značilko f_i in kategorijo c, ki jo lahko definiramo kot:

$$F_{i,c}(d,c') = \begin{cases} 1, & n_i(d) > 0 \text{ in } c' = c \\ 0, & \text{v nasprotnem primeru} \end{cases} \tag{2.4}$$

Z izbrano množico značilk postavimo omejitve v modelu. Značilka se za par (d,c') denimo upošteva, če in samo če obstaja dokument z značilko, ki je povezana s c'. Kot primer lahko navedemo besedno zvezo *"tega ne maram"*, ki bo upoštevana samo za dokumente z negativnim sentimentom. Seveda ni nujno, da funkcija vrača samo binarno vrednost, lahko bi vračala število, denimo pogostost pojavitve besede ipd. $\lambda_{i,c}$ predstavlja parameter za določanje uteži značilk. Značilka f_i predstavlja močan indikator za razred c v primeru velikega $\lambda_{i,c}$ [17].

2.1.3 Metoda podpornih vektorjev

Metoda podpornih vektorjev (SVM) se je izkazala za eno boljših v domeni klasifikacije besedil. V [17], kjer so določali sentiment opisom filmov, prekaša

obe poprej predstavljeni metodi. Dobro se odreže tudi v primeru neformalnih besedil. V [24] so primerjali različne metode strojnega učenja na objavah iz platforme Twitter. Na podlagi teh dveh raziskav smo sklepali, da bi metoda utegnila dati dobre rezultate tudi pri analizi sentimenta neformalnih besedil v slovenskem jeziku. SVM dobro preprečuje pretirano prilagajanje rezultata klasifikacije učnim podatkom (angl. *overfitting*). Prav tako težav ne predstavlja niti velik prostor značilk, kar je v primeru klasifikacije sentimenta, ko lahko operiramo tudi z več sto tisoč značilkami, pomembna prednost [13].

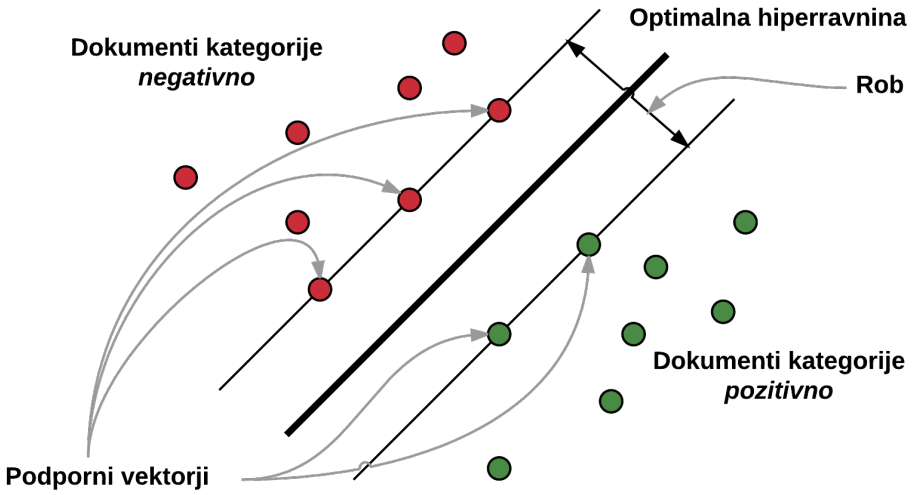
Medtem ko sta NB in LR verjetnostna (angl. *probabilistic*) klasifikatorja, je princip delovanja SVM drugačen. Cilj je poiskati takšno hiperravnino (angl. *hyperplane*), ki loči dokumente ene kategorije od dokumentov druge kategorije, pri čemer je ločitvena meja med kategorijami čim večja (na sliki 2.3). Čim večja je ta razdalja, manjšo klasifikacijsko napako lahko pričakujemo za nove dokumente [13]. Klasifikacija novih primerov se izračuna z naslednjo linearno funkcijo [13]:

$$D(x) = x \times w + b, \tag{2.5}$$

kjer je x vektor značilk (angl. *feature vector*) novega primera za klasifikacijo, b predstavlja prag, vektor uteži w predstavlja funkcijo podpornih vektorjev. V primeru, da je rezultat funkcije negativen, lahko primer klasificiramo kot negativen in obratno.

2.2 Predobdelava besedila

Pred zajemom relevantnih značilk je potrebno besedilo pripraviti in ga očistiti. Ta korak imenujemo predobdelava besedila. Naš primarni cilj je izgradnja modela za klasifikacijo uporabniških komentarjev. Tu je ta korak še posebej pomemben, saj tovrstno besedilo pogosto vsebuje veliko šuma. V besedilu je pogosta raba posebnih struktur, ki so velikokrat neinformativne z vidika klasifikacije sentimenta, denimo značke HTML. Z ustrezno predobdelavo besedila lahko izboljšamo rezultate klasifikacije. Prav tako lahko bistveno zreduciramo prostor značilk že v tem koraku. Zamislimo si množico besedil,

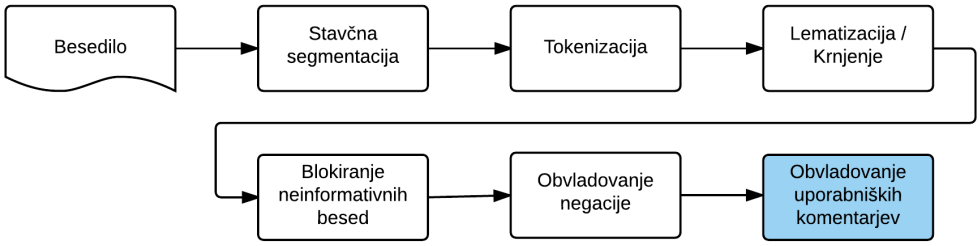


Slika 2.3: Princip metode SVM - poiskati hiperravnino z največjo razdaljo (ali robom) med dokumenti različnih kategorij.

ki vsebujejo različne oblike besede *uporabljati*. V učnih podatkih se lahko pojavijo besede *uporabljam*, *uporabljava*, *uporabljamo*, *uporabljajo* ipd. Z različnimi oblikami besede *uporabljati* smo v tem primeru pridobili kar štiri nepotrebne značilke. Z ustrezno predobdelavo besedila lahko opisan primer uspešno eliminiramo. V nadaljevanju predstavimo standardne pristope, ki se uporabljajo pri predobdelavi besedila ter nekaj načinov obvladovanja modernih, spletnih besedil, ki imajo svoje posebnosti (na sliki 2.4).

S **stavčno segmentacijo** (angl. *sentence segmentation*) besedilo členimo na posamezne stavke. Ker v tem delu izvajamo analizo sentimenta na nivoju celotnega besedila, je v izogib nejasnostim potrebno pojasniti, zakaj bi sploh želeli razdeliti besedilo na posamezne stavke. Ta korak smo izvajali zato, ker je bilo členjenje besedila na stavke v pomoč nadaljni obdelavi besedila. Kot primer lahko navedemo negacijo. Členjenje stavkov predstavlja pomemben del denimo tudi pri oblikoslovnem označevanju[1] (angl. *POS tagging*).

[1]Oblikoslovnega označevanja v tej nalogi nismo izvajali.



Slika 2.4: Koraki pri predobdelavi besedila.

Stavčna segmentacija je precej zahtevnejša naloga, kot je videti na prvi pogled. Za primer lahko navedemo stavek *"V letu 2010 je g. Novak preko svojega d.o.o. samo z delnicami (!!!) pridelal več kot 1.000 EUR minusa."*. Običajno z ločili *"!"*, *"?"* in *"."* ločimo posamezne stavke. Na zgornjem primeru opazimo, da smo s piko tudi krajšali besedo *"gospod"*, v krajši obliki zapisali družbo z omejeno odgovornostjo *"d.o.o."*, oblikovali število *"1.000"* ter s klicaji poudarili *"...samo z delnicami pridelal več kot 1.000 EUR minusa..."*. Slabo orodje za segmentacijo bi lahko zgornje besedilo razdelilo na vsaj tri stavke, kar je seveda narobe. S stališča stroja so lahko ločila zelo dvoumna.

Tokenizacija (angl. *tokenization*) je postopek, s katerim besedilo razdelimo na pojavnice. Pojavnice so besede in ločila [1]. Podobno kot pri segmentaciji lahko tudi tukaj naletimo na težave z okrajšavami, pojavijo se lahko težave z deljenimi besedami (denimo *"C-vitamin"*) ipd. Pri klasifikaciji neformalnih besedil je potrebno dodatno upoštevati posebne pojavnice, denimo emotikone. Tokenizacijo lahko izvedemo na več načinov. Pri tem se moramo zavedati, da izbira načina tokenizacije temelji na problemu, ki ga rešujemo, npr. tokenizacija neformalnih besedil je lahko drugačna od tokenizacije formalnih besedil. Tokenizacija s presledki je osnoven način ločitve besedila na pojavnice. Pri tej tokenizaciji pojavnice ločimo s presledki, znaki za novo vrstico in tabulatorjem. Med zelo uporabljane spada Treebank tokenizator, ki uporablja regularne izraze. Christopher Potts [25] je izdelal

poseben tokenizator, ki upošteva nekatere dodatne vidike, ki se pojavijo pri analizi sentimenta. V tabeli 1.2 je prikazana primerjava različnih tokenizatorjev. Tudi v našem delu smo merili vpliv tokenizacije na rezultat klasifikacije (več v poglavju 5).

Tok. s presledki	**Tokenizator Treebank**	**Tokenizator Potts**
ej	ej	ej
@novak,	@	@novak
kako	novak	,
si	,	kako
dons?	kako	si
:-)	si	dons
	dons	?
	?	:-)
	:	
	-	
	)	

Tabela 2.1: Primerjava različnih tokenizatorjev na besedilu *"ej @novak, kako si dons? :-)"*.

Z **lematizacijo** (angl. *lemmatization*) besedam določamo njihovo osnovno obliko - lemo (angl. *lemma*). Leme si lahko predstavljamo kot slovarske oblike besed. Ker slovenščina spada med pregibno bogate jezike, je lematizacija kompleksna. Posamezne besede imajo lahko po več deset pregibnih oblik [26]. Lematizacija uporabniških komentarjev in drugih UGC predstavlja še dodatno težavo, saj takšna besedila vsebujejo veliko skovank in pisanja v žargonu. Lematizacijska pravila vsega tega ne zaobjamejo [1]. Enostavnejši postopek od lematizacije je **krnjenje** (angl. *stemming*). Pri krnjenju določamo krn besede - besedi se odreže pripona. Slabost je ta, da se lahko del informacije izgubi, saj ima lahko več različnih besed isti koren. Izmed obeh načinov normalizacije besed smo v našem delu uporabili lematizacijo. V tabeli 2.2 je prikazan princip lematizacije in krnjenja. Z normalizacijo

besed pri klasifikaciji lahko bistveno zmanjšamo prostor značilk.

Lematizacija	**Krnjenje**
pisati	pi

Tabela 2.2: Lematizacija in krnjenje na primeru besede *"pišemo"*.

Z **blokiranjem neinformativnih besed** (angl. *stop words*) poskrbimo, da nepomembne besede za klasifikacijo sentimenta odstranimo. Te pomensko šibke besede (npr. *kajti, ter, kar, tako*) običajno dobimo v obliki seznamov. Obstajajo tudi tehnike za dinamičnejše iskanje takšnih besed. Z njihovim odstranjevanjem zmanjšamo šum in posledično izboljšamo rezultate klasifikacije. Raziskovalci, ki se ukvarjajo z analizo sentimenta, so deljenega mnenja glede rabe tega postopka. Nekateri so z odstranjevanjem izboljšali klasifikacijo, spet drugi rezultate poslabšali. Pri eksperimentiranju bomo preizkusili klasifikacijo z in brez odstranjevanja teh besed in videli ali nemara tudi takšne besede nosijo informacijo o sentimentu.

Z **obvladovanjem negacije** skušamo zaznati spremembo polarnosti besed. Na primeru *"Meni pa ni všeč najnovejši iPhone."* je razvidno, da beseda *"ni"* služi kot negator besede *"všeč"*, ki sicer vzbuja pozitivno občutje. Negacija običajno obrne izraženo mnenje, ni pa vedno tako. Negacija besede z negativno konotacijo denimo obrne sentiment v pozitiven (*"ni problema"*). Dodaten problem so fraze. V stavku *"iPhone je res super ne samo zato, ker je lepe oblike."* negator *"ne"* ne naredi ničesar. Zaznavanje negacije je zaradi vseh pravil kompleksna naloga. Za popolno obvladovanje negacije bi morali identificirati besede in pa tudi fraze, ki nosijo sentimentno vrednost ter s pravili pravilno popraviti polariteto besede ali besednih zvez na mestu, kjer se negacija pojavi. V našem delu se s posebnimi lingvističnimi pravili okrog problema negacije nismo ukvarjali. Negacijo smo izpostavili z dodajanjem posebne značke *"NEG_w"* vsem besedam, ki sledijo negatorju. V navezi s stavčno segmentacijo lahko bolj natančno opredelimo doseg negatorja, lahko pa tudi preprosto s prvim naslednjim ločilom ustavimo domet negacije. Stavek *"Jaz ne maram iPhonea zaradi njegove oblike."* se tako

preoblikuje v *"Jaz ne NEG_maram NEG_iPhonea NEG_zaradi NEG_njegove NEG_oblike."*. Pri obvladovanju negacije smo se zgledovali po raziskavi Pang in sod. [17].

Obvladovanje uporabniških komentarjev predstavlja zadnji sklop pristopov za predobdelavo besedila. Medtem, ko so bili prejšnji koraki splošno usmerjeni, tukaj rešujemo nekaj posebnosti, povezanih z načinom komentiranja uporabnikov, kot so:

- Spletne povezave (URL): uporabniki v svojih komentarjih pogosto dodajajo povezave na bolj ali manj relevantne zunanje vsebine. Naš klasifikator takšnih vsebin ne zajame. Zaradi možnosti, da pojavnost povezav v besedilu pripomore k boljši klasifikaciji, vse povezave zamenjamo z *"URL"*. Besedilo *"...jutri bo vroče. Več na http://www.rtvslo.si/..."* z normalizacijo povezav preuredimo v *"...jutri bo vroče. Več na URL..."*.

- Identifikator uporabnika: v spletnih komentarjih se za omembo drugega uporabnika uporablja predpona @ k uporabniškemu imenu (npr. *"@janeznovak"*). Na ta način uporabnik odgovarja drugemu uporabniku (nekateri portali ne omogočajo neposrednih replik v komentarjih), se strinja, oziroma nasprotuje mislim drugega uporabnika ipd. Podobno kot pri spletnih povezavah tudi tukaj naredimo zamenjavo. Identifikator uporabnika zamenjamo z besedo *"USER"*.

- Ponavljanje črk v besedi: uporabniki pretirano veselje ali žalost pogosto izkazujejo skozi ponavljajoče se zaporedje črk v besedi (*gooooooool*, *neeee* ipd.). Zaradi velikosti slovarja in posledično števila značilk, zaporedja skrajšamo na največ tri znake (iz *gooooooool* v *goool*). Krajšanju na dva znaka smo se želeli izogniti zaradi možnosti zlivanja pojmov, saj obstajajo veljavne besede z zaporedjem istih črk (denimo *kooperacija*).

- Emotikoni: pogosteje jih imenujemo smeškoti in so pomemben del spletnih besedil. Za avtomatiziran sistem prepoznave sentimenta je nujno, da emotikone pravilno obravnava, saj z njimi avtorji izražajo veselje, žalost, presenečenje ipd. S pravilnim obvladovanjem emotikonov bi

lahko lažje izluščili dejanski pomen, saj pogostokrat izboljšajo interpretacijo zapisanih besed [1]. Emotikone najpogosteje sestavljajo posebni znaki (*":)"*, *":"*, *"="* ipd.) in oponašajo obrazno mimiko. Pri obravnavanju emotikonov lahko uporabljamo posebne slovarje, ki vsebujejo emotikone in njihovo sentimentno oceno. Emotikone lahko tudi normaliziramo, torej tiste, ki izražajo veselje (*":)"*, *":D"* ipd.) uvrstimo v eno, tiste, ki izražajo žalost (*":("*, *":-("* ipd.) pa v drugo skupino.

- Mešanje malih in velikih črk: v uporabniških komentarjih pogosto naletimo na primere, ko se v besedi pojavi kombinacija velikih in malih črk, npr. *"PoZdrAvljEn"*. Za zagotavljanje istih oblik lahko vse besede preoblikujemo tako, da vsebujejo samo male črke. Ker pa lahko na identifikacijo sentimenta vpliva tudi pojavnost besed v obliki samih velikih črk, s čimer avtor še poudari napisano, lahko v izogib izgubi takšnih informacij to rešimo z dodatnimi značilkami, še preden besedilo pretvorimo.

2.3 Priprava značilk

Pred izgradnjo klasifikacijskega modela je potrebno zajeti značilke. Značilke naj predstavljajo čim relevantnejše vidike testnih podatkov. Premalo skrbna izbira značilk lahko botruje k izgradnji klasifikacijskega modela z veliko šuma in slabo klasifikacijsko točnostjo. Pri klasifikaciji besedil nastopa običajno mnogo značilk. Vsaka beseda, ki se pojavi med učnimi podatki lahko predstavlja svojo značilko. V nadaljevanju poglavja so opisani splošni pristopi pri pripravi značilk za izgradnjo klasifikacijskega modela.

2.3.1 Vektorski prostor

Besedilo je potrebno pretvoriti v obliko, ki jo razume klasifikator. Pri klasifikaciji besedila se običajno učni podatki pri učenju in neoznačeni primeri v fazi napovedovanja preslikajo v vektorski prostor. Komponente vektorskega

prostora imenujemo značilke. Množico dokumentov in terminov (besede, besedne zveze ipd.) predstavimo z vektorji v večdimenzionalnem geometrijskem prostoru. V tabeli 2.3 vidimo predstavitev vektorskega prostora z dvojiškimi vrednostmi. V stolpcih so termini, v vrsticah dokumenti. V kolikor se posamezen termin nahaja v dokumentu ima vrednost 1, če ne 0. Na tem preprostem modelu vidimo, da so vsi termini enako pomembni, kar pa ni vedno zaželjeno. Običajno se v praksi uporablja uteževanje, s čimer pozameznim dimenzijam dvignemo pomembnost.

	rad	imam	potovanja
dokument 1	1	0	1
dokument 2	0	1	1
dokument 3	1	1	1
...	...	...	...
dokument N	0	0	0

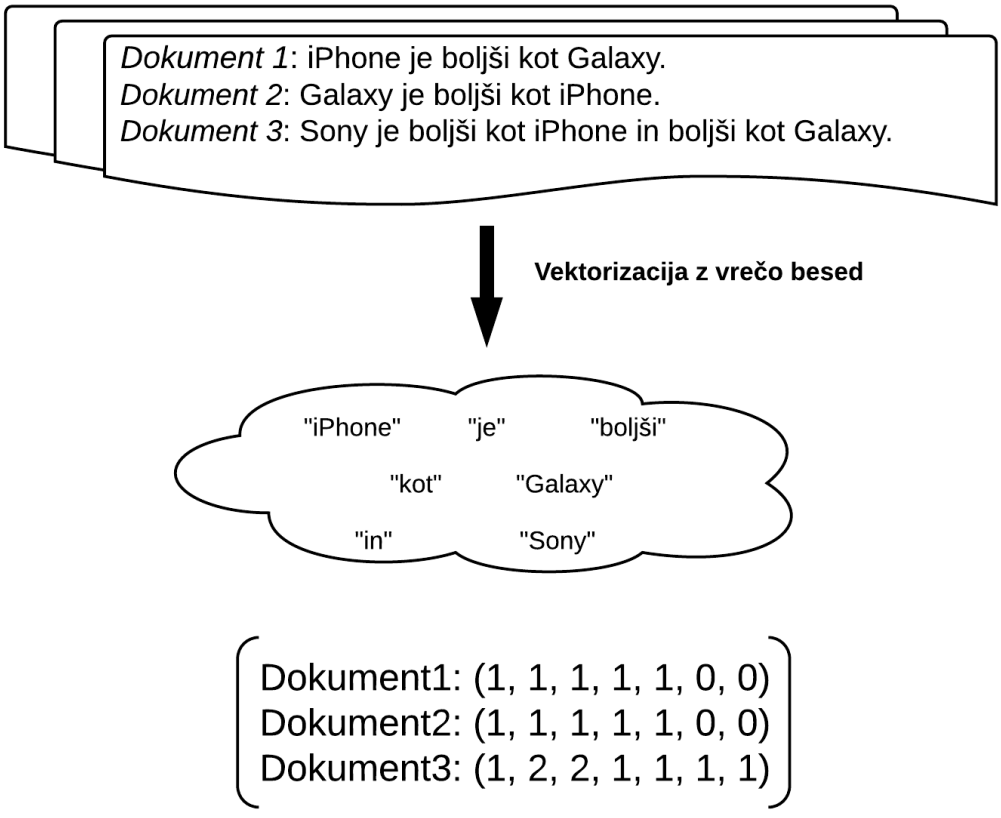
Tabela 2.3: Model vektorskega prostora z binarnimi vektorji.

Model vektorskega prostora je zelo uporabljan matematični pristop, ki je namenjen učinkovitemu obvladovanju velike množice dokumentov. S sabo prinese tudi nekaj pomankljivosti, kot so odsotnost semantičnih in sintaktičnih informacij v besedilih, predpostavlja pa tudi neodvisnost terminov. Kljub nekaterim slabostim model v praksi dobro deluje.

2.3.2 Izločanje značilk

Izločanje značilk (angl. *feature extraction*) z **vrečo besed** (angl. *bag of words*) je eden preprostejših pristopov pri klasifikaciji sentimenta. V tem modelu so značilke pojavitve besed v dokumentu. Formalno lahko napisano opredelimo z množico besed $\{w_1, ..., w_m\}$, ki se lahko pojavijo v dokumentu. $n_i(d)$ je število pojavitev w_i v dokumentu d. Dokument je predstavljen z vektorjem $\vec{d} = (n_1(d), n_2(d), ..., n_m(d))$ [17]. Velikost vektorskega prostora je v osnovi omejena s številom besed v besedilnem korpusu oziroma učni

množici. Slabost tega pristopa je, da se izgubijo nekatere informacije, npr. vrstni red besed ali struktura besedila. Na sliki 2.5 je prikazan primer, ko izguba informacij pripelje do tega, da sta dva povsem različna dokumenta enoznačno predstavljena. Z vrečo besed sta dokumenta *"iPhone je boljši kot Galaxy."* in *"Galaxy je boljši kot iPhone."* predstavljena identično. Pri klasifikaciji sentimenta ta pomankljivost ni kritična, pri bolj natančni analizi (analiza sentimenta na nivoju stavka ali posameznih vidikov) pa bi lahko izguba sintaktične in semantične informacije predstavljala težavo. Model s posameznimi besedami (unigrami) smo v našem delu uporabili kot osnovo za primerjavo z bolj naprednimi načini izločanja značilk.



Slika 2.5: Predstavitev dokumentov z vrečo besed.

Vrečo besed lahko poimenujemo tudi vrečo unigramov. Običajno pri ana-

lizi sentimenta značilk ne omejimo le na posamezne besede, ampak želimo iz učnih podatkov zajeti tudi širši kontekst, poiskati skrite relacije ipd. Besedna zveza *"rad imam"* je ponavadi dober indikator piščevega odobravanja obravnavane tematike, *"častna taborniška"* lahko predstavlja zvezo, ki se pojavlja samo pri dokumentih s pozitivnim sentimentom ipd. Za zajetje teh struktur si lahko pomagamo z **N-grami**. Večina raziskav analize sentimenta uporablja višje vrednosti N-gramov le v navezi z unigrami. Pri tem se povečini osredotočajo na največ trigrame (zaporedja treh besed). Kljub temu, da se pristop z višjimi N-grami zdi obetaven, ni nujno, da s tem izboljšamo klasifikacijski model. Pang in sod. ([17]) so pri klasifikaciji sentimenta opisa filmov prišli do tega, da so samo z unigrami dosegli boljši rezultat kot z mešanico unigramov in bigramov. Nasprotno so Go in sod. [24] z mešanico unigramov in bigramov dosegli večjo natančnost kot samo z unigrami. Klasificirali so objave na mikroblogih, konkretneje na Twitterju. Medtem, ko se je pri klasifikatorjih LR in NB z bigrami natančnost povečala, so s SVM dosegli slabši rezultat. Slednja raziskava se osredotoča na področje naše naloge (analiza sentimenta neformalnih besedil), zato smo se odločili, da pri eksperimentiranju poizkusimo tudi z višjimi N-grami.

Pri klasifikaciji sentimenta običajno zgornja dva pristopa za zajem značilk ne zadostujeta. Pogosto se uporablja naprednejše pristope, npr. zaznavanje negacije z lingvističnimi pravili. Ena od tehnik je vključevanje slovarja sentimentnih besed. Uporaba tovrstnih slovarjev je osnova pri leksikalni klasifikaciji sentimenta. Zgrajene slovarje lahko uporabimo na način, da koristne dele dodajamo kot značilke obstoječim unigramom, bigramom ipd. S slovarjem sentimenta identificiramo besede s pozitivno in negativno konotacijo, jih preštejemo ter s posebno značilko to informacijo izrazimo, npr. *"SENTILEX_POS = 10"* za besedilo, kjer smo identificirali deset besed iz slovarja sentimenta. V raziskavah, kjer kombinirajo znanje iz slovarjev z drugimi pristopi, običajno uporabijo več različnih slovarjev sentimenta. Na ta način pokrijejo več terminov. Podoben pristop smo ubrali tudi v tem delu, kjer smo želeli klasifikacijo sentimenta izboljšati z večimi slovarji orientiranosti

besed. Seveda pa opisani pristopi niso edini, ki jih lahko uporabimo pri zajemu značilk. Kot smo pisali že v sekciji o predobdelavi besedila, lahko semantično vrednost predstavlja tudi prisotnost ali število pojavitev povezav v dokumentu, pojav besed v zapisu s samimi velikimi črkami in podobno.

2.3.3 Izbira značilk

Značilk je lahko poljubno mnogo, vektorski prostor lahko meri več sto tisoč ali celo več milijonov elementov. Že sama uporaba predstavitve besedil z vrečo besed lahko prispeva k temu, da je razsežnost vektorja enaka velikosti slovarja, kar pri obširnih korpusih lahko predstavlja problem, saj za nekatere klasifikatorje postaneta učenje in klasifikacija počasna. Klasifikator LR je občutljiv na število značilk, tako da lahko proces učenja traja (pre)dolgo [17]. Naučen model se lahko tudi pretirano prilagaja učnim podatkom. S postopkom izbire značilk (angl. *feature subset selection*) želimo izbrati najbolj diskriminatorne. Cilj izbire značilk je torej v množici vhodnih značilk izbrati najbolj informativne. V nadaljevanju sledi pregled nekaterih metod za izbiro značilk.

Medsebojna informacija (angl. *mutual information*, MI) meri, koliko informacije prispeva prisotnost oziroma odsotnost termina t na pravilnost klasifikacije kategorije c. Zapisano lahko predstavimo s formulo:

$$I(U;C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U = e_t, C = e_c) log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)}, \quad (2.6)$$

kjer U predstavlja spremenljivko z vrednostmi $e_t = 1$ ali $e_t = 0$, kar pomeni, da je termin t vsebovan v dokumentu, oziroma ni vsebovan v dokumentu. Analogno temu C predstavlja spremenljivko z $e_c = 1$ ali $e_c = 0$, kar pomeni, da dokument je v kategoriji c, oziroma dokument ni v kategoriji c [27].

Če je termin dober indikator za kategorijo, doseže MI visoko vrednost. To se zgodi v primeru, ko se termin pojavi v dokumentu, če in samo če je dokument v dani kategoriji [27]. Za termine z enakimi pogojnimi verje-

tnostmi, imajo redki termini višjo vrednost kot splošni termini, tako da ocena ni primerljiva med termini z veliko razliko v pogostosti pojavitve [28].

Druga pogosto uporabljana metoda za izbiro značilk je **Hi-kvadrat** (angl. *chi-square*, χ^2). S statističnim testom χ^2 testiramo neodvisnost dveh dogodkov. Dogodka A in B sta neodvisna, če P(AB) = P(A)(B) oziroma P(A|B) = P(A) in P(B|A) = P(B). Pri izbiri značilk je dogodek A pojavitev termina in dogodek B pojavitev kategorije. Enačbo lahko zapišemo kot:

$$\chi^2(D,t,c) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}, \tag{2.7}$$

kjer je N opazovana pogostost v množici dokumentov D, E pa pričakovana pogostost. Z χ^2 merimo odstopanje E od N. Visoka vrednost χ^2 kaže na to, da je hipoteza o neodvisnosti napačna. V primeru, da sta dogodka odvisna, pojavitev termina bolj verjetno (in obratno) pripomore k pojavitvi kategorije [27]. Za razliko od MI je vrednost χ^2 normalizirana za termine iste kategorije. Ker slednje ne velja za termine z nizko pogostostjo pojavitve, zanesljivost mere χ^2 pade v primeru terminov z majhno pogostostjo [28].

Izbira značilk na podlagi **pogostosti pojavitve** (angl. *frequency-based feature selection*) temelji na izbiri tistih terminov, ki so za kategorijo najbolj pogosti. Pri tem lahko merimo bodisi število dokumentov v kategoriji c, ki vsebujejo termin t ali število pojavitev termina t, ki se pojavijo v dokumentih v kategoriji c. V primeru izbire več tisoč značilk se ta preprosta metoda običajno dobro obnese in jo lahko uporabimo kot alternativo prvima dvema. Kjer je značilk malo, lahko pride do tega, da prevladajo termini, ki ne nosijo posebnih informacij o kategoriji, se pa pojavljajo v več dokumentih. [27].

2.3.4 Uteževanje značilk

Pri opisu vektorskega prostora smo navedli, da enostavnejši modeli operirajo z binarnimi vektorji. Ker termini v dokumentih običajno niso enako pomembni, dvojiška vrednost ne zadostuje. Ne zanima nas samo prisotnost oziroma odsotnost značilke v dokumentu, ampak tudi njena teža za doku-

ment. Z uteževanjem značilk (angl. *feature weighting*) želimo posameznim značilkam nastaviti pomembnost z ozirom na njihovo zmožnost ločevanja oznak razredov. Tako z izbiro kot tudi uteževanjem značilk želimo izboljšati natančnost klasifikatorja. V nasprotju z izbiro značilk, kjer značilke pustimo nedotaknjene, pri uteževanju z utežmi spremenimo njihove vrednosti.

Učni podatki vsebujejo besede kot so *je*, *iz*, *mogoče*, ki se pojavljajo pogosto, a hkrati je njihov prispevek na klasifikacijo vprašljiv. Metodo blokiranja pogostih besed smo že spoznali. Z uteževanjem bi radi dosegli bodisi večjo ali manjšo pomembnost posameznega termina za dokument v korpusu. V raziskavah klasifikacije sentimenta se za uteževanje terminov pogosto uporablja metrika **tf-idf**. Za termin t in dokument d, mero tf-idf formalno izrazimo z:

$$w_{t,d} = tf_{t,d} \times log\frac{N}{df_t}. \tag{2.8}$$

Iz zgornje formule vidimo, da je utež tf-idf sestavljena iz dveh delov in sicer produkta pomembnosti termina znotraj dokumenta ($tf_{t,d}$) ter pomembnosti termina v korpusu. Mero tf izrazimo s številom pojavitev termina t v dokumentu deljeno s številom vseh terminov v dokumentu. *idf* predstavlja število vseh dokumentov v korpusu deljeno z dokumenti, ki vsebujejo termin t. Termin je pomemben, če pogosto nastopa v manjšem številu dokumentov, če je pogosto prisoten v večini dokumentov v korpusu, ga težko označimo za pomembno značilko. Pogosti termini bodo imeli nizko vrednost tf-idf, redkeje zastopani pa višjo.

Tako *tf* kot *tf-idf* sta pogosto vključeni meri v raziskave analize sentimenta. Enoznačnega odgovora na to kaj je boljše ni. Smailović [13] je pri klasifikaciji objav iz mikrobloga Twitter ugotovila, da *tf* deluje precej boljše kot kompleksnejša *tf-idf*. V določenih raziskavah se bolje od uteževanja obnese preprosto binarno izbiranje značilk, kjer dvojiška vrednost 1 pomeni, da je termin prisoten, 0 pomeni odsotnost termina. Agarwal in Mittal [29] pojasnita, zakaj je lahko pri klasifikaciji sentimenta takšen odklon od denimo kategorizacije tem, kjer naprednejše sheme uteži dobro delujejo. Pisci uporabljajo različne sentimentne besede za izražanje sentimenta. Če nekdo izraža

mnenje o vozilu, bi lahko zapisal *"Avto je hiter, menjalnik natančen ter volanski mehanizem odziven."*. V tem stavku pisec sentiment izraža s tremi različnimi sentimentnimi besedami (*"hiter"*, *"natančen"*, *"odziven"*). Malo verjetno je, da bi za vse tri vidike uporabil isto sentimentno besedo. Na primeru vidimo, da je lahko prisotnost oziroma odsotnost termina pomembnejša od pogostosti pojavitve.

Poglavje 3

Uporabljena orodja in tehnologije

Za namen diplomske naloge smo razvili spletno aplikacijo ter spletno storitev (angl. *web service*). V nadaljevanju poglavja so opisana najpomembnejša orodja in tehnologije, ki smo jih pri tem uporabili.

3.1 Programski jeziki

Pri izdelavi praktičnega dela smo se trudili, da število uporabljenih programskih jezikov omejimo, zaradi lažje obvladljivosti in s tem tudi enostavnejšega pristopa nekoga, ki bi želel aplikacijo v prihodnosti razširiti. Veliko večino nalog (obdelava besedila, klasifikacija, programiranje na strani strežnika ipd.) smo opravili s pomočjo programskega jezika Python. V okviru uporabniškega vmesnika spletne aplikacije, smo si dodatno pomagali s programskim jezikom JavaScript.

3.1.1 Python

Python je široko uporabljan visokonivojski programski jezik. Podpira več programerskih pristopov, od proceduralnega, funkcijskega, do objektno orientiranega. CPython - referenčna implementacija Pythona, je odprtokodna

in prosto dostopna. Tolmači za Python so na voljo za številne operacijske sisteme, tako da je poganjanje Python kode mogoče na različnih sistemih. Z uporabo orodij kot so Py2exe ali Pyinstaller je mogoče Python kodo zapakirati v samostojne izvršljive programe [30].

Python je zelo razširjen v akademski sferi. Po raziskavi ([31]) ga že osem od desetih najboljših ameriških univerz uporablja kot programski jezik s katerim študente vpeljejo v svet programiranja. Berljivost kode, preprostost, razširljivost in delovanje v različnih sistemih je le nekaj razlogov zakaj je Python tako popularen za poučevanje [32]. Raziskovalci bodo cenili velik nabor dostopnih knjižnic. Naj omenimo samo numerično matematični knjižnici NumPy (Numeric Python) in SciPy (Scientific Python). Zaradi velikega nabora orodij za procesiranje besedila je izredno priljubljen tudi pri opravilih, ki se tičejo obdelave naravnega jezika.

3.1.2 JavaScript

JavaScript je visokonivojski, dinamičen, netipiziran, interpretiran programski jezik [33]. Postal je de-facto standard za razvoj spletnih rešitev. Iz tega razloga je podpora za JavaScript vgrajena v vse glavne spletne brskalnike. Brez JavaScripta si dinamičnih spletnih vsebin ni mogoče predstavljati. JavaScript se, poleg spleta, lahko uporablja tudi v drugih okoljih.

Koda JavaScript se izvaja na strani odjemalca, v spletnem brskalniku uporabnika. Z JavaScriptom lahko torej sprogramiramo, kako se spletna stran odziva na različne dogodke, kot je denimo klik miške na enega od elementov spletne strani. Z JavaScriptom lahko posledično kontroliramo obnašanje spletne strani [34].

3.2 Knjižnice

Sam programski jezik ali platforma kot lupina, bi bil brez dostopnih knjižnic precej neuporaben. Kot smo že napisali, je bil pri izbiri programskega jezika pomemben atribut dostopnost knjižnic, ki nam olajšajo zadane naloge. V

našem primeru so to knjižnice za obdelavo jezika, za strojno učenje ipd. Pomembnejše uporabljene knjižnice so predstavljene v nadaljevanju poglavja.

3.2.1 NLTK

NLTK [35] je prosto dostopna knjižnica za obdelavo naravnega jezika v programskem jeziku Python. Začetki segajo v leto 2001, ko so na Oddelku za računalništvo in informatiko Univerze v Pensilvaniji razvili prvo verzijo. Od tedaj se knjižnica neprestano razvija in razširjuje. Knjižnico pri poučevanju uporabljajo številne univerze po svetu, prav tako pa je postala izredno priljubljena tudi med raziskovalci, ki se ukvarjajo z različnimi aspekti obdelave naravnega jezika [20].

Pri obdelavi naravnega jezika se srečujemo z različnimi opravili. Knjižnica NLTK takšna opravila deli v zaokrožene celote. Vsaka takšna zaokrožena celota je predstavljena z modulom. Tako imamo modul za dostop do korpusov in leksikonov, modul za klasifikacijo, modul za evalvacijo ipd. NLTK ponuja širok nabor funkcionalnosti in malo je opravil, za katera bi bilo potrebno poseči po drugih knjižnicah. Morda največja pomankljivost v tem oziru je odsotnost nekaterih priljubljenih metod strojnega učenja, denimo SVM.

NLTK je odlično dokumentirana knjižnica. Na voljo je prosto dostopna knjiga [20] z razlago in primeri nekaterih najpogostejših NLP opravil. Na spletni strani projekta [35] je na voljo popoln pregled vseh modulov, razredov in funkcij, vključno s primeri uporabe.

Za programski jezik Python je na voljo še vrsta drugih tovrstnih knjižnic. Matej Martinc se je v svojem diplomskem delu osredotočil na primerjavo le-teh. Knjižnico NLTK je priporočil vsem, ki se želijo bolje spoznati s procesiranjem naravnega jezika ali želijo prilagodljivo knjižnico, z veliko funkcionalnostmi in fleksibilnostjo. Poleg tega je bila to edina knjižnica med primerjanimi, ki nudi vsaj delno podporo obdelavi slovenskega jezika. Kar se slabosti tiče je omenil predvsem performančne težave ter malce kompleksnejšo rabo [12]. Na podlagi napisanega ocenjujemo, da je bila odločitev za uporabo knjižnice NLTK v tem diplomskem delu smotrna.

3.2.2 scikit-learn

scikit-learn [36] je odprtokodna knjižnica za programski jezik Python, ki vsebuje vrsto orodij za podatkovno analitiko. V knjižnici najdemo širok nabor algoritmov za klasifikacijo, regresijo in gručenje, kot so SVM, naključni gozdovi ipd. Knjižnica je dobro dokumentirana. Na voljo so številni praktični primeri uporabe, tudi iz domene obdelave naravnega jezika.

Preko ovijalnih razredov (angl. *wrapper classes*) je mogoča izraba scikit-learn klasifikatorjev znotraj knjižnice NLTK. Tako lahko funkcionalnost slednje dodobra razširimo z nekaterimi popularnimi metodami pri obdelavi naravnega jezika, denimo SVM. V kodi 3.1 je takšen princip tudi predstavljen.

```
1 from sklearn.svm import LinearSVC
2 from nltk.classify.scikitlearn import SklearnClassifier
3 classif = SklearnClassifier(LinearSVC())
```

Koda 3.1: Inicializacija NLTK klasifikatorja z algoritmom SVM iz knjižnice scikit-learn.

3.2.3 LemmaGen

Za pretvorbo besed v njihovo osnovno (slovarsko) obliko smo uporabili knjižnico LemmaGen. LemmaGen [37] je odprtokodna platforma za lematizacijo. Začetni cilj projekta je bila izdelava kvalitetnega lematizatorja za slovenski jezik. Trenutno lematizator podpira še 11 drugih evropskih jezikov. Za nove jezike se je sistem sposoben naučiti lematizacijskih pravil s podajanjem obstoječih primerov parov beseda-lema.

Lematizator je na voljo za številne programske jezike, od leta 2013 tudi za programski jezik Python. Inštalacija je na voljo preko standardnega Python Pypi repozitorija [38]. Raba knjižnice je predstavljena v kodi 3.2.

```
from lemmagen.lemmatizer import Lemmatizer

lematizator = Lemmatizer(dictionary =
                  lemmagen.DICTIONARY_SLOVENE)
print(lematizator.lemmatize("delamo"))
```

Koda 3.2: Lematizacija besede *delamo* z lematizatorjem LemmaGen v programskem jeziku Python.

3.2.4 Beautiful Soup

Beautiful Soup [39] je knjižnica, ki omogoča razčlenjevanje (angl. *parsing*) dokumentov HTML in XML. Z izgradnjo razčlenitvenega drevesa lahko izvlečemo (angl. *extract*) podatke s spletnih strani, kar je uporabno za luščenje podatkov s spleta (angl. *web scraping*). Z uporabo te knjižnice smo denimo izluščili komentarje uporabnikov s spletnih strani.

Knjižnica omogoča preprosto navigacijo, iskanje in spreminjanje razčlenitvenega drevesa. V kodi 3.3 je prikazano iskanje po drevesu - iskanje HTML elementa *div*, ki pripada razredu *comment_paginator*.

Beautiful Soup samodejno pretvori vhodne dokumente v Unicode in izhodne dokumente v UTF-8. Tako se programerju ni potrebno ubadati s formatom besedila, razen v primeru, da ga dokument ne specificira in ga knjižnica ne ugotovi samodejno [39].

```
import requests
from bs4 import BeautifulSoup

url = "http://www.rtvslo.si/sport/kosarka/..."
bs_drevo = BeautifulSoup(requests.get(url).text)
bs_divelement = bs_drevo.find("div",
                    {"class": "comment_paginator"})
if bs_divelement != None:
  ...
```

Koda 3.3: Iskanje po dokumentu HTML z uporabo knjižnice Beautiful Soup.

3.2.5 Google API Client Library for Python

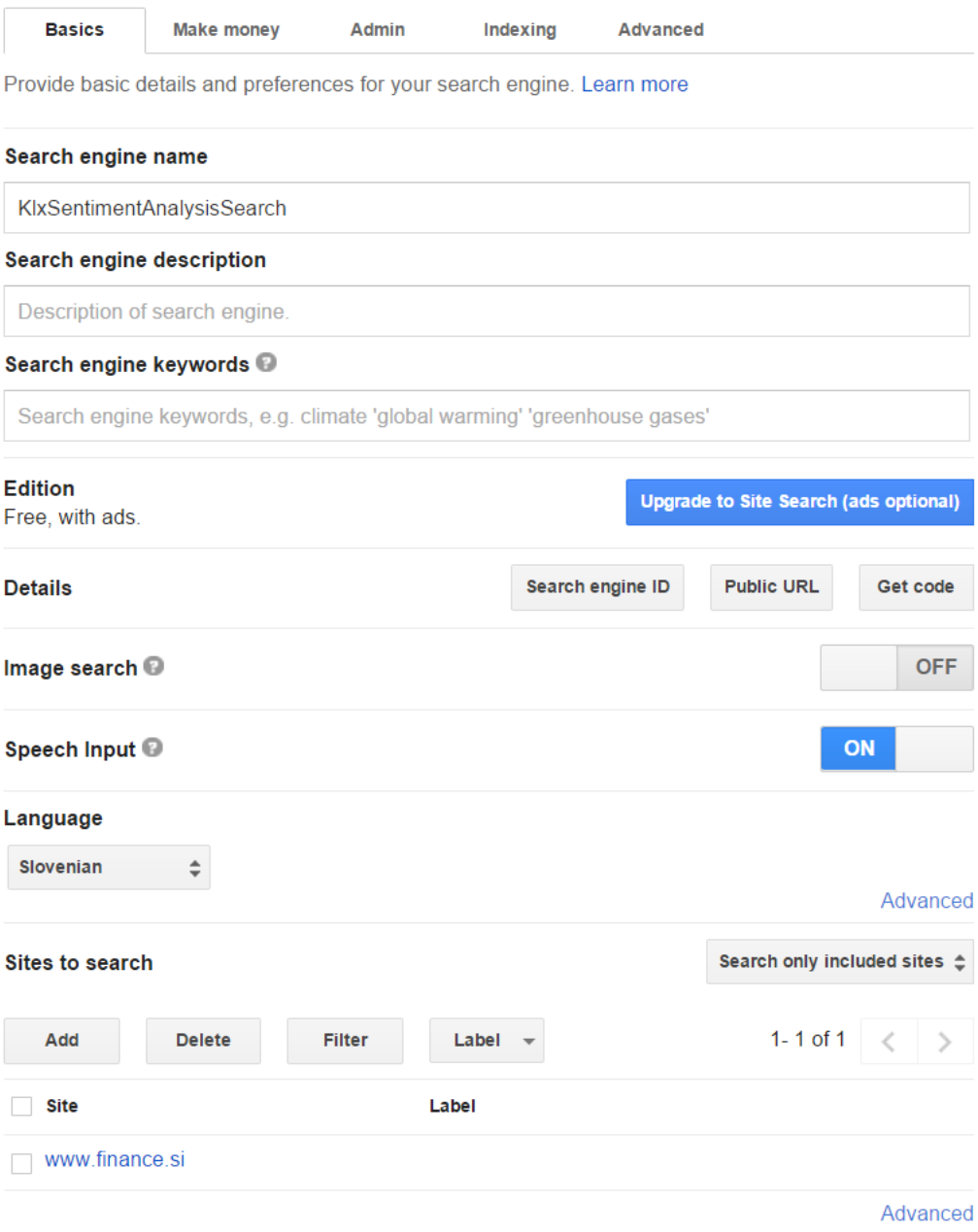
Google API Client Library for Python [40] je knjižnica za programski jezik Python, ki omogoča dostop do spletnih storitev, ki jih nudi podjetje Google.

V našem delu smo uporabili API za iskanje spletnih strani. V kodi 3.4 je prikazana raba *CustomSearch API*, kjer iščemo izraz *"gospodarska kriza 2008"* po spletnem mestu *rtvslo.si*. Na voljo imamo tudi razširjeno iskanje, s katerim preko metapodatkov vplivamo na rezultate, denimo določitev datumskega intervala.

```
1  from googleapiclient.discovery import googledisc
2
3  iskalnik = googledisc("customsearch",
4                        "v1",
5                        developerKey="API KEY..")
6  return iskalnik.cse().list(q="gospodarska kriza 2008",
7                             cx="ENGINE KEY...",
8                             lr='lang_sl',
9                             siteSearch="rtvslo.si",
10                            start=0).execute()
```

Koda 3.4: Iskanje po spletu z uporabo knjižnice Google API Client.

Izraba storitve iskanja po spletnih straneh ni povsem trivialna. Poleg prijave na storitev, je potrebno nujno določiti še iskalnik. Z iskalnikom določimo spletna mesta, po katerih iskalnik išče. Tako nam iskalnik predstavlja neke vrste storitev iskanja po meri, saj lahko s parametri iskanje precej prilagodimo. Na sliki 3.1 vidimo konfiguracijo takšnega iskalnika. Iskalnik je nastavljen tako, da išče izključno po spletnem mestu *finance.si*.



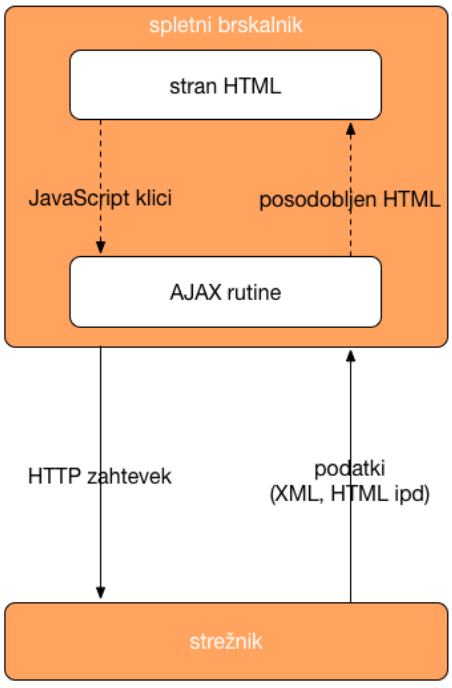
Slika 3.1: Konfiguracija iskalnika Google Custom Search.

3.3 Spletne tehnologije

V tem podpoglavju predstavimo tehnologije, ki smo jih rabili pri izdelavi spletne aplikacije. Spletno aplikacijo smo želeli narediti kar se da enostavno za uporabo. S tehnikami, kot je denimo AJAX, smo poskrbeli tudi za optimiranje prenosa podatkov med strežnikom in odjemalci.

3.3.1 AJAX

AJAX [41, 42] je tehnika, koncept za izdelavo boljših, hitrejših in interaktivnejših spletnih strani. Ajax je skripta na odjemalcu, ki komunicira s strežnikom/bazo asinhrono (izvajanje v ozadju), brez potrebe po ponovnem nalaganju celotne strani. S tem posredno vplivamo na drastično zmanjšanje količine prenešenih podatkov na relaciji odjemalec - strežnik.



Slika 3.2: Model spletne aplikacije z uporabo Ajaxa.

Kljub očitnim prednostim pa lahko izpostavimo tudi nekaj slabosti. Uporabniki, katerih spletni brskalniki ne podpirajo JavaScripta ali pa je slednji izklopljen, ne morejo pravilno uporabljati spletnih strani, ki temeljijo na Ajaxu. Asinhrono programiranje s povratnimi klici (angl. *callbacks*) hitro privede do kompleksne kode, ki jo je težko vzdrževati in testirati.

3.3.2 Bootstrap

Bootstrap [43] je prosto dostopen in odprtokoden skupek orodij za izdelavo spletnih strani in spletnih aplikacij. Vsebuje na HTML in CSS osnovane predloge za tipografijo, gumbe, navigacijo in druge elemente uporabniškega vmesnika, kot tudi JavaScript razširitve.

Podprt je širok nabor spletnih brskalnikov. Z verzijo 2.0 podpira odziven spletni dizajn (angl. *responsive web design*), kar pomeni, da se postavitev spletne strani dinamično prilagaja karakteristikam naprave (tablica, prenosni telefon, namizni računalnik ipd.).

3.3.3 Django

Django je ogrodje za gradnjo spletnih aplikacij. Je prosto dostopen in temelji na programskem jeziku Python. Ponuja razvojno paradigmo temelječo na načrtovalskem vzorcu model-pogled-krmilnik (angl. *model-view-controller*, MVC). Z ogrodjem lahko izdelamo velike in kompleksne spletne aplikacije. Django že v osnovi ponuja več modulov, ki so nam pri razvoju aplikacije v veliko pomoč. Med njimi je tudi varnostni podsistem, ki omogoča rokovanje z avtentikacijo in avtorizacijo uporabnikov ter delo s skupinami uporabnikov. Za razliko od nekaterih drugih tovrstnih ogrodij, denimo Microsoftove alternative ASP.NET MVC, je vključen še poseben administratorski modul, ki nad definiranimi modeli omogoča operacije kreiranja, pregledovanja, urejanja in brisanja zapisov, brez potrebe po izdelavi namenskih pogledov.

3.4 Storitve v oblaku - Microsoft Azure

O računalništvu v oblaku [44] govorimo takrat, ko so dinamično razširljivi in pogosto virtualizirani računalniški viri (angl. *resources*) na voljo kot storitev preko interneta.

Oblak prinaša nekaj privlačnih prednosti [45]. Kot glavne lahko naštejemo:

- Samooskrbovanje z viri: sistem omogoča, da končni uporabniki postavijo in zaganjajo aplikacije in storitve brez posredovanja ponudnika storitev. Uporabniki z računalniškimi viri upravljajo samostojno. Fizična interakcija z infrastrukturo ni potrebna.

- Elastičnost: podjetja lahko skalirajo računske vire in kapacitete glede na trenutne potrebe. Ko potrebe narastejo, preprosto skalirajo navzgor in obratno, ko se potrebe zmanjšajo. V primeru klasičnega računalništva bi tako uporabniki morali fizično nadgraditi infrastrukturo, da bi ta sledila povečanim potrebam.

- Plačaj po porabi: viri so merjeni na granularnem nivoju, s čimer je uporabnikom omogočeno, da plačajo dejansko porabo po virih in delovnih obremenitvah (angl. *workloads*).

Microsoft Azure je Microsoftova platforma in infrastruktura računalništva v oblaku.

Odločitev za uporabo (Microsoftove platforme) računalništva v oblaku je bila sprejeta iz naslednjih razlogov:

- podpora programskemu jeziku Python,

- enostavno skaliranje virov, v kolikor bi potrebovali povečane zmogljivosti. Tu ciljamo predvsem na razširitev aplikacije, kjer bi analitičnost prišla do izraza in s tem tudi večja potreba po računski moči. Denimo spremljanje socialnih medijev, novičarskih portalov in uporabnikov v realnem času;

- odlična integracija z Visual Studio IDE. Z le nekaj kliki lahko posodobimo spletno stran, spletne in podporne storitve.

3.4.1 SQL Database

SQL Database [47] je ena od storitev v oblaku Microsoft Azure, ki omogoča shranjevanje podatkov. SQL Database omogoča uporabnikom, da izvajajo relacijska povpraševanja na shranjenih podatkih, ki so lahko strukturirani, pol strukturirani ali nestrukturirani dokumenti. SQL Database kot osnovo uporablja posebno verzijo strežnika Microsoft SQL. Strežnik Microsoft SQL je sistem za upravljanje relacijskih podatkovnih baz. Prenos podatkov med strežnikom in odjemalci se vrši v formatu osnovanem na XML.

3.4.2 Cloud Services

Azure Cloud Services [46] zagotavljajo platformo kot storitev, kar je zaradi velike skalabilnosti idealno za računsko intenzivne aplikacije in storitve. Število instanc, tj. virtualnih strežnikov v infrastrukturi, lahko poljubno povečamo.

Na voljo sta dve vrsti vlog: spletna vlogo (angl. *web role*) in vloga delavca (angl. *worker role*). Glavna razlika je v tem, da spletna vloga teče na strežniku Windows z nameščenim spletnim strežnikom IIS, vloga delavca pa teče na strežniku brez IIS. V našem delu smo uporabili obe, spletno vlogo za spletno storitev, vlogo delavca za podporne storitve.

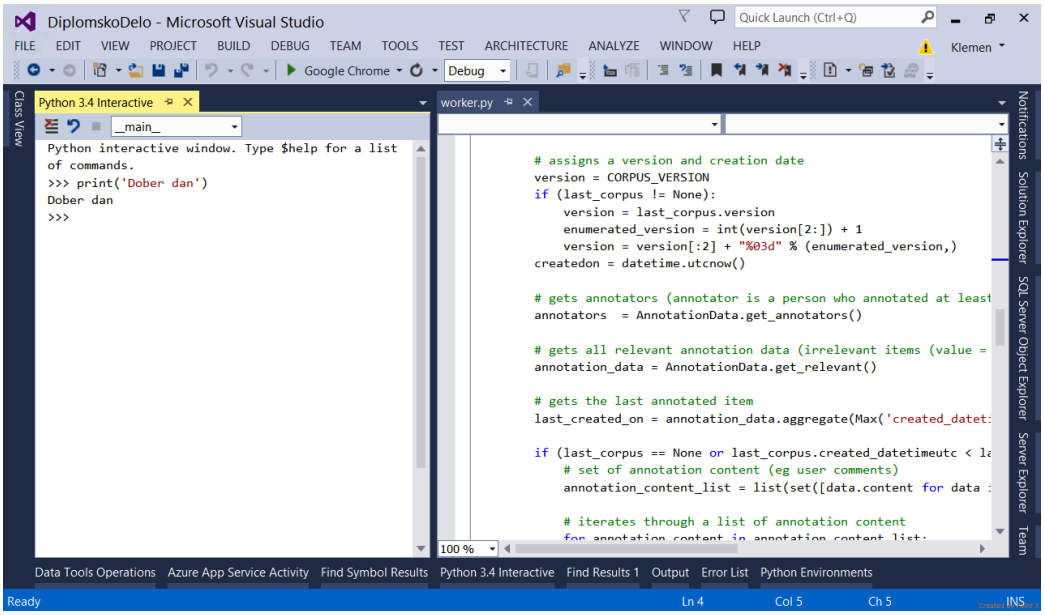
3.5 Ostalo

V tem podpoglavju naštejemo orodja, ki smo jih uporabili za urejanje izvorne kode, pisanje SQL povpraševanj ipd.

3.5.1 Microsoft Visual Studio

Visual Studio je zaokrožena celota razvojnih orodij za razvoj celotnega spektra izdelkov, od spletnih aplikacij, spletnih storitev XML, namiznih aplikacij do mobilnih aplikacij. Podpira več programskih jezikov. Vsi uporabljajo isto integrirano razvojno okolje [48].

Privzeto Visual Studio nima podpore za programski jezik Python. Podporo dobi šele z namestitvijo vtičnika Python Tools for Visual Studio. S tem vtičnikom lahko Visual Studio preobrazimo v pravi Python IDE. Podprto je tako razhroščevanje, kot ponujanje ukazov (angl. *intellisense*) [49]. Preko interaktivnih oken je možna tudi uporaba ukaznega tolmača Python, kar je prikazano tudi na sliki 3.3.



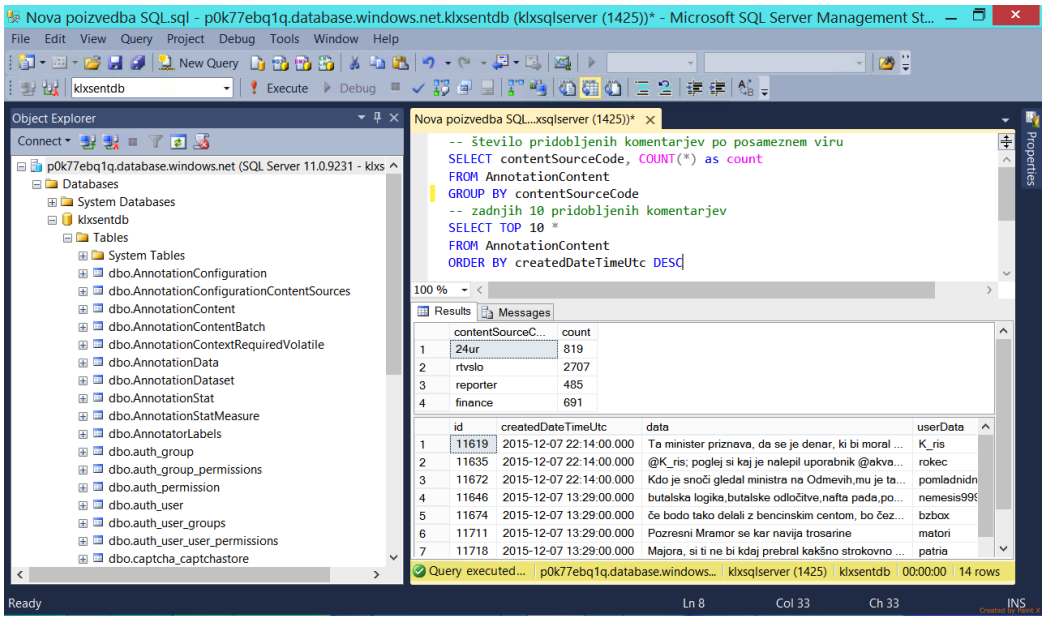
Slika 3.3: Podpora programskemu jeziku Python v IDE Visual Studio.

3.5.2 Microsoft SQL Server Management Studio

SQL Server Management Studio (v nadaljevanju SSMS) je integrirano okolje za administracijo, dostop, konfiguracijo in urejanje strežnika SQL. SSMS

združuje širok nabor orodij in pripomočkov za dostop do strežnika SQL. Namenjen je tako razvijalcem kot administratorjem [50].

Iz aspekta razvijalca so tipične naloge, pri katerih si pomagamo s SSMS, pisanje in poganjanje poizvedb SQL, pisanje in razhroščevanje najrazličnejših skript SQL ter performančno profiliranje. Velika prednost orodja je prikaz vseh objektov (tabele, shranjene procedure, uporabniki, skupine uporabnikov, dostopi, ipd.) strežnika SQL v obliki drevesne strukture na enem mestu, kar je vidno tudi na sliki 3.4.

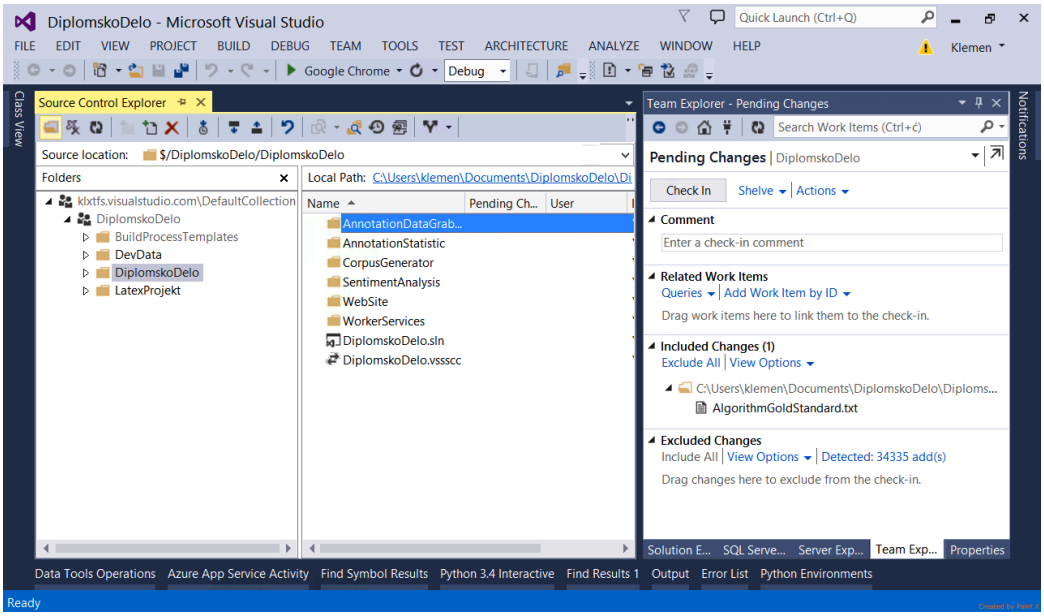


Slika 3.4: Okolje SSMS s prikazom objektov strežnika SQL v obliki drevesne strukture na levi ter izvedenima poizvedbama SQL na desni strani.

3.5.3 Microsoft Team Foundation Version Control

Microsoft Team Foundation Version Control (v nadaljevanju TFVC) je centraliziran sistem za nadzor verzij. TFVC omogoča shranjevanje vseh vrst datotek. Za ta sistem nadzora verzij smo se odločili zaradi odlične integracije z Visual Studio IDE, kar je lepo razvidno tudi iz slike 3.5. Podpira dva

načina delovanja in sicer strežniško ter lokalno delovanje. Pri prvem načinu je razvijalcem omogočena izstavitev datoteke. Pri izstavitvi se datoteka samodejno zaklene in jo drugi uporabniki, v času izstavitve, ne morejo urejati. Težava tega načina je ta, da so neizstavljene datoteke na disku razvijalca označene samo za branje (angl. *read-only*). V kolikor strežnik postane nedosegljiv mora razvijalec preklopiti v način brez povezave (angl. *go offline*). Z drugim načinom delovanja lahko te težave obidemo. V lokalnem načinu datoteke niso označene samo za branje, prav tako ni potrebe po izstavitvi le teh pred začetkom urejanja. Morebitni konflikti se rešujejo šele pri shranjevanju sprememb v repozitorij [51]. Težava tega načina je, da ni nobene kontrole nad tem, katere datoteke se trenutno urejajo in lahko pride do večjih konfliktov, ko je potrebno spremembe shraniti na repozitorij.



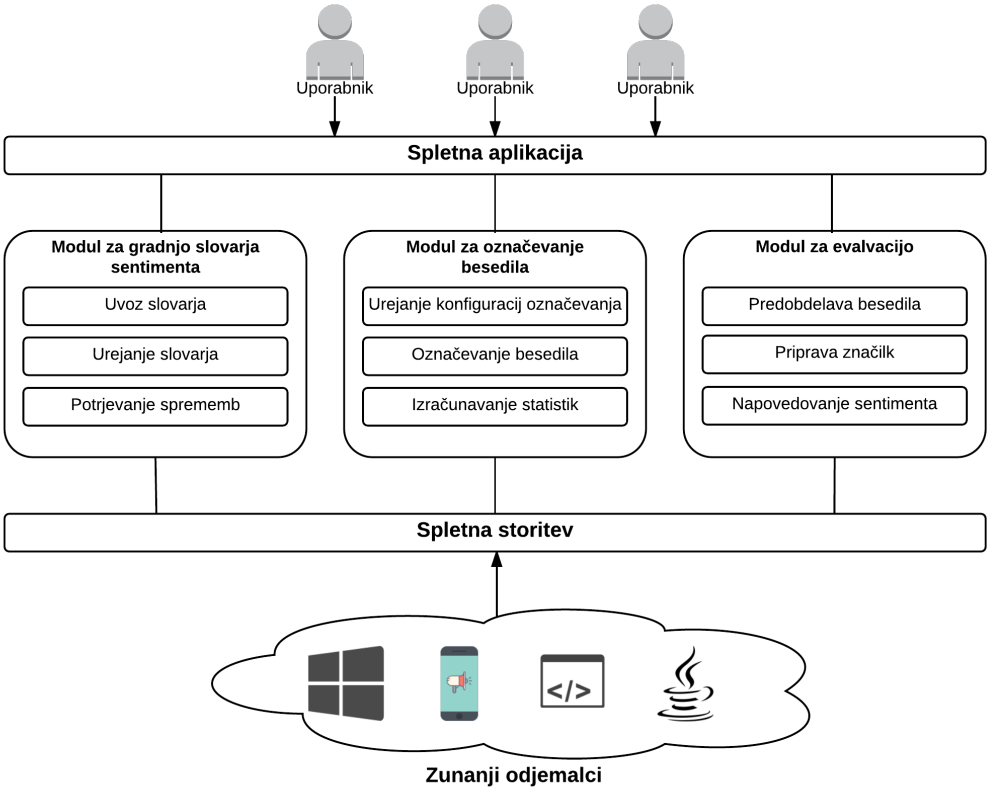
Slika 3.5: Integracija TFVC z razvojnim okoljem Microsoft Visual Studio. Na levi strani je prikazana vsebina repozitorija, na desni seznam izstavljenih datotek.

Poglavje 4

Izdelava orodja za klasifikacijo uporabniških komentarjev

Poglavje opisuje izgradnjo orodja za klasifikacijo uporabniških komentarjev v slovenskem jeziku. Predstavljeni bodo pristopi pri gradnji slovarjev sentimenta ter kreiranje korpusa učnih podatkov za gradnjo klasifikatorjev. V poglavju se posvetimo tudi tehničnemu opisu delov orodja, kot je denimo spletna aplikacija. Rezultati razvite rešitve bodo predstavljeni v poglavju 5.

Pri snovanju smo želeli izdelati celovito orodje, ki bo sposobno tudi komunikacije z zunanjim svetom. Komunikaciji z zunanjimi odjemalci je namenjena spletna storitev, preko katere dostopajo do funkcionalnosti sistema. Orodje sestavlja več funkcionalnih sklopov. Posameznemu sklopu lahko rečemo tudi modul. Shema rešitve je prikazana na sliki 4.1. V nadaljevanju so sklopi podrobno predstavljeni.



Slika 4.1: Shema orodja za klasifikacijo.

V okviru praktičnega dela je bilo razvito naslednje:

- Podatkovna baza: za shranjevanje podatkov smo uporabili relacijsko podatkovno bazo. Izogibali smo se pretirani kompleksnosti podatkovnega modela, tako da pri migraciji na katerega od drugih ponudnikov DBMS ne bi smelo biti večjih težav. Podatkovni model bo podrobneje predstavljen v opisu posameznih sklopov sistema.

- Spletni uporabniški vmesnik: interakcija uporabnikov in sistema temelji na spletni aplikaciji. Za spletno aplikacijo smo se odločili zaradi dostopnosti in neodvisnosti od uporabnikovega operacijskega sistema. Spletne tehnologije so v zadnjih letih napredovale v tolikšni meri, da so se meje med njimi in namiznimi aplikacijami skorajda zabrisale.

- Spletna storitev: poleg uporabniškega vmesnika spletne aplikacije so funkcionalnosti sistema dostopne tudi preko spletne storitve. Slednja ponuja nabor funkcij, ki jih kliče zunanji odjemalec (zunanja aplikacija). Preko spletne storitve je mogoča integracija s sistemom. V nadaljevanju bodo operacije spletne storitve podrobneje opisane.
- Podporne storitve: nekatere operacije se izvajajo v ozadju (denimo izračunavanje statistik pri označevanju besedila). Takšne scenarije rešujemo z uporabo podpornih storitev, ki se izvajajo na strežniku in prožijo ob določenem časovnem intervalu. Običajno z njimi rešujemo primere, ko bi bilo takojšnje procesiranje pri uporabniški interakciji prezahtevno oziroma želimo podatke pripraviti vnaprej; npr. vsakokratna izgradnja korpusa bi bila potratna rešitev, zato ga zgradimo preko podporne storitve, ki korpus dnevno posodablja.

Zgoraj našteto smo implementirali na oblačni platformi Microsoft Azure.

4.1 Spletni vmesnik

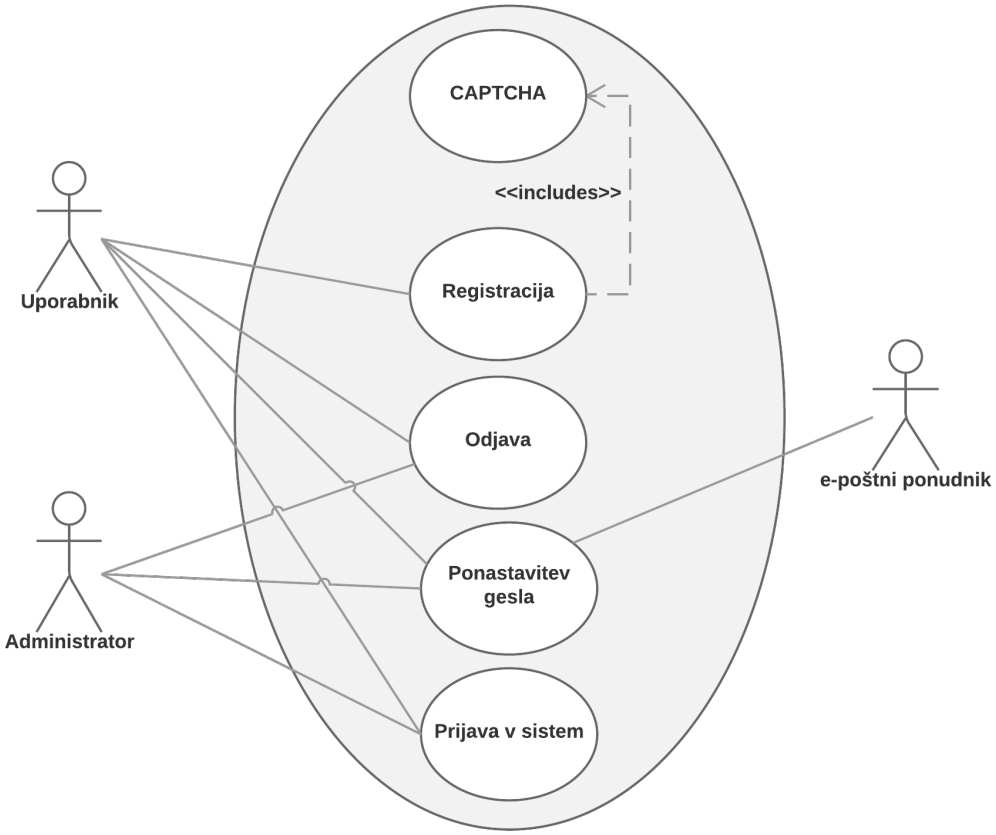
Uporabniki do sistema dostopajo preko spletnega vmesnika (slika 4.2). Aplikacija temelji na ogrodju Django, ki smo ga spoznali v poglavju 3.3.3. Za zagotovitev celostne grafične podobe in prilagodljivost spletnega vmesnika smo uporabili orodja Bootstrap. Običajno je razvoj kompleksnejše spletne aplikacije zahtevnejši in dolgotrajnejši od razvoja alternativne namizne aplikacije, vendar prednosti spletnega razvoja pogosto odtehtajo dodatne vložke. Med prednosti lahko uvrstimo dostopnost, neodvisnost od operacijskega sistema uporabnika ter lažje obvladovanje aplikacije v celotnem življenjskem ciklu. Pri namiznih aplikacijah je potrebno napisati dodaten inštalacijski program, ki namesti aplikacijo pri uporabniku. Prav tako se lahko pojavijo problemi, ko je potrebno takšno aplikacijo nadgraditi. Pri spletnih aplikacijah je namestitev enostavnejša, saj vsa opravila potekajo na spletnih strežnikih in povsem neodvisno od računalnika uporabnika, prav tako je enostavnejše tudi posodabljanje.

Slika 4.2: Vstopna stran spletne aplikacije.

4.1.1 Avtentikacija in avtorizacija

Dostop do virov sistema je zaščiten, zato je potrebna prijava uporabnikov (na sliki 4.3). Za avtentikacijo in avtorizacijo smo prilagodili vgrajen varnostni podsistem iz ogrodja Django. Sistem pozna dve vlogi, in sicer *Administrator* ter *Uporabnik*. Slednja vloga je privzeta ob registraciji novega uporabnika. Glede na vlogo uporabnik sistema dostopa do funkcionalnosti, ki so mu na voljo. Po odjavi je uporabnik preusmerjen na vstopno stran. Omogočili smo ponastavitev gesla, v kolikor ga uporabnik pozabi. Za ta namen se pošlje pošta na elektronski naslov, ki ga je uporabnik navedel ob registraciji. Za pošiljanje elektronske pošte nismo postavljali lastnega poštnega strežnika,

ampak smo uporabili storitev pošiljanja elektronske pošte preko zunanjega ponudnika[1]. Registracija je zaščitena z metodo za preverjanje prisotnosti človeškega faktorja (CAPTCHA), v našem primeru je to slika s slabo berljivim besedilom, ki jo mora uporabnik prepisati v za to namenjeno tekstovno polje, ko se želi registrirati.



Slika 4.3: Diagram primerov uporabe pri prijavi in registraciji.

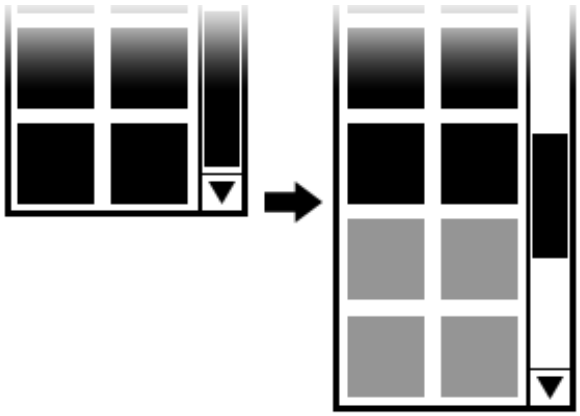
[1]Za ponudnika pošiljanja elektronske pošte smo izbrali SendGrid (`http://sendgrid.com`). Pošiljanje do nekaj tisoč elektronskih sporočil na mesec je brezplačno.

4.1.2 Gradniki uporabniškega vmesnika

Pri razvoju spletne aplikacije smo kot eno zahtevo postavili enovitost rešitve, v smislu minimizacije števila različnih gradnikov, ki se pojavijo v okviru različnih delov sistema. Z gradniki imamo v mislih elemente uporabniškega vmesnika: gumbe, vnosna polja ter izbirne sezname. V nadaljevanju sledi predstavitev nekaterih naprednejših konceptov, ki smo jih uporabili pri načrtovanju uporabniškega vmesnika.

Posamezna entiteta ima lahko mnogo elementov. Za primerjavo, slovar sentimentnih besed ima več tisoč vnosov. Običajno uporabniku množico entitet predstavimo v obliki seznamov. Zaradi velikega števila elementov lahko hitro naletimo na tehnične omejitve in praktične težave pri prikazu tako velikih seznamov. Zavedati se moramo, da pridobivanje velikega števila zapisov iz vira podatkov (v našem primeru podatkovna baza) obremeni strežnik. Hkrati se pojavi težava prenosa tako velike količine podatkov k uporabniku. Za odpravo te težave, se je uveljavil prikaz seznamov s pomočjo paginacije. Paginator se običajno nahaja na vrhu ali dnu seznama. Preko njega se uporabnik pomika naprej oziroma nazaj po seznamu. S pomočjo paginacije omejimo prenos od strežnika do odjemalca na tiste podatke, ki bodo uporabniku dejansko vidni. S tem razbremenimo procesiranje na strežniku in sam prenosni medij. V zadnjem času se je pojavil še koncept seznamov z **neskončno paginacijo** (angl. *endless pagination*, tudi *infinite scrolling*). Tukaj se običajno prenaša samo toliko zapisov, kot jih uporabnik lahko naenkrat vidi. Ko se uporabnik pomakne po seznamu navzdol se naložijo novi zapisi. Običajno prožilec za nalaganje novih zapisov predstavlja drsni trak (angl. *scrollbar*). V okviru diplomske naloge smo večino seznamskih entitet predstavili na ta način. Dodana prednost tega načina je tudi intuitivna raba na tablicah in drugih prenosnih napravah. Slabost v primerjavi z navadno paginacijo je, da je implementacija težavnejša, hkrati pa ni mogoče preskakovati večjega števila zapisov, saj prikaz poteka v določenem zaporedju.

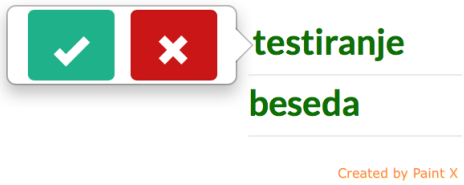
Običajno lahko nad entiteto v seznamu izvedemo eno ali več operacij, kot je npr. operacija brisanja. Klasičen seznamski prikaz ponavadi vključuje



Slika 4.4: Način delovanja neskončne paginacije. Novi zapisi se naložijo, ko drsnik doseže mejno točko. Vir: M. Tilley, ngInfiniteScroll, 2012 [52].

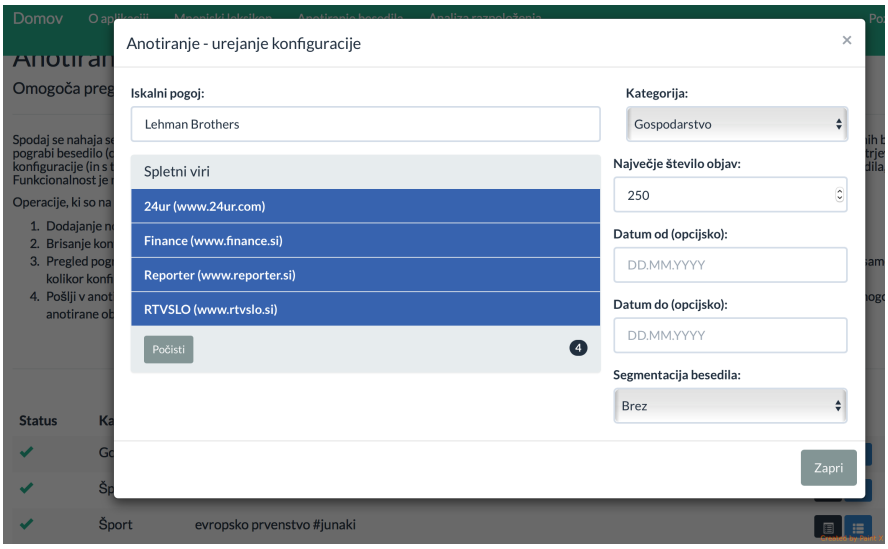
stolpec z ukazi za izvedbo določene akcije. Slabosti tega načina sta vsaj dve. Prvič, zaradi prikaza ukazov v vsaki vrstici seznama je izris spletne strani zahtevnejši, v primeru, ko je posamezen ukaz omogočen oziroma onemogočen glede na status entitete (npr. entitete ni mogoče izbrisati, če je starejša od 30 dni), pa se poveča tudi obremenitev strežnika, saj mora preverjati vrstico po vrstico, ne glede na to ali bo uporabnik ukaz sploh izrabil. Drugič, uporabnik lahko ukaz klikne po pomoti. To težavo lahko rešimo na način, da uporabnik izvedbo ukaza potrdi preko pogovornega okna (angl. *dialog box*). V naši aplikaciji smo hoteli zaobiti oba opisana problema z uvedbo **pojavne ukazne vrstice**. Ukazi so na voljo šele, ko se uporabnik s kurzorjem postavi nad element v seznamu. Poleg elementa se pojavi ukazna vrstica z operacijami, ki so nad izbranim elementom na voljo. Izris spletne strani pohitrimo, hkrati pa preprečimo, da uporabnik ukaz izvede po nesreči, saj izvedba operacije poteka v dveh korakih. Princip delovanja je prikazan na sliki 4.5. Slabost trenutne implementacije je, da je uporaba na nekaterih prenosnih napravah otežena ali celo nemogoča, saj se ukazna vrstica prikaže s tem, ko se kurzor nahaja nad elementom. To bi lahko rešili, tako da bi omogočili prikaz ukazne vrstice tudi s klikom na izbran element.

Pri urejanju elementov v seznamu smo se poslužili dveh načinov. V pri-

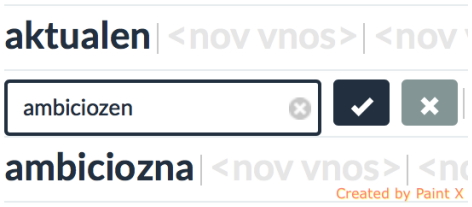


Slika 4.5: Akcije nad zapisom v seznamu z uporabo pojavne ukazne vrstice.

meru enostavne entitete (npr. beseda v slovarju sentimenta), urejanje poteka na način, da uporabnik klikne na element v seznamu. Prikaz entitete se spremeni in sicer iz načina za vpogled v način urejanja (slika 4.7), pojavi se urejevalnik, npr. vnosno tekstovno polje. Pri zahtevnejših entitetah, kjer je podatkov več in bi jih bilo nemogoče vse prikazati v seznamu, urejanje/vpogled v entiteto poteka preko **pojavnega okna** (angl. *popup window*). Po kliku na element v seznamu se prikaže pojavno okno z vsemi lastnostmi izbrane entitete (na sliki 4.6).



Slika 4.6: Prikaz podatkov entitete preko pojavnega okna.



Slika 4.7: Neposredno urejanje zapisa v seznamu.

4.1.3 Uporaba povratnih klicev

Kot smo že zapisali, je z uporabo AJAX-a mogoče zmanjšati prenos podatkov in narediti spletno stran uporabniku bolj prijazno, saj osveževanje celotne strani ni potrebno. Osvežimo le del strani, včasih uporabniku zgolj prikažemo sporočilno okno s statusom izvedene akcije. V okviru naše spletne aplikacije se večina akcij izvaja na način, kot je to predstavljeno v kodi 4.1. Ko uporabnik izvede ukaz, se najprej sproži klic strežnika v obliki asinhronega zahtevka HTTP. V zahtevku je identifikator akcije (*url*) ter parametri, potrebni za izvedbo akcije. V našem primeru sta to parametra *entryid* in *confirmationAction*. V kolikor klic uspe, se preko povratne funkcije (*success*) preveri status akcije. Odločili smo se, da bomo uspešno izvedene akcije označili s kodo 200 ($HTTPRESPONSE = 200$). V tem primeru poskrbimo za posodobitev izrisa spletne strani. Če se operacija ne izvede uspešno, uporabnika o tem obvestimo preko standardnega sporočilnega okna *alert*. Sporočilo vsebuje tudi razlog, zakaj se operacija ni izvedla uspešno. Na opisan način smo obvladovali večino uporabniških akcij.

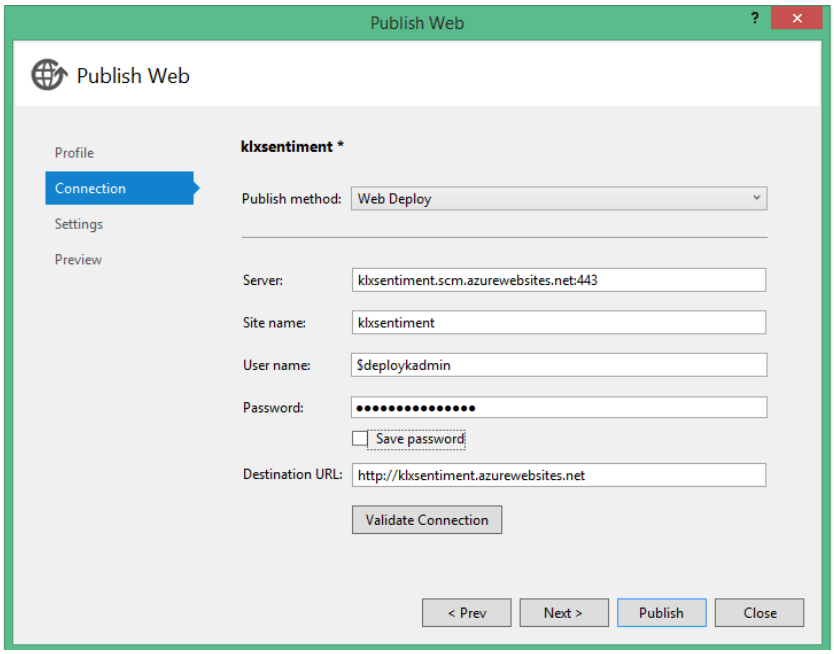
```
1 // id = entry ID
2 // action = 0: rollback, 1: confirm
3 function DoProcessEntry(id, action) {
4   if (id != null) {
5     $.ajax({
6       url: "{% url 'opinionlexicon_confirm' %}",
7       type: "POST",
8       dataType: "json",
9       data: { entryid: id, confirmationAction: action,
```

```
10                      csrfmiddlewaretoken: '{{ csrf_token }}'
11          },
12          success: function (json) {
13            if (json.HTTPRESPONSE == 200) {
14              $("#popcmd" + json.entryid.toString()).removeClass(
                    "lexicon-popcmd");
15              $("#entry" + json.entryid.toString()).addClass("
                    pending-confirmation-confirmed");
16                      }
17            else {
18              alert('Akcija ni uspela. Razlog: ' + json.msg);
19            }
20          }
21        });
22    }
23  }
```

Koda 4.1: Izvedba akcije z asinhronim zahtevkom HTTP in povratnim klicem.

4.1.4 Postavitev aplikacije

Velika prednost spletnih aplikacij v primerjavi z namiznimi je, da aplikacije ni potrebno fizično namestiti na računalnik vsakega uporabnika posebej. V primeru nove namestitve oziroma posodobitve obstoječe aplikacije datoteke zgolj prenesemo na spletni strežnik. Z uporabo VS IDE našo rešitev postavimo z nekaj kliki, saj je postopek povsem avtomatiziran. Tako ni potrebe po ročni izdelavi posebnih paketov (angl. *package*), potrebnih za namestitev. Na sliki 4.8 je prikazana postavitev aplikacije v oblak Azure z uporabo VS IDE preko pogovornega dialoga. Tudi v primeru zamenjave cilje infrastrukture, npr. uporabe lokalnega spletnega strežnika, postavitev ne bi bila težja. Pri strukturi projekta smo se držali predpisanih Django smernic, tako da lahko aplikacijo namestimo povsod tam, kjer je možno namestiti druge Django projekte. Kljub temu, da smo aplikacijo prvenstveno razvili za platformo Azure, bi jo lahko postavili tudi na drugo infrastrukturo.



Slika 4.8: Postavitev aplikacije v oblak Microsoft Azure preko VS IDE.

4.2 Modul za gradnjo slovarja sentimenta

V okviru tega sklopa smo izdelali slovar sentimentnih besed v slovenskem jeziku ter poskrbeli za možnost naknadnega posodabljanja slovarja. V nadaljevanju sledi opis izgradnje slovarja, seznanimo se s pristopi, ki se uporabljajo za gradnjo slovarjev sentimenta ter si ogledamo slabosti naše rešitve.

4.2.1 Splošno o slovarjih sentimenta

Slovarji oziroma leksikoni sentimentnih besed so osnova za leksikalno analizo sentimenta. Leksikalno analizo sentimenta smo pobliže spoznali že v uvodu. V raziskavah je bilo prikazano, da lahko uporaba tovrstnih slovarjev vodi tudi k izboljšanju klasifikatorjev, ki temeljijo na metodah strojnega učenja. V našem delu smo se osredotočili na slednjo možnost. Iz slovarja smo izluščili koristne podatke ter jih uporabili kot značilke pri gradnji klasifikatorja. Rezultate si lahko ogledate v poglavju 5.

Običajno slovarji sentimenta vsebujejo dva seznama. Prvi predstavlja besede s pozitivno konotacijo (pisci z njimi izražajo zadovoljstvo, srečo ipd.), drugi seznam pa vključuje besede z negativno konotacijo (izražanje jeze, nezadovoljstva ipd.). Slovarji lahko vključujejo tudi besedne zveze. Poleg preproste razdelitve besed v dva seznama nekateri slovarji besedam pripišejo posebno številsko vrednost, s katero lahko bolj poudarijo pozitivnost oziroma negativnost, npr. vrednost *-10* označuje skrajno negativne besede. Določeni slovarji vnose uvrstijo v posebne kategorije, npr. v kategoriji *jeza* so vključene besede, s katerimi pisci izražajo jezo, v kategoriji *preklinjati* so prisotne grobe besede ipd. Nekateri slovarji uporabljajo tudi oblikoslovne oznake - vsak vnos v slovarju je opremljen z oblikoslovno oznako. Koristi tega lahko navedemo na primeru stavka *"Na gori gori."*. Samostalnik *"gora"* je s stališča sentimenta očitna nevtralna beseda, da nekaj *"gori"* pa ima zagotvo negativen prizvok.

Večina slovarjev je na voljo za angleški jezik. Običajno se v druge jezike tovrstni slovarji prevedejo. Ta pristop smo izbrali tudi mi. Poglejmo si nekaj slovarjev sentimenta v angleškem jeziku:

- Bing Liuov [53] slovar sentimenta: avtor je uveljavljen raziskovalec analize sentimenta. Slovar sestoji iz seznama pozitivnih in negativnih besed. Pozitivnih besed v slovarju je 2006, negativnih 4783. V slovar so namenoma vključene tudi napačno črkovane in žargonske besede. Prva verzija sega v leto 2004, kot rezultat raziskave sentimenta, v kateri so kategorizirali ocene kupcev različnih proizvodov. Ta slovar predstavlja osnovo za izdelavo našega slovarja sentimenta.

- General Inquirer [54]: obširen leksikon vsebuje več kategorij. Za raziskavo sentimenta sta najbolj zanimivi kategoriji *"Positiv"* in *"Negativ"*. Vseh besed je okrog 4200. Za razliko od Liuovega slovarja vsak vnos vsebuje še dodatne metapodatke, npr. označba vseh kategorij, v katerih se beseda nahaja, pri večpomenskih besedah pomen, na katerega se beseda nanaša ipd. Slovar je dostopen tudi v slovenščini. Za potrebe

diplomske naloge ga je prevedla Mateja Volčanšek [11]. Za razliko od izvirnika preveden slovar ne vsebuje dodatnih metapodatkov.

- MPQA [55] slovar subjektivnosti: vsak vnos v slovarju vsebuje oblikoslovno oznako. Besede so razdeljene v katergoriji pozitivno in negativno. Za razliko od prejšnjih dveh slovarjev je prisoten še indikator o subjektivnosti posamezne besede. Besede, ki so subjektivne v večini kontekstih, so označene kot močno subjektivne (*"strongsubj"*), npr. beseda *osramočen*. Besede, ki so subjektivne le v nekaterih kontekstih in posledično niso tako močan indikator razpoloženja, so označene kot šibko subjektivne (*"weaksubj"*), npr. beseda *zmanjšati* [56]. Slovar za našo raziskavo ni relevanten, saj ni dostopen v slovenskem jeziku.

- SentiWordNet [57] temelji na leksikalni bazi WordNet. Vsakemu vnosu v WordNetu priredi tri numerične vrednosti, s katerimi meri pozitivnost, negativnost ter objektivnost oziroma nevtralnost pojmov. Ker obstaja leksikalna baza WordNet tudi v slovenskem jeziku[2], smo se odločili, da v raziskavo vključimo tudi ta leksikon. Prav tako je podpora SentiWordNetu že vgrajena v knjižnico NLTK.

Nabor leksikalnih virov sestavljajo naš prevod Bing Liuovega slovarja sentimenta (KSS), slovenski prevod General Inquirerja (GIS) ter SentiWordNet (SWN).

4.2.2 Pristopi h gradnji slovarjev sentimenta

Izdelave slovarjev so se raziskovalci lotili na različne načine, od povsem ročnega sestavljanja seznamov besed do pol-avtomatskega in avtomatskega grajenja.

Pri prvem načinu raziskovalci ročno gradijo različne sezname besed in fraz iz najrazličnejših leksikalnih virov. Slovarje tudi združujejo; npr. slovar

[2]Slovenska verzija Wordneta se imenuje sloWNet in je dostopna na naslovu `http://lojze.lugos.si/darja/research/slownet/`

General Inquirer je združek različnih slovarjev. Pri posameznem terminu je med dodatnimi podatki navedeno, katerim slovarjem vnos pripada. Ročna gradnja slovarjev je draga, saj je opravilo zamudno, pogosto pa terja še sodelovanje strokovnjakov. Težava se pojavlja z vzdrževanjem slovarjev, saj se jezik hitro razvija, tako da je slovarje potrebno neprestano dopolnjevati. Eden od primerov ročno grajenega slovarja je tudi Bing Liuov slovar sentimenta.

Avtomatizirani pristopi običajno potekajo tako, da se najprej ročno določi majhna začetna množica besed (semen), ki jo nato s strukturiranimi leksikalnimi bazami širimo. Za izgradnjo slovarja sentimenta bi začetno množico lahko sestavljale besede, kot so *odličen*, *prijazen*, *superioren* [58]. Najbolj znana leksikalna baza je WordNet. Predstavlja bazo besed, ki so med seboj semantično povezane. WordNet združuje besede v množice sinonimov, imenovane tudi sinseti. Vsak sinset ima enolično oznako, tako da je mogoča neposredna identifikacija istih sinsetov med različnimi jeziki. Sinseti so med seboj povezani v mrežo. Sprehajanje med njimi je mogoče z različnimi relacijami. Povedano drugače, za besedo *superiornost* lahko preko različnih relacij pridobimo sopomenke, npr. *prvovrstnost* ali *večvrednost*, ter antonime, tj. besede z nasprotnim pomenom, npr. *inferiornost*. Beseda ima lahko več pomenov, nastopa lahko v različnih sinsetih. Pomen posameznega sinseta je predstavljen z gloso (angl. *gloss*). Kratek opis pomena sinseta nam omogoča lažje pomensko razdvoumljanje besed v besedilu. Na žalost so glose v slovenskem WordNetu pomankljive, večinoma neobstoječe. Primer avtomatiziranega pristopa je SWN, ki vsak sinset iz WordNeta opremi s posebnimi sentimentnimi ocenami.

4.2.3 Izdelava slovarja sentimenta

Izdelava slovarja je potekala tako, da smo za osnovo vzeli slovar sentimentnih besed v angleškem jeziku ter ga ročno prevedli v slovenščino. Najprej smo nameravali postopek avtomatizirati z izdelavo prevajalnega orodja, ki bi se ga dalo uporabiti za nadaljne širjenje slovarja z združevanjem drugih angleških

slovarjev sentimenta. S tem bi dobili večji nabor besed. Nekateri ponudniki spletnih slovarjev ponujajo dostop tudi preko spletne storitve. Eden takšnih primerov je PONS[3]. Za ta pristop se nismo odločili, ker je uporaba storitev bodisi plačljiva ali pa omejena na dnevno kvoto povpraševanj. Ker je bila izdelava slovarja samo eden od praktičnih ciljev te naloge in ker gre pri prevajanju za enkratno opravilo, smo se odločili, da slovar izdelamo povsem ročno. Pri tem smo si pomagali s spletnimi prevajalskimi orodji. Poleg že omenjenih PONSovih slovarjev, smo uporabili še AMEBISovo prevajalsko platformo[4], ki se dobro odreže pri ponujanju sinonimov iskane besede. Ker angleški slovar namenoma vsebuje tudi besede s pravopisnimi napakami, smo za preverjanje pravopisa manj znanih besed uporabili orodje Google Translate[5]. V kolikor smo naleteli na nam nepoznane besede, smo uporabili angleški slovar z definicijami in primeri uporabe[6], s katerim smo poizkusili ugotoviti pomen besed.

Včasih angleške besede nimajo neposrednega prevoda, zato smo v teh primerih uporabili približek oziroma besedo izpustili. Skupaj s prevodi smo dodajali tudi sinonime. Za razliko od angleščine je slovenščina morfološko precej bogatejši jezik. V angleščini je različnih oblik besede le nekaj, pri slovenščini za nekatere besede obstaja tudi več deset pregibnih oblik. Ker angleški slovar običajno vsebuje več oblik besede (npr. poleg osnovne oblike se beseda *adore* pojavi še kot *adored*) smo se tudi mi odločili, da nekaj oblik besede vključimo v slovar. Zaradi bogate pregibnosti slovenskega jezika je uporaba lematizatorja še toliko priporočljivejša.

Prvo verzijo slovarja smo izdelali, še preden je bila spletna aplikacija končana. Angleški slovar smo uvozili v program, ki omogoča tabelarično urejanje podatkov. Po zaključku prevajanja smo slovenske prevode izvozili v formatu CSV (comma-separated values), kjer so prevodi ločeni z vejico, vrstice pa predstavljajo posamezne vnose v angleškem slovarju. Uvoz slovarja

[3]Dostopno na `http://www.pons.si`

[4]Dostopno na `http://presis.amebis.si/prevajanje`

[5]Dostopno na `https://translate.google.com`

[6]Dostopno na `http://dictionary.reference.com`

v našo podatkovno bazo smo izvedli z uporabo posebne skripte v Pythonu, ki je iz prevodov v formatu CSV zgenerirala ustrezne stavke SQL, ki smo jih izvedli na podatkovnem strežniku. S tem je bila izdelava slovarja sentimenta končana.

4.2.4 Podatkovni model

Podatkovni model za pridobivanje in urejanje slovarja sentimenta je preprost (na sliki 4.9). Vsebuje sledeči tabeli:

- *OpinionLexicon*: hrani podatke o vnosih v slovarju. Preko zastavice *orientation* je mogoče ugotoviti ali gre za pozitivno (vrednost 1) oziroma negativno (vrednost 0) besedo. Z zastavico *enabled* ugotovimo ali je vnos v slovarju še veljaven (vrednost 0 pomeni, da je bila beseda odstranjena). Zanimiv je tudi podatek *pendingconfirmation*, ki hrani informacijo o stanju vnosa. Vrednost 1 pomeni, da spremembe vnosa še niso bile potrjene, zato teh sprememb v slovarju ne bo. Ko se sprememba potrdi, se ustrezno dopolni tabelo, ki hrani zgodovino sprememb.

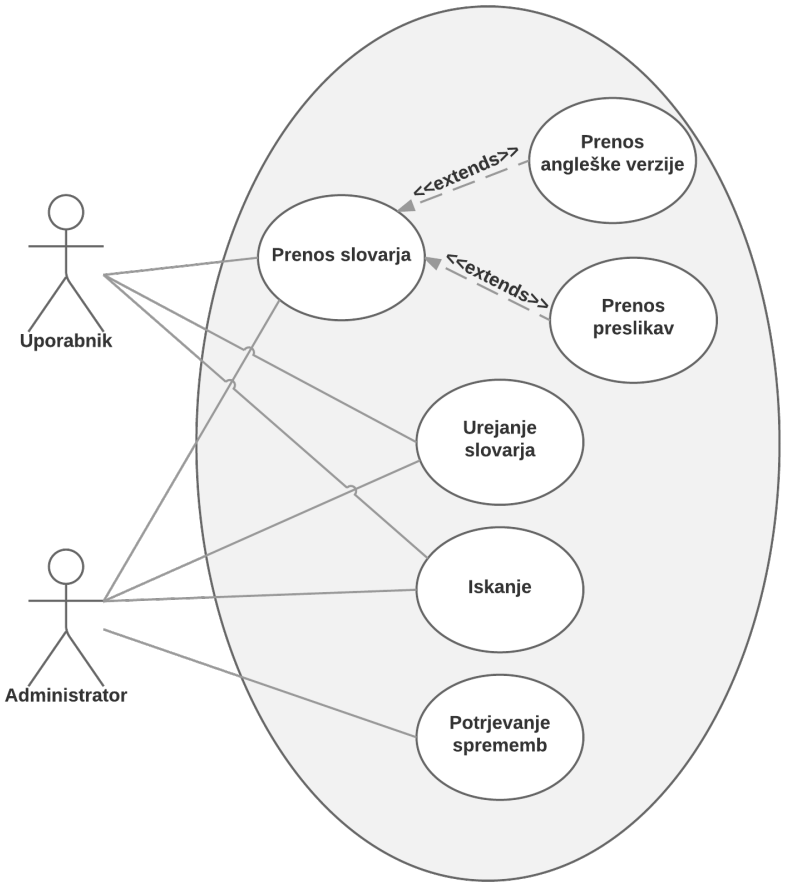
- *OpinionLexiconHistory*: vsebuje spremembe vnosov v slovarju. Čim nekdo popravi besedo in spremembo potrdi, se prejšnja vrednost zapiše v zgodovino. Preko te tabele lahko vodimo revizijsko sled. Tabela je s tujim ključem *entryId* povezana z glavno tabelo slovarja sentimenta.

Zaradi skrbnega vodenja zgodovine sprememb nam model zagotavlja, da lahko v vsakem trenutku pridobimo slovar po željeni časovni komponenti, npr. želimo pridobiti verzijo slovarja, ki je bila veljavna do leta 2016. Prav tako bi lahko povrnili vse spremembe, v kolikor bi ugotovili, da spremembe slovarja botrjujejo k slabšim rezultatom klasifikacije ipd.

Slika 4.9: Podatkovni model za slovar sentimenta.

4.2.5 Uporabniške akcije nad slovarjem

Nad slovarjem sentimenta uporabniki izvajajo različne akcije (slika 4.10). Najbrž je najpomembnejša funkcija **prenos slovarja** k uporabniku. Pri prenosu se zgenerira zadnja verzija slovarja, ki vsebuje vse potrjene popravke. Slovar je uporabniku dosegljiv v obliki kompresirane datoteke ZIP, ki vsebuje seznam pozitivnih (*positive_words.txt*) ter seznam negativnih (*negative_words.txt*) besed. Poleg prenosa slovenskega slovarja sentimenta smo uporabnikom dali možnost prenosa originalnega angleškega slovarja ter prenos preslikav med angleškimi in slovenskimi besedami, ki so nastale tekom izdelave slovarja. Slovar je mogoče pregledovati tudi na spletu. Poleg izbire orientiranosti besed je mogoče **iskanje** po korenu besed. **Urejanje slovarja** je implementirano preko hitrega, neposrednega spreminjanja besed v seznamu. Urejanje je na voljo tako administratorjem kot navadnim uporabnikom. Veljavna verzija slovarja vedno vključuje samo spremembe, ki so bile potrjene. Tako pridemo do pojma **potrjevanja sprememb**. S potrjevanjem sprememb se popravki slovarja bodisi potrdijo (popravek je dober in naj se vključi v slovar) bodisi zavrnejo (popravek naj se izloči). S popravki imamo v mislih tako urejanje obstoječih vnosov v slovarju kot tudi brisanje in dodajanje novih. Akcija potrjevanja je namenjena samo administratorjem sistema.



Slika 4.10: Diagram primerov uporabe slovarja sentimenta.

4.2.6 Slabosti slovarja in implementacije

Tekom izdelave diplomskega dela smo se srečali z nekaterimi pomankljivostmi izbranega pristopa izdelave slovarja ter tehnične implementacije.

Osnovna pomankljivost tovrstnih slovarjev je v tem, da ne upoštevajo konteksta. Za primer lahko navedemo pridevnik *velik*. V kontekstu opisa prenosnega telefona (*"Telefon je velik."*) ima negativno konotacijo. Če ga uporabimo v kontekstu oglasa za hišo (*"Hiša ima velik balkon."*), pa gre za zaželjeno lastnost. Določena beseda ima, s stališča sentimenta, v različnih

kontekstih različen pomen. To je pogosta težava tovrstnih slovarjev. Težavo bi lahko odpravili (ali vsaj omilili) z izdelavo kontekstnih slovarjev. Za splošno analizo sentimenta bi morala zadostovati tudi naša rešitev. Nadalje se postavi vprašanje smiselnosti vključevanja različnih morfoloških oblik besede v slovar. Ker je slovenščina pregibno bogat jezik, bi to pomenilo, da bi posamezna beseda, seveda v drugi obliki, lahko bila prisotna tudi več desetkrat. Za dodajanje različnih oblik bi lahko uporabili slovar besednih oblik. Pojavi se vprašanje, če bi nemara bilo bolj smiselno v slovar dodajati samo osnovne oblike besed ter uporabnike slovarja prepričati, da je pri rabi slovarja obvezna predhodna lematizacija.

S stališča tehnične implementacije je ročno potrjevanje popravkov nepraktično opravilo. Napake uporabnikov, ki slovar urejajo, lahko s potrjevanjem resda eliminiramo. Težava se pojavi, ker ne vemo, kako dobri so ti popravki. Manjka mera, s katero bi lahko določili kvaliteto slovarja. Če bi popravek poslabšal kvaliteto slovarja, bi bil izločen že s strani sistema. Za ta namen bi lahko dodali posebno podporno storitev, ki bi v ozadju, dnevno oziroma od zadnje uspešne preverbe, preverjala kvaliteto slovarja. Za namen ugotavljanja kvalitete slovarja bi denimo lahko uporabili kar naš klasifikator sentimenta. Če bi s popravki dosegli slabšo natančnost klasifikatorja, bi popravke zavrnili.

4.3 Modul za označevanje besedila

V sklopu modula za označevanje besedila smo izdelali orodje, ki omogoča pridobivanje poljubnih uporabniških komentarjev ter podpira njihovo označevanje. Končni izdelek je korpus označenih uporabniških komentarjev v slovenskem jeziku. V nadaljevanju si ogledamo potek izdelave orodja, predstavimo pomen označevanja besedila in izpostavimo nekatere slabosti ter možne izboljšave orodja in samega korpusa.

4.3.1 Zakaj označevanje besedila?

Označevanje besedila pomeni besedilo označiti z eno ali večimi vrednostmi, ki imajo v kontekstu označevanja določen pomen. V našem primeru so to sentimentne ocene, ki najboljše opišejo dano besedilo s stališča klasifikacije sentimenta. Označevanje besedila izvajamo s ciljem pridobitve zadostnega števila primerov za učenje klasifikatorja ter preverjanja natančnosti klasifikacije.

Označevanje primerov izvajamo ročno, polavtomatsko ali avtomatsko. Pri prvem pristopu se primeri označujejo ročno na način, da človeški označevalec besedilo prebere in označi z oznako za katero misli, da ga najboljše opisuje. Običajno je opravilo prepuščeno strokovnjakom iz področja problemske domene. Obstajajo tudi spletne platforme[7], ki omogočajo najetje delavcev za to opravilo. Pri polavtomatskem pristopu se običajno označi manjše število primerov in nato z različnimi metodami širi njihovo število. Povsem avtomatski način pa vključuje pregledovanje pojavnic v besedilu ali koriščenje metapodatkov, ki se nanašajo na besedilo. Kot primer prvega načina lahko izpostavimo iskanje emotikonov v besedilu. Če je emotikonov, ki predstavljajo veselje, več kot tistih, ki predstavljajo žalost, besedilo označimo z eno, sicer pa z drugo kategorijo. Kot primer izkoriščanja metapodatkov lahko navedemo spletno filmsko bazo IMDB[8], ki je zanimiva zaradi tega, ker omogoča svojim uporabnikom komentiranje in ocenjevanje filmov. Komentator poleg zapisa komentarja film tudi oceni na lestvici od 1 do 10. Z interpretacijo vrednosti na lestvici lahko besedilo samodejno označimo, npr. opis z oceno med 7 in 10 predstavlja besedilo s pozitivnim sentimentom.

4.3.2 Ideja

Večina raziskav sentimenta je v angleškem jeziku. Analogno temu je tudi večina prosto dostopnih korpusov označenih besedil v tem jeziku. Na voljo

[7]Ena najbolj znanih je Amazon Mechanical Turk, ki je dostpna na `http://www.mturk.com`

[8]Dostopno na `http://www.imdb.com`

so korpusi neformalnih in formalnih besedil, od objav na mikroblogih, političnih debat do označb novic. Za slovenski jezik je zgodba drugačna. V času pisanja diplomske naloge nismo našli prosto dostopnega korpusa označb UGC v slovenskem jeziku, zato smo se odločili, da korpus zgradimo sami.

Poudarek smo želeli dati klasifikaciji neformalnih besedil. Odločali smo se med objavami na platformi Twitter ter uporabniškimi komentarji na slovenskih novičarskih portalih. S tehničnega stališča je v prid uporabe platforme Twitter govoril prosto dostopen API[9], ki omogoča iskanje tvitov na podlagi različnih kriterijev. Za pridobivanje komentarjev bi ga morali napisati sami, saj slovenski portali tovrstne storitve ne nudijo. Odločili smo se za drugo možnost. Menili smo, da bo s komentarji identifikacija subjektivnega besedila lažja, saj Twitter veliko uporabljajo tudi uradne entitete, kot so podjetja in tiskovne agencije, ki objavljajo objektivnejša besedila. Obenem bi lahko izmerili, v kolikšni meri je ugotovitev sentimenta komentarja odvisna od novice, na katero se komentar nanaša.

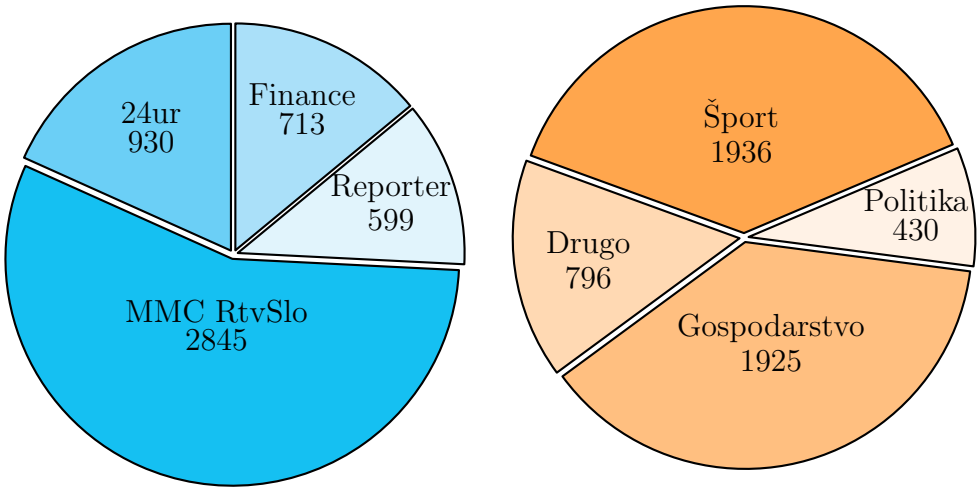
Za luščenje komentarjev smo izbrali naslednje spletne vire: 24ur[10], Finance[11], Reporter[12], MMC RtvSlo[13]. Po določitvi spletnih virov smo se srečali s težavo, kako pridobiti komentarje, ki bi se nanašali na poljubno izbrano temo. Naredili smo naivno predpostavko, da se komentarji vedno nanašajo na novico. Za pridobivanje relevantnih komentarjev je bilo potrebno najti način, kako programsko pridobiti spletne naslove novic na katere se komentarji nanašajo. Odločili smo se za uporabo Googlovega APIja za iskanje, ki ponuja možnost iskanja po poljubnih spletnih straneh na podlagi iskalnega pogoja (ključne besede, obdobje objave ipd.). Kombinacijo iskalnega pogoja, spletnih virov ter dodatnih podatkov, denimo omejitev števila komentarjev, smo poimenovali konfiguracija. Ideja je bila, da ima administrator sistema možnost določitve poljubnega števila konfiguracij, s poljubnimi iskalnimi po-

[9]Dostopno na `https://dev.twitter.com/rest/public/`
[10]Dostopno na `http://www.24ur.com/`
[11]Dostopno na `http://www.finance.si/`
[12]Dostopno na `http://www.reporter.si/`
[13]Dostopno na `http://www.rtvslo.si/`

goji in možnostjo omejitve dosega konfiguracije na točno določen spletni vir, npr. s konfiguracijo naj se iz spletnih virov *"24ur"* in *"MMC RtvSlo"* pridobi komentarje, ki ustrezajo iskalnemu pogoju *"Poleti v Planici"*. S tem bi dobili dovolj prilagodljivo orodje za luščenje zadostnega števila komentarjev.

4.3.3 Izdelava korpusa uporabniških komentarjev

Za izdelavo korpusa smo pridobili 5087 uporabniških komentarjev iz različnih spletnih virov. Razporeditev komentarjev po virih je prikazana na grafu 4.11(a). Razporeditev komentarjev po kategorijah oziroma temah je prikazana na grafu 4.11(b). Za pridobivanje komentarjev smo določili 24 konfiguracij, uporabljeni iskalni pogoji so prikazani v tabeli 4.1. Vseh različnih spletnih strani, iz katerih smo izluščili komentarje je bilo 427. Iz vsake strani smo povprečno pridobili 11 komentarjev.



(a) Po spletnih virih (b) Po kategorijah

Slika 4.11: Razporeditev uporabniških komentarjev.

iskalni pogoj	**iskalni pogoj**
Lehman Brothers	odbojka prvenstvo tretje mesto
Olimpija Maribor naslov	puščavski lisjak dakar
evropsko prvenstvo #junaki	Cimos rešen DUTB
prevc skupni seštevek	Volkswagen afera
TEŠ6 Zares	Trg nepremičnin rast
Slovenija Ukrajina evropsko prvenstvo	Cipras grška kriza
gospodarska rast izboljšanje	begunska kriza
slovensko smučanje smučarija	cenejši bencin
Peter Prevc skoki	Tina maze prvakinja prvenstvo
SDH slovenski državni holding	maribor liga prvakov
brezposelnost nižja zmanjšanje	turizem rast 2015
Goran Dragić pogodba	

Tabela 4.1: Uporabljeni iskalni pogoji za luščenje komentarjev.

Za označevanje smo določili naslednje oznake oziroma kategorije:

- **Pozitivno**: v primeru, da komentar jasno izraža pozitiven sentiment, ga uvrstimo v to kategorijo. Pri tem pazimo, da v kategorijo ne uvrščamo komentarjev, ki so lahko pozitivni, vendar predstavljajo preprosto dejstvo ali se berejo kot novice.

- **Negativno**: sem spadajo komentarji z izraženim negativnim sentimentom. Če se avtor na temo sklicuje v negativnem smislu, ga uvrstimo sem. Pogosto se najdejo tudi vprašanja z negativno konotacijo, npr. *"Pa kaj ti je danes??"*.

- **Nevtralno**: v to kategorijo spadajo komentarji, ki ne vsebujejo izraženega mnenja ali elementov subjektivnosti, npr. dejstva, novice. V kolikor komentar ne izraža pozitivnega ali negativnega sentimenta, ga uvrstimo sem. Ta kategorija je namenjena tudi komentarjem, ki sicer vsebujejo tako pozitiven kot negativen sentiment, vendar nobeden ni prevladujoč. Tudi v primeru dvoumnosti komentar uvrstimo v to kategorijo.

- **Irelevantno**: s to kategorijo označujemo komentarje, ki za naš sistem niso relevantni. Pri izgradnji korpusa se ne upoštevajo. V to kategorijo spadajo komentarji, ki niso v slovenskem jeziku, komentarji, ki vsebujejo samo spletne povezave, slike ipd.

Odločili smo se, da uvedemo še posebno oznako, ki bi nosila informacijo o tem, ali je bilo za označevanje potrebno zunanje znanje oziroma poznavanje konteksta, tj. novice na katero se komentar nanaša. Označevalci te oznake ne postavljajo neposredno. Izkoristili smo sledenje ogleda spletnega vira s strani označevalca. V kolikor si označevalec ogleda zunanji vir, se komentarju nastavi ta oznaka. Zanimalo nas je, koliko informacije potrebuje označevalec za kategorizacijo komentarja, ali zadostuje samo besedilo ali je za odločitev o izbiri kategorije potreboval poznavanje širšega konteksta.

Posamezne komentarje smo združili v serije po 500 komentarjev. V posamezni seriji so bili naključno izbrani komentarji iz različnih konfiguracij. Menili smo, da 500 komentarjev predstavlja obvladljivo količino dela, ki ga označevalec lahko opravi brez prekinitev in hkrati ne vpliva na slabšo kvaliteto dela. Serije smo označili po vrstnem redu. S tem je bilo zagotovljeno, da je šel vsak označevalec skozi identičen postopek označevanja. Dobra lastnost tega je, da bi v primeru sočasnega dela več anotatorjev lahko sprotno spremljali parametere pri poteku izgradnje korpusa, npr. sprotno merjenje strinjanja med označevalci.

V procesu označevanja so sodelovali trije označevalci. Vsi so šli skozi isti nabor 5087 uporabniških komentarjev. Pred pričetkom dela smo jim razložili posamezne kategorije. Z nalogo so opravili povsem samostojno. Strategija se je izkazala za pomankljivo, saj smo dosegli zgolj povprečne rezultate v okviru označevanja. Možne izboljšave strategije za označevanje predlagamo v poglavju 4.3.7. Pri označevanju sentimenta gre za zelo subjektivno opravilo, zato je težko pričakovati visoko stopnjo strinjanja med označevalci. Zanesljivost korpusa označenih uporabniških komentarjev smo merili z večimi merami. Stopnjo strinjanja dveh označevalcev smo izmerili s statistično me-

ritvijo Cohen Kappa po naslednji enačbi [59]:

$$\kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e}, \tag{4.1}$$

kjer je p_o relativno strinjanje med označevalcema, p_e je hipotetična verjetnost strinjanja. Za razliko od preprostega izračuna odstotka strinjanja med označevalcema, κ po Cohenu torej upošteva še strinjanje po naključju. $\kappa = 1$ pomeni popolno strinjanje med označevalcema [59]. Izračunali smo naslednje vrednosti koeficienta κ po Cohenu: med prvim in drugim označevalcem znaša 0,326, med prvim in tretjim 0,310 ter med drugim in tretjim 0,542. Ker smo dobili večje razlike med ujemanjem prvega in drugih dveh označevalcev, bomo v poglavju 5 eksperimentirali z učenjem klasifikatorja na podlagi tistih primerov, katerih kategorija je bila pri označevanju zastopana vsaj z dvotretjinsko večino.

Stopnjo strinjanja vseh označevalcev skupaj smo izračunali z uporabo mere Fleiss Kappa. Za razliko od κ po Cohenu, je Fleiss Kappa namenjena ugotavljanju zanesljivosti strinjanja med večimi označevalci. Izračunali smo jo po sledeči formuli [60]:

$$\kappa = \frac{\overline{P} - \overline{P}_e}{1 - \overline{P}_e}, \tag{4.2}$$

kjer faktor $1 - \overline{P}_e$ pomeni stopnjo strinjanja, ki je dosegljiva nad strinjanjem po naključju, $\overline{P} - \overline{P}_e$ je stopnja strinjanja, ki je nad naključno dejansko dosežena. V kolikor med označevalci ni drugega strinjanja kot zgolj tisto, pričakovano po naključju, je vrednost $\kappa \leq 0$ [60]. Dosežena vrednost koeficienta κ po Fleissu je bila 0,379. Landis in Koch sta izdelala tabelo za interpretacijo vrednosti κ [60]. Po njuni interpretaciji smo dosegli ustrezno ujemanje $(0,21 \leq \kappa \leq 0,40)$.

Zanimivi so bili rezultati povezani s spremljanjem posebne oznake, ki je označevala komentarje, za kategorizacijo katerih so označevalci potrebovali poznavanje zunanjega konteksta. Takšnih primerov je bilo 328, kar znese dobrih 6% vseh komentarjev. Iz tega lahko sklepamo, da na ugotavljanje semantične vrednosti komentarja poznavanje novice nima prevelikega vpliva. Interpretacija je lahko tudi drugačna. Pomeni lahko, da se na dejansko novico

nanaša manjše število komentarjev, ali da označevalci že poznajo kontekst, saj gre za dovolj znane teme.

Uravnotežen korpus označenih uporabniških komentarjev verzije 1.0 tako vsebuje po 580 pozitivnih, negativnih in nevtralnih komentarjev. Verzija brez uravnoteženja vsebuje 898 pozitivnih, 3291 negativnih ter 588 nevtralnih komentarjev. Podrobnejši razpored primerov po posameznih kategorijah novic in spletnih virih je prikazan v tabelah 4.2 in 4.3.

	Gospodarstvo	**Politika**	**Šport**	**Drugo**
Pozitivno	129	26	679	64
Nevtralno	262	33	240	53
Negativno	1420	351	882	638

Tabela 4.2: Razporeditev komentarjev po kategorijah novic.

	MMC RtvSlo	**24ur**	**Finance**	**Reporter**
Pozitivno	566	255	54	23
Nevtralno	441	48	75	24
Negativno	1614	584	554	539

Tabela 4.3: Razporeditev komentarjev po spletnih virih.

4.3.4 Podatkovni model

Podatkovni model (na sliki 4.12) za modul označevanja besedila sestoji iz naslednjih tabel:

- *OpinionContentSources*: vsebuje seznam podprtih spletnih portalov oziroma spletnih virov. Preko njih lahko izluščimo uporabniške komentarje. Vsebuje osnovne lastnosti o mediju, kot so naslov, oznaka ter logo.

- *AnnotationConfiguration*: hrani konfiguracije za označevanje. Vsaka konfiguracija vsebuje iskalni pogoj, omejitev števila izluščenih komentarjev ter določitev kategorije. Vsebuje še nekaj zastavic, ki povedo ali je bila konfiguracija izbrisana (*deleted*), so bili komentarji za konfiguracijo že izluščeni (*processed*) ter ali je konfiguracija že dostopna za označevalce (*inUse*).

- *AnnotationConfigurationContentSources*: vmesna tabela med tabelo konfiguracij in spletnimi viri. Za vsako konfiguracijo je lahko določenih več spletnih virov.

- *AnnotationContent*: vsebuje komentarje uporabnikov. Poleg besedila hranimo tudi metapodatke, ki so povezani s komentarjem, in sicer čas objave komentarja (*createdDateTimeUtc*), uporabniško ime (*userData*), sliko (*avatar*) ter spletni naslov (*url*). Vsak komentar je povezan s konfiguracijo. Komentar vsebuje tudi zgoščeno kodo (*contentHash*), s katero preprečujemo njihovo podvajanje.

- *AnnotationContentBatch*: tabela hrani serije uporabniških komentarjev za označevanje. Serija vključuje skupek največ 500 komentarjev, ki so urejeni po vrstnem redu (*position*). Komentar se vedno pojavi samo v eni seriji.

- *AnnotationData*: v tabeli se nahajajo označbe uporabniških komentarjev. Glavna polja so orientiranost komentarja (*value*), uporabnik, ki je komentar označil (*userId*), čas označbe (*createdDateTimeUtc*) ter zastavica *contextRequired*, ki hrani podatek o tem ali je bil za označbo potreben ogled zunanjega vira (novice), na katerega se komentar nanaša.

- *AnnotationContextRequiredVolatile*: hrani identifikatorje tistih komentarjev, za katere je uporabnik izvedel vpogled v zunanji vir oziroma novico. Ker v času vpogleda označba komentarja še ne obstaja, je potrebna dodatna tabela. Ko se za komentar naredi označba (tabela

AnnotationData), se nastavi tudi zastavica *contextRequired*, v kolikor je uporabnik izvedel vpogled v novico.

- *AnnotationDataset*: v tabeli se nahajajo generirani korpusi označenih uporabniških komentarjev. Vsaka verzija korpusa je shranjena v dveh različicah, uravnotežni (*balanced*) in takšni, ki vsebuje vse označbe. Korpus je shranjen v formatu XML. Poleg korpusa se nahajajo še nekateri dodatni podatki, kot so število komentarjev po posameznih oznakah (*negCount*, *posCount*, *neuCount*), čas izdelave (*createdDateTimeUtc*) in verzija (*version*) korpusa.

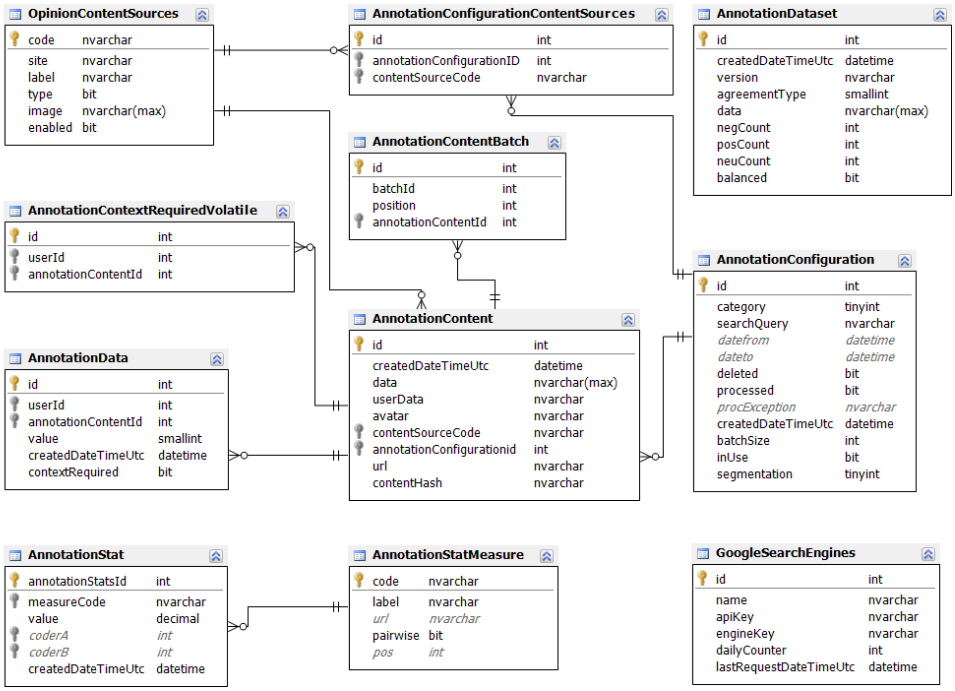
- *AnnotationStatMeasure*: tabela z definicijami mer, ki jih vodimo v okviru preračunavanja statistik pri označevanju. Zanimiva je zastavica *pairwise*, ki pove, ali gre pri meri za primerjavo dveh označevalcev; npr. stopnja strinjanja dveh označevalcev.

- *AnnotationStat*: hrani izračunane statistike. Vsaka statistika je povezana z mero preko enolične oznake (*measureCode*). Če gre za statistiko, ki velja med dvema označevalcema, se identifikator prvega shrani v polje *coderA*, za drugega pa v polje *coderB*. V tabelo piše izključno podporna storitev, ki dnevno izračunava statistike.

- *GoogleSearchEngines*: vsebuje podatke o Googlovih iskalnikih in ključih, ki nam omogočajo iskanje po spletnih virih z uporabo Googlovega APIja.

4.3.5 Podporne storitve

V okviru implementacije so nastale podporne storitve, ki zagotavljajo podporo procesu pridobivanja komentarjev iz različnih spletnih virov, generiranju korpusa ter izračunavanju statistik. V nadaljevanju jih bomo podrobneje predstavili.



Slika 4.12: Podatkovni model za označevanje besedila.

4.3.5.1 Pridobivanje uporabniških komentarjev

Uporabniške komentarje je bilo potrebno najprej izluščiti iz različnih spletnih virov. Ker gre za časovno potratno opravilo, bi bilo to nesmotrno storiti že v okviru shranjevanja konfiguracije. Proces luščenja za posamezno konfiguracijo, v odvisnosti od števila najdenih relevantnih spletnih naslovov, števila ter omejitve relevantnih uporabniških komentarjev, lahko traja tudi več minut.

Odločili smo se, da pridobivanje komentarjev izvedemo kot podporno storitev. Izdelali smo orodje **AnnotationDataGrabber**, ki periodično pregleduje nove, še neobdelane konfiguracije. V kolikor takšna konfiguracija obstaja, se preko Googlovega APIja za iskanje pridobi množico spletnih naslovov, za vsakega izmed spletnih virov glede na določen iskalni niz. Ko se pridobi ciljne spletne naslove, se v zanki sprehodi čez vse s strani konfiguracije določene spletne vire. Za vsak spletni vir mora biti narejena implementacija luščenja uporabniških komentarjev. Definirali smo abstraktni razred *Web-*

SiteGrabber, ki služi kot osnova za vse konkretne implementacije. Razred vsebuje abstraktno metodo *grab(url, maxcount = 1000)*, ki kot parameter prejme ciljni spletni naslov in maksimalno število izluščenih komentarjev. Sledila je implementacija pridobivanja komentarjev za različne spletne vire, *FinanceGrabber* za spletni vir *Finance*, *RtvSloGrabber* za spletni vir *MMC RtvSlo* ipd. Po zaključku pridobivanja komentarjev za spletne vire je potekala izbira ciljnih komentarjev. Za vsakega izmed virov smo izbrali naključne komentarje ter jih čimbolj enakomerno razporedili. Sledilo je zapisovanje v podatkovno bazo. S tem je bila obdelava ene konfiguracije za označevanje zaključena. V primeru, da je med pridobivanjem komentarjev prišlo do napake, smo le-to ujeli in razlog za neuspešno obdelavo spletnega vira zapisali v bazo.

Poleg pridobivanja komentarjev je naloga orodja tudi združevanje komentarjev v serije po 500 za potrebe lažjega vodenja procesa označevanja. Za primer združevanja se pridobi množico komentarjev, ki še niso uvrščeni v nobeno izmed serij. Glede na število komentarjev se naredi eno ali več novih serij. Naj omenimo, da se v serije združuje samo komentarje za konfiguracije, ki jih administrator eksplicitno označi. Tako se naredi več konfiguracij, medtem ko se jih za označevanje označi samo nekaj.

Pri implementaciji smo naleteli na nekaj težav. Prvič, Googlov API za iskanje ima omejitev 100 zahtevkov na dan. To omejitev lahko hitro presežemo. Zato smo se odločili, da število iskanj povečamo z generiranjem večih Googlovih računov. Vpeljali smo posebno tabelo z naštetimi ključi in iskalnimi pogoni. Za potrebe diplomskega dela smo jih določili 5. Število navzgor ni omejeno. S tem smo povečali kapaciteto na 500 iskanj dnevno. Implementirali smo razred *GoogleSearchManager*, katerega naloga je preklapljanje med Google API računi ter vodenje iskanj. V bazi detajlno, za vsak račun, vodimo datum in čas zadnjega iskanja ter število iskanj. Težave je povzročilo tudi luščenje komentarjev. Za nekatere spletne vire je luščenje komentarjev trivialno, denimo za spletni vir *Reporter*, saj so komentarji kot statična vsebina del strani. Pri drugih spletnih virih npr. *MMC RtvSlo* pa komentarji

niso del strani in jih je z ločenimi zahtevki HTTP potrebno pridobiti ločeno.

4.3.5.2 Generiranje korpusa uporabniških komentarjev

Korpus označenih uporabniških komentarjev se zgradi z orodjem **CorpusGenerator**. Vedno se zgradita dve verziji korpusa. Prva predstavlja uravnotežen korpus, kjer komentarje enakomerno razporedimo po različnih oznakah. Druga predstavlja korpus brez uravnoteževanja, v tem primeru se v korpusu nahajajo vsi označeni komentarji.

Pri gradnji korpusa najprej preverimo, ali obstaja kakšen nov uporabniški komentar. V kolikor ne obstaja, generiranje ni potrebno. V nasprotnem primeru se pridobi seznam označenih komentarjev za množico označevalcev. Ker lahko označevalci isti komentar označijo z različnimi oznakami, smo naredili mehanizem za izbiro najprimernejše oznake. Izbira oznake se izvede po principu večinskega glasovanja (angl. *majority voting*). To pomeni, da izberemo tisto oznako, ki je bila izbrana s strani največ označevalcev.

Korpus je predstavljen v obliki dokumenta XML (primer je na sliki 4.13). Za format XML smo se odločili, ker se je uveljavil kot de-facto standard pri izmenjavi podatkov med različnimi sistemi, podpora za branje strukture XML pa je dostopna v veliki večini programskih jezikov. Izdelali smo tudi shemo XSD, ki nosi definicijo XML za korpus označb.

4.3.5.3 Izračunavanje statistik za označevanje

Z **AnnotationStatistic** izračunavamo statistike za mere, kot so skupna stopnja strinjanja človeških označevalcev, medsebojno ujemanje dveh označevalcev, število vseh označenih komentarjev ipd. Preračunavanje izvajamo dnevno.

Pri izračunu stopenj strinjanj vedno vzamemo le relevantne komentarje, torej tiste, ki niso označeni z oznako *999*. V kolikor ne obstajata vsaj dva označevalca, stopenj strinjanja ne izračunavamo. Med označevalci so lahko velike razlike v številu označenih komentarjev. Kot spodnji prag smo določili 500 označenih komentarjev. Za to smo se odločili, ker so nekatere mere

```
<?xml version="1.0" encoding="UTF-8"?>
<AnnotatedData version="0.803" createdDateTimeUtc="2015-12-18T21:25:26.138390"
xmlns="http://klxsentiment.azurewebsites.net/annotation/annotateddata.xsd">
  - <AnnotatedItems>
      - <AnnotatedItem id="12298">
            <Source>rtvslo</Source>
            <SourceUri>http://www.rtvslo.si/sport/oi-2014/alpsko-smucanje/maze-moj-sportni-cikel-se-
                koncuje-z-olimpijado/329767</SourceUri>
            <User>habela</User>
            <CreatedDateTimeUtc>2014-02-13T12:24:00</CreatedDateTimeUtc>
            <Category>Šport</Category>
          - <Content>
                <![CDATA[čestitke tini ekipi in meni je zelo žal,da bo končala kariero.Mislim da ne bomo kmalu ali
                nikoli imeli smučarke kot je tina.Ampak sedaj je čas za druge stvari,srečno tina]]>
            </Content>
            <Score contextRequired="false" codersNum="1">1</Score>
        </AnnotatedItem>
      - <AnnotatedItem id="9746">
            <Source>rtvslo</Source>
            <SourceUri>http://www.rtvslo.si/gospodarstvo/januarja-se-je-brezposelnost-krepko-
                povecala/357413</SourceUri>
            <User>kolnkista</User>
            <CreatedDateTimeUtc>2015-02-03T12:18:00</CreatedDateTimeUtc>
            <Category>Gospodarstvo</Category>
          - <Content>
                <![CDATA[Krisss, tole ja vsaj za Tednik, če ne še za Preverjeno. Ljudje morajo vedeti kako se delajo
                norca zaposleni na zavodu z zagarantiranimi službami z ljudmi, ki so obupani in naredijo vse, kar jim
                te birokrati naročijo.]]>
            </Content>
            <Score contextRequired="false" codersNum="1">-1</Score>
        </AnnotatedItem>
    </AnnotatedItems>
</AnnotatedData>
```

Slika 4.13: Korpus uporabniških komentarjev v formatu XML. Zaradi preglednosti vsebuje samo dva komentarja.

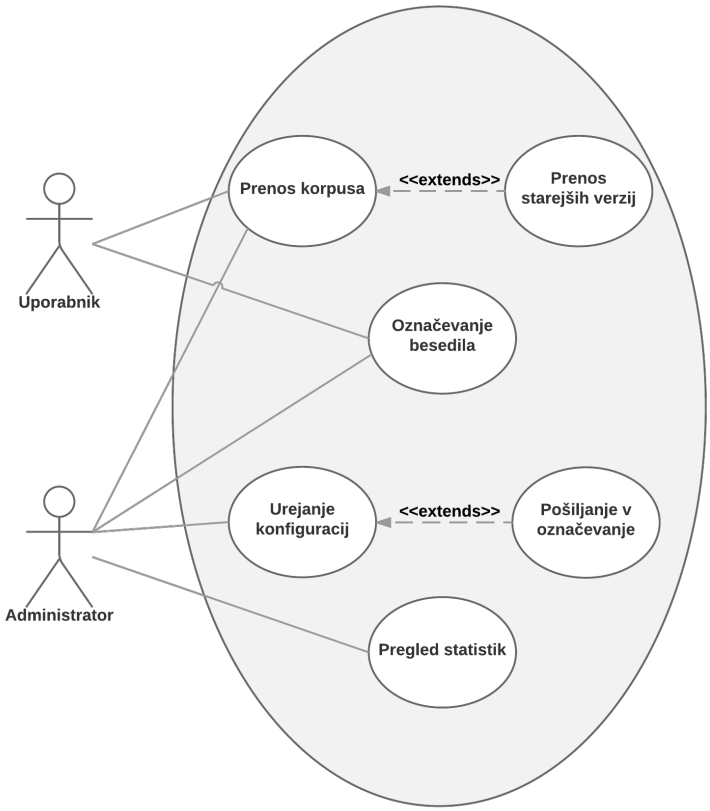
občutljive na vhodne podatke. S tem smo se hoteli izogniti situaciji, ko bi primerjali označevalca s 1000 komentarji in označevalca z 10 komentarji. Za izračunavanje vseh stopenj strinjanj smo uporabili razred *AnnotationTask* iz knjižnice NLTK. Uporabniške komentarje smo pretvorili v množico trojčkov (*3-tuples*), kjer prvi podatek predstavlja označevalca, drugi identifikator komentarja ter tretji oznako oziroma kategorijo. S seznamom trojčkov se razred *AnnotationTask* inicializira.

4.3.6 Uporabniške akcije pri označevanju besedila

Diagram na sliki 4.14 predstavlja akcije, ki jih uporabniki izvajajo v okviru spletnega vmesnika.

Možnost prenosa korpusa označenih komentarjev predstavlja za večino uporabnikov najpomembnejšo funkcijo. Dosegljiv je v obliki kompresirane

datoteke ZIP, ki vsebuje korpus v formatu XML, opis strukture korpusa v obliki sheme XSD ter krajšo tekstovno datoteko z nekaterimi informacijami. Če želijo, lahko starejše verzije prenesejo preko pojavnega okna, v katerem se nahaja seznam prejšnjih verzij korpusa. Za razliko od slovarja sentimenta označenih komentarjev na spletu ni mogoče pregledovati.

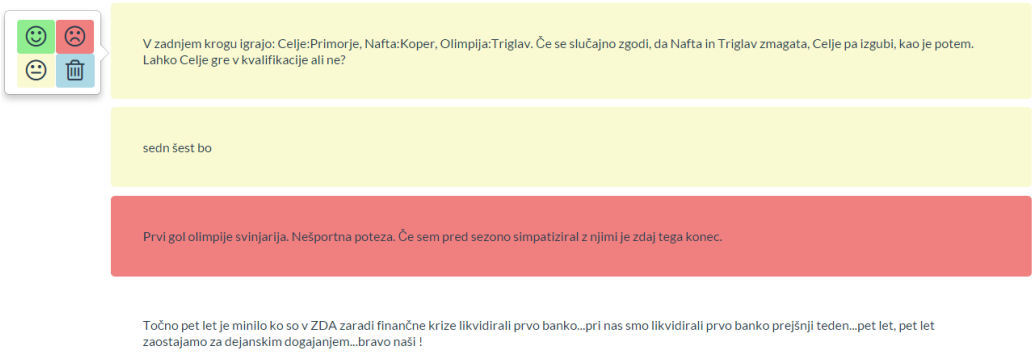


Slika 4.14: Diagram primerov uporabe v okviru modula za označevanje besedila.

Uporabnikom je na voljo posebna stran za označevanje. Preko pojavne ukazne vrstice izberejo sentimentno oznako za komentar (na sliki 4.15), s posebnim gumbom pa jim je omogočen vpogled v novico, na katero se komentar nanaša. Do prve osvežitve strani lahko oznako komentarja tudi spremenijo. Stran prikazuje posamezne serije komentarjev. V vsakem trenutku je tako

prikazanih največ 500 komentarjev. Na naslednjo serijo se lahko uporabnik premakne šele, ko konča z označevanjem trenutne.

Administratorju je na voljo še ogled statistik v obliki enostavnega seznama ter urejanje konfiguracij za označevanje. Za vsako konfiguracijo si lahko ogleda seznam izluščenih spletnih komentarjev. Konfiguracijo, ki jo želi poslati v označevanje, mora administrator eksplicitno potrditi.



Slika 4.15: Označevanje uporabniških komentarjev preko spletnega vmesnika.

4.3.7 Slabosti in možne izboljšave

Pri uporabljenem pristopu izdelave korpusa označenih komentarjev in same tehnične implementacije smo naleteli na nekatere slabosti, za odpravo katerih je zmanjkalo časa. V nadaljevanju naštejemo nekaj očitnih slabosti in predlogov za izboljšanje:

- Strategija označevanja: k zgolj zadovoljivi stopnji strinjanja med označevalci je botrovala slabo zamišljena strategija označevanja. Krajši napotki glede pomena oznak so bili premalo. Kljub temu, da gre pri označevanju sentimenta za zelo subjektivno opravilo, bi morali pred nalogo označevalcem ponuditi več označenih primerov. Kasneje, po nekaj narejenih označbah, bi morali skupaj razrešiti morebitne nejasnosti, ki so se pri nalogi pojavile. Za ponovitev označevanja po tej strategiji, s katero bi morda dosegli višjo stopnjo strinjanja, je zmanjkalo časa.

- Spremljanje procesa označevanja: proces označevanja bi morali sprotno spremljati. Ker gredo označevalci čez isti nabor komentarjev, po istem vrstnem redu, bi lahko hitro ugotovili morebitno preveliko neujemanje z uporabo različnih metrik.

- Število komentarjev na posameznega uporabnika: pri novicah se pogosto srečamo s pojavom, da eni in isti uporabniki objavijo tudi 10 in več komentarjev, kar lahko privede do pristranskosti (angl. *bias*). Pri pridobivanju komentarjev bi bilo idealno, da bi zajeli mnenja kar največ različnih uporabnikov. Naše orodje trenutno ne omogoča izbire komentarjev na takšen način. Opisani težavi se poizkušamo izogniti tako, da pridobimo čimvečje število komentarjev in jih nato naključno izberemo.

- Večinsko glasovanje pri izbiri oznake: v primeru, ko označevalci komentar označijo z različnimi oznakami, uporabimo večinsko glasovanje. Ta način ni najboljši, saj vsi označevalci predstavljajo enako težo pri odločanju. Kot primer lahko navedemo označevalca, ki bi teoretično vse komentarje dal v eno samo kategorijo, npr. *pozitivno*. Neresnosti navkljub bi po večinskem glasovanju njegove označbe nosile enako težo. Temu bi se lahko izognili na dva načina. Pri prvem bi hranili seznam označevalcev s števcem, ki bi ga povečali, v kolikor bi bila izbrana označevalceva označba. Teža odločitve vsakega označevalca bi bila enačena s številom izbranih označb. Označevalec, ki bi vse komentarje označil z isto oznako, bi tako imel manjšo težo. Drug način bi bil lahko s stopnjo strinjanja med različnimi označevalci. Trenutno je podprto samo večinsko glasovanje.

- Kategorija/tema na katero se nanašajo komentarji: pri urejanju konfiguracije se ročno izbere kategorija, za katero mislimo, da najboljše opisuje iskalni pogoj; npr. za pogoj *"Novoletna skakalna turneja"* je primerna izbira kategorije *"Šport"*. Ker pa lahko iskalnik vrne manj relevantne rezultate in se najdena stran ne identificira s športom, ampak

kakšno drugo kategorijo, bi bilo smiselno razmisliti o tem, da bi kategorijo najdene strani avtomatsko razpoznali, ali pa najdene zadetke še dodatno filtrirali preko ročno izbrane kategorije. V prvem primeru bi podatek iz konfiguracije umaknili in ga zapisali neposredno h komentarju. V drugem primeru bi podatek ostal del konfiguracije, omejili bi le pridobivanje komentarjev na dejansko kategorijo. V obeh primerih bi potrebovali dodaten klasifikator za kategorizacijo tem. Alternativna rešitev bi bila podrobna definicija Googlovih iskalnih pogonov; npr. na *"MMC RtvSlo"* naj iskalnik za kategorijo *"Šport"* išče spletne strani samo na podstraneh, ki se začnejo z *"http://www.rtvslo.si/sport/"*, namesto po korenski domeni *"http://www.rtvslo.si/"*.

- Kasnejša nedostopnost spletnih strani: naš podatkovni model ne omogoča shranjevanje vsebine spletne strani. V komentarjih shranjujemo zgolj povezave na spletne vire. Podatkovni model bi bilo potrebno razširiti, da bi omogočal shranjevanje vsebine spletne strani, saj se lahko pojavi težava, ko stran postane nedostopna. V okviru diplomske naloge povezava na novico ne predstavlja nobene dodane vrednosti pri klasifikaciji, saj vsebine spletne strani ne vključujemo v naš klasifikacijski model. Lahko bi model razširili z upoštevanjem zunanjih informacij, tudi vsebino spletnih strani. V podatkovni model bi bilo potrebno dodati novo tabelo, ki bi hranila vsebine spletnih strani, vsak komentar pa bi bil s svojim izvorom povezan preko tujega ključa na novo tabelo.

- Izboljšave pri iskanju z Google API: Googlovo omejitev 100 iskanj na posamezen račun dnevno smo uspešno zaobšli z definiranjem večih uporabniških računov in iskalnih pogonov. Omenili smo izdelavo namenskega razreda, ki detajlno vodi število iskanj, skupaj s časom zadnjega iskanja, za vsakega izmed računov ter po potrebi preklaplja med njimi. Težava je, ker razred ne podpira drsnega okna, tj. sprotnega sproščanja iskanj na posameznem pogonu, kar pomeni, da v kolikor smo z enim

računom dosegli dnevno kvoto 100 iskanj, bo ta račun 24 ur za iskanje nedosegljiv, čeprav smo denimo 99 iskanj izvedli 20 ur nazaj. Težava ni kritična, saj v vsakem trenutku lahko dodamo dodatne iskalne pogone in tako razširimo iskalne zmogljivosti.

- Pridobivanje komentarjev za druge namene: v tej diplomski nalogi smo funkcionalnost pridobivanja komentarjev implementirali za potrebe označevanja besedila. Model bi lahko razširili in sistem pridobivanja komentarjev uporabili za praktične analize, npr. spremljanje podpore predsednika države tekom let.

4.4 Modul za evalvacijo

V modulu za evalvacijo se nahaja objektni model za predobdelavo besedila, pripravo značilk ter napovedovanje sentimenta, skupaj z demonstracijo klasifikacijskega modela.

Rešitve v objektnem modelu sledijo smernicam za analizo sentimenta, ki smo jih postavili v poglavju "Klasifikacija sentimenta". Glavna cilja gradnje modela sta bila realizacija klasifikacije sentimenta ter možnost vključevanja funkcionalnosti v druge aplikacije. Objektni model za klasifikacijo smo uporabili tako v spletni aplikaciji kot tudi v spletni storitvi. Model sestoji iz:

- Razred **Database**: je namenjen delu s podatkovno bazo. Omogoča povezavo na bazo ter izvedbo stavkov SQL.
- Datoteka **SharedTypes**: vsebuje definicije tipov, ki se uporabljajo v več komponentah modula; npr. enumeracija **Orientation** vsebuje vrednosti, s katerimi predstavimo semantično vrednost besedila.
- Korpusi
  - Razred **AnnotatedCorpusBase**: omogoča dostop do označenih uporabniških komentarjev. Na voljo je uravnotežena verzija korpusa (razred **BalancedAnnotatedCorpus**) ter korpus z vsemi

primeri (razred **AnnotatedCorpus**). Prva inicializacija korpusa traja nekaj sekund, zato je predpomnjen za vsako nadaljno uporabo.

- Razred **SloOpinionLexicon**: predstavlja slovar slovenskih sentimentnih besed. Preko funkcij lahko pridobimo ločene sezname vseh besed, vseh pozitivnih ter vseh negativnih besed. Slovar je prav tako predpomnjen.
- Razred **VolcansekLexicon**: slovar sentimenta, ki je nastal v okviru diplomske naloge Mateje Volčanšek [11]. Za osnovo smo uporabili NLTKjev razred **OpinionLexiconCorpusReader**, ki je namenjen branju seznama pozitivnih in negativnih besed.
- Razred **SloStopwords**: vsebuje seznam slovenskih neinformativnih besed. Tudi v tem primeru smo uporabili infrastrukturo iz knjižnice NLTK in sicer smo za osnovo vzeli razred **WordListCorpusReader**, ki je namenjen branju tekstovnih datotek, kjer so besede ločene z znakom za novo vrstico.
- Razred **EmoticonLexicon**: vsebuje seznam emotikonov s pozitivno in negativno konotacijo. Za osnovo smo vzeli Hogenboomov slovar emotikonov s sentimentnimi oznakami[14]. Uporabili smo emotikone s pozitivno in negativno konotacijo, nevtralne smo prezrli.

- Datoteka **Preprocessing**: vsebuje razrede, s katerimi smo implementirali funkcionalnost predobdelave besedila. Pomembnejši med njimi so:
  - **RegExPreprocItemBase**: abstraktni razred služi kot osnova za vse konkretne implementacije, ki se tičejo predobdelave na način, da se na vhodnem besedilu uporabijo poljubni regularni izrazi.

[14]Slovar je dostopen na naslovu `https://www.w3.org/community/sentiment/wiki/Datasets#Emoticon_Sentiment_Lexicon`

Večino specifičnih načinov predobdelave smo reševali na ta način, npr. z razredom **UrlPreprocItem** smo spletne naslove zamenjevali z značko *URL*.

- **TokenPreprocItemBase**: osnova za vse načine predobdelave, ki jih izvajamo nad posameznimi pojavnicami oziroma po tokenizaciji in ne neposredno nad vhodnim besedilom. Primer tega je razred **StopwordsPreprocItem**, s katerim identificiramo in odstranimo neinformativne besede.
- **RawPreprocItemBase**: služi kot osnova za vse načine predobdelave neposredno nad vhodnim besedilom, ki jih samo s procesiranjem pojavnic ali uporabo regularnih izrazov ne moremo rešiti, npr. obvladovanje negacije z razredom **NegationPreprocItem**.
- **TextPreprocessor**: krovni razred za predobdelavo besedila. Z njim besedilo tokeniziramo ter uporabimo vse specifične načine predobdelave, od lematizacije do obvladovanja negacije, glede na konfiguracijo. Razred se lahko uporabi ločeno, ali v navezi s knjižnico scikit-learn (koda 4.2).

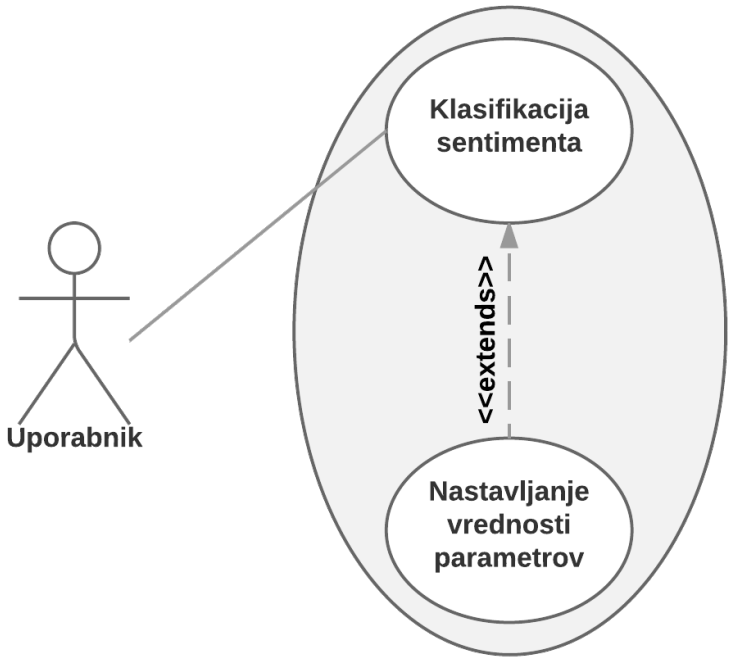
- Razred **Evaluator**: namenjen je klasifikaciji sentimenta. Za dano vhodno besedilo ugotovi polariteto ter kot rezultat vrne eno izmed vrednosti: pozitivno, negativno ali nevtralno. Poleg evalvacije besedila je na voljo še možnost testiranja poljubnih konfiguracij ter evalvacija rezultatov klasifikacije z več merami.

```
1 from Preprocessing import TextPreprocessor
2 from sklearn.feature_extraction.text import CountVectorizer
3
4 preproc = TextPreprocessor(tokenizer_type=Tokenizers.potts,
5             raw_preprocessors=[RawPreprocessorType.negation])
6 vectorizer = CountVectorizer(tokenizer=preproc)
7 ...
```

Koda 4.2: Inicializacija razreda za vektorizacijo z vrečo besed iz knjižnice scikit-learn, ki za predobdelavo uporablja našo rešitev.

4.4.1 Uporabniške akcije pri evalvaciji besedila

Diagram na sliki 4.16 predstavlja akcije, ki jih uporabniki izvajajo v okviru tega modula. Najpomembnejša je klasifikacija sentimenta s privzetimi vrednostmi parametrov modela za evalvacijo. Uporabniki vnesejo poljubno besedilo v vnosno polje ter s klikom na gumb sprožijo postopek analize sentimenta nad vnešenim besedilom. Poleg analize s privzetimi vrednostmi parametrov, lahko uporabniki obnašanje modela za klasifikacijo spremenijo s spremembo vrednosti nekaterih parametrov, npr. izbiro tokenizatorja ali izbiro metode za strojno učenje.



Slika 4.16: Diagram primerov uporabe v okviru modula za evalvacijo besedila.

4.5 Spletna storitev

Integracija s sistemom je mogoča preko spletne storitve, ki uporablja protokol SOAP 1.1. Med odjemalcem in strežnikom se prenašajo sporočila XML. Spletna storitev ponuja naslednje metode:

- **Evaluate(text)**: za podano besedilo se izvede popolna analiza sentimenta. Če je besedilo pozitivno, se vrne vrednost 1, če je negativno, vrednost -1, ter vrednost 0 za nevtralno besedilo.
- **GetCorpus(balanced = True)**: vrne zadnjo verzijo korpusa označenih uporabniških komentarjev v formatu XML. Privzeto se vrne uravnotežen korpus.
- **GetCorpusDefinition()**: vrne shemo XSD, ki določa strukturo korpusa uporabniških komentarjev.
- **GetOrientation(word)**: v slovarju sentimentnih besed poišče dano besedo in vrne njeno polarnost. Če besede v slovarju ni, vrne vrednost 0. V nasprotnem primeru vrne vrednost 1 (beseda s pozitivno konotacijo) oziroma vrednost -1 (beseda z negativno konotacijo). Če je parameter metode daljše besedilo, se polarnost ugotovi za prvo besedo v besedilu.

Nabor funkcionalnosti, ki so dostopne preko spletne storitve, bi lahko še razširili, npr. z vračanjem seznama verzij korpusa uporabniških komentarjev, pridobivanjem specifične verzije korpusa, vračanjem seznama statistik ali določitvijo dodatnih parametrov pri analizi sentimenta.

Poglavje 5

Rezultati eksperimentov

V tem poglavju eksperimentiramo z evalvacijo različnih metod za klasifikacijo, merimo vpliv predobdelave besedila ter priprave značilk na uspešnost klasifikacije.

Najprej določimo zlati standard za učenje in testiranje klasifikatorja, opišemo uporabljene mere za ocenjevanje uspešnosti klasifikacije ter definiramo osnovno konfiguracijo (angl. *baseline*), s katero bomo primerjali vse nadaljne poskuse izboljšave klasifikacijskega modela. Sledi eksperimentiranje z različnimi konfiguracijami. Na koncu poglavja si ogledamo končni klasifikacijski model, tj. konfiguracijo, s katero smo v fazi eksperimentiranja dosegli najboljše rezultate. Ocenimo še vpliv velikosti korpusa na klasifikacijsko uspešnost. Omeniti moramo, da kategorije novic komentarjev nismo posebej upoštevali. Osredotočili smo se na izgradnjo splošnega klasifikacijskega modela.

5.1 Priprava

Naš **zlati standard** za učenje in testiranje klasifikacije sestavljajo označeni uporabniški komentarji. Izbrali smo uravnotežen korpus komentarjev z enakomerno razporeditvijo po vseh treh kategorijah. Vsaka kategorija vsebuje 580 komentarjev. Za zlati standard predpostavljamo, da so komentarji pra-

vilno označeni. Vključeni so le tisti komentarji, za katere je obstajalo določeno strinjanje med označevalci (več v poglavju 4.3.3).

Komentarjev v korpusu nismo ročno razdelili na učno in testno množico, ampak smo za preverjanje uspešnosti klasifikacije uporabili **prečno preverjanje**. Pri prečnem preverjanju vzamemo nekaj podatkov za potrebe testiranja, ostale uporabimo za učenje klasifikatorja, v več iteracijah. Uporabili smo 10 kratno prečno preverjanje. Za prečno preverjanje namesto ločene testne množice smo se odločili zaradi razloga majhnega korpusa komentarjev. Pri razdelitvi na učni in testni del bi obe množici vsebovali le malo primerov, tako bi bilo preverjanje bolj občutljivo na izbiro primerov.

Za ocenjevanje uspešnosti klasifikacije smo uporabili več mer. Za lažjo ponazoritev posameznih mer si bomo pomagali z **matriko zmot** (angl. *confusion matrix*) [61], ki prikazuje napovedane in prave razrede. V tabeli 5.1 je predstavljena matrika zmot za dvorazredni problem. Vsota posamezne vrstice predstavlja delež pravih razredov. Vsota posameznega stolpca nam pove delež primerov, ki jih je klasifikator za posamezen razred napovedal. Diagonala matrike predstavlja delež pravilnih klasifikacij.

	napovedani razred		
pravi razred	P	N	vsota
P	TP	FN	POS=TP+FN
N	FP	TN	NEG=FP+TN
vsota	PP=TP+FP	PN=FN+TN	n=TP+FP+FN+TN

Tabela 5.1: Matrika zmot za dvorazredni problem.

Prva uporabljena mera je **klasifikacijska točnost** (angl. *classification accuracy, CA*) [61], s katero ocenjujemo uspešnost klasifikacije. Klasifikacijska točnost je podana z:

$$T = \frac{TP + TN}{TP + FP + FN + TN} = \frac{TP + TN}{n}, \tag{5.1}$$

kjer vrednost koeficienta T predstavlja razmerje med vsoto pravilno klasifici-

ranih primerov ter številom vseh primerov n. Za opredelitev tega ali je klasifikator dober ali ne, običajno CA ne zadostuje. Zamislimo si primer, kjer je zastopanost razreda N 100 primerov, razreda P pa 10 primerov. Klasifikator lahko vse primere klasificira v večinski razred N (v tabeli 5.2). S tem bomo dobili visoko vrednost CA, v našem primeru je $T = \frac{0+100}{0+10+0+100} = 90.9\%$. Kljub temu je klasifikator neuporaben, saj je informativnost napovedovanja nična [62]. Zato običajno CA kombiniramo z drugimi merami.

	napovedani razred	
pravi razred	P	N
P	0	10
N	0	100

Tabela 5.2: Paradoks klasifikacijske točnosti.

Poleg CA se običajno uporabljata še **priklic** (angl. *recall*) in **natančnost** (angl. *precision*). S priklicem ocenjujemo delež pravilno klasificiranih pozitivnih primerov. Priklic je določen z [61]:

$$Priklic = \frac{TP}{TP + FN} = \frac{TP}{POS} \tag{5.2}$$

Z natančnostjo ocenjujemo delež pravilno klasificiranih primerov, ki so bili klasificirani kot pozitivni. Natančnost je določena z [61]:

$$Natančnost = \frac{TP}{TP + FP} = \frac{TP}{PP} \tag{5.3}$$

Meri sta mnogokrat obratno sorazmerni. Izboljšanje priklica privede do znižanja natančnosti in obratno. Za spremljanje obeh mer istočasno imamo na voljo mero F_1 (angl. F_1-*measure*), ki jo izrazimo kot [61]:

$$F_1 = \frac{2 \times Priklic \times Natančnost}{Priklic + Natančnost} = \frac{2TP}{2TP + FP + FN} \tag{5.4}$$

Za eksperimentiranje smo uporabili klasifikatorje iz knjižnice scikit-learn. V tabeli 5.3 so navedeni razredi iz knjižnice scikit-learn za uporabljene me-

tode strojnega učenja. Vse klasifikatorje smo preizkušali s privzetimi vrednostmi parametrov.

Metoda strojnega učenja	**scikit-learn klasifikator**
Logistična regresija (LR)	linear_model.LogisticRegression
Metoda podpornih vektorjev (SVM)	svm.LinearSVC
Multinomski naivni Bayes (MNB)	naive_bayes.MultinomialNB
Bernoullijev naivni Bayes (BNB)	naive_bayes.BernoulliNB

Tabela 5.3: Uporabljeni klasifikatorji.

Za konec je bilo potrebno določiti osnovo oziroma osnovno konfiguracijo, s katero bi lahko primerjali izboljšave klasifikacijskega modela. Osnovna konfiguracija bi bila lahko določena s klasifikacijo primerov v večinski razred, ker pa smo uporabili povsem uravnotežen korpus in je bila učna množica enakomerno razporejena med tremi razredi, bi osnovo predstavljala vrednost CA 33.3%. Ta vrednost predstavlja spodnjo mejo še sprejemljive klasifikacijske točnosti [61]. Ker smo želeli dobiti čimbolj realne podatke o vplivu možnih izboljšav sistema (predobdelava besedila, izbira značilk ipd.) na uspešnost klasifikacije, večinska klasifikacija ne bi bila dovolj informativna. Odločili smo se, da osnovo izberemo na podlagi rezultatov testiranja različnih metod strojnega učenja. Za značilke smo izbrali posamezne besede. Razen predstavitve z unigrami drugih značilk nismo izločali, prav tako komentarji niso bili posebej predobdelani. Osnova vsebuje 20388 značilk. Za tokenizacijo smo uporabili tokenizator s presledki. V tabeli 5.4 vidimo, da vsaka izmed metod strojnega učenja preseže izbiranje z naključno izbiro. Glede na rezultate, našo osnovno konfiguracijo sestavlja klasifikator LR z vrečo besed kot značilkami, ki ima nekoliko višji CA in mere F_1 kot MNB.

Klasifikator	CA	mera F_1			
		pos	*neg*	*neu*	povp.
LR	**54,5**	57,6	51,5	54,3	**54,5**
SVM	52,7	54,6	49,2	53,8	52,5
MNB	54,1	55,9	58,2	45,8	53,3
BNB	42,2	54,3	30,8	22,8	36,0

Tabela 5.4: Izbira osnovne konfiguracije, vrednosti so v procentih. Mera F_1 je izražena za vsak razred posebej, skupaj s povprečjem. S krepko pisavo so označene najboljše vrednosti za CA in povprečje mere F_1.

5.2 Vpliv predobdelave besedila na rezultate klasifikacije

Najprej nas je zanimal vpliv izbire tokenizatorja na uspešnost klasifikacije. V osnovni konfiguraciji je uporabljena tokenizacija s presledki. V okviru eksperimentiranja smo preizkusili še druga dva tokenizatorja (več v poglavju 2.2). V tabeli 5.5 vidimo, da oba tokenizatorja izboljšata uspešnost klasifikacije tudi do slabih 5% v primerjavi z osnovno konfiguracijo. Tokenizacija po Pottsu za več kot 20% zmanjša število unigramov. Ker mera F_1 v tabeli ni prikazana, omenimo, da je približno enaka pri obeh tokenizatorjih. Zaključimo lahko, da smo najboljše rezultate dobili s Pottsovim tokenizatorjem, kar je povsem v skladu s Pottsovimi raziskavami [25] glede vpliva tokenizacije na uspešnost klasifikacije. Pottsove raziskave so bile sicer osredotočene na klasifikacijo tvitov v angleškem jeziku, vendar iz dobljenih rezultatov vidimo, da je njegov tokenizator v veliko pomoč tudi pri klasifikaciji uporabniških komentarjev v slovenskem jeziku.

Nadalje nas je zanimal vpliv lematizacije besedila na uspešnost klasifikacije. Resda se je z lematizacijo besedila uspešnost klasifikacije izboljšala samo za 1 – 2%, vendar v tabeli 5.6 opazimo, da se je zmanjšalo tudi število unigramov za več kot 23%. To pomeni, da hranjenje različnih morfoloških oblik

Tokenizator	# značilk	LR	SVM	MNB	BNB
s presledki	20388	54,5	52,7	54,1	42,2
Treebank	17264	58,7	**56,0**	**58,7**	45,4
Potts	16027	**59,1**	55,5	58,6	**46,1**

Tabela 5.5: Vpliv tokenizacije na uspešnost klasifikacije. S krepko pisavo je za vsako metodo strojnega učenja označen najboljši tokenizator.

besed nima dodane vrednosti na klasifikacijo sentimenta, ravno nasprotno, z lematizacijo smo dosegli celo boljše rezultate.

Lematizacija	# značilk	LR	SVM	MNB	BNB
NE	20388	54,5	52,7	54,1	42,2
DA	15584	**55,5**	**53,8**	**55,6**	**44,1**

Tabela 5.6: Vpliv lematizacije na klasifikacijo sentimenta.

Naslednji eksperiment (tabela 5.7) je vključeval izločitev neinformativnih besed (SW), dodali smo še preverjanje vpliva dolžine odstranjenih SW. Malce presenetljivo je blokiranje SW poslabšalo CA vseh metod strojnega učenja. Prav tako je padla mera F_1. Očitno je, da imajo tudi (nekatere) SW sentimentno vrednost. Lahko bi prilagajali seznam SW za potrebe sentimenta, vendar to presega okvire tega dela. SW smo odstranjevali na način, da smo najprej uporabili tokenizator ter nato nad posameznim elementom poiskali ujemanje s seznamom SW. Preizkusili smo tudi Pottsov tokenizator. Tudi tukaj smo prišli do podobnih rezultatov, z razliko, da se je za klasifikator MNB klasifikacijska točnost celo nekoliko izboljšala. Iz obeh primerov lahko zaključimo, da je odstranjevanje SW nevtralno za klasifikator MNB, pri ostalih klasifikatorjih uspešnost klasifikacije pade. Nadalje smo poizkusili še s pragom dolžine besed, za katere se preverja ali so v seznamu SW. Ker nam v nobeni konfiguraciji ni uspelo izboljšati rezultatov klasifikacije, smo sklenili, da se blokiranju SW pri nadaljnem eksperimentiranju odpovemo.

Blokiranje besed	# značilk	LR	SVM	MNB	BNB
osnova	20388	**54,5**	**52,7**	**54,1**	**42,2**
+ SW	19091	51,8	49,3	53,3	40,1
+ SW <3	20133	54,0	52,1	54,0	41,7
+ SW <4	19949	52,2	49,3	53,7	41,0

Tabela 5.7: Blokiranje neinformativnih besed. *SW* pomeni odstranjevanje vseh neinformativnih besed, *<3* krajših od treh znakov, *<4* krajših od štirih znakov.

Eksperimentirali smo tudi z obvladovanjem negacije in merili njen vpliv na uspešnost klasifikacije. Negacijo smo obravnavali tako, da smo dodajali predpono *"NEG"* vsem besedam, ki se pojavijo v kontekstu negacije, tj. od besede, s katero smo negacijo identificirali, pa do konca stavka oz. do prvega ločila. Besede, s katerimi smo identificirali negacijo so bile: ne, ni, nikoli, nikakor, noče. Zavedamo se, da je nabor teh besed pomankljiv. Za pravilnejše obvladovanje negacije bi morali upoštevati kompleksna lingvistična pravila. Z negacijo na predstavljen način nismo dosegli željenega (tabela 5.8) - uspešnost vseh klasifikatorjev se je poslabšala. Tudi število značilk se je opazno povečalo. Sklepamo lahko, da smo prispevali le več šuma. Pang in sod. [17] so na opisan način reševali problem negacije pri klasifikaciji opisov filmov v angleškem jeziku. Dosegli so zanemarljivo izboljšanje v primerjavi z osnovo.

Negacija	# značilk	LR	SVM	MNB	BNB
NE	20388	**54,5**	**52,7**	**54,1**	**42,2**
DA	21454	52,5	52,1	54,0	41,5

Tabela 5.8: Obvladovanje negacije in rezultati klasifikacije.

Nadaljevali smo z meritvami specifičnih načinov predobdelave besedila, kot je denimo zamenjava spletnih povezav s pojavnico *URL* (več v poglavju 2.2). Zanimal nas je vpliv posameznih načinov predobdelave na uspešnost

klasifikacije. Rezultati so prikazani v tabeli 5.9. Zanimivo je, da posamično največ načinov predobdelave pripomore k izboljšanju klasifikatorja SVM, vendar pri preizkusu z vsemi načini skupaj, pri SVM pride do najmanjšega izboljšanja uspešnosti klasifikacije. Vse predobdelave skupaj ugodno vplivajo na uspešnost klasifikacije.

Predobdelava	LR	SVM	MNB	BNB
osnova	54,5	52,7	54,1	42,2
naslovi URL	**54,8**	**52,8**	54,0	42,2
id uporabnika	54,4	**53,6**	**54,5**	**42,4**
hashtagi	**54.7**	**52,8**	54,0	42,2
zaporedje črk	54,4	**53,0**	**54,3**	42,2
emotikoni	**54,7**	**53,3**	**54,3**	42,2
vse velike črke	54,5	52,7	54,1	42,2
zaporedje ločil	**56,2**	**54,3**	**55,5**	**42,4**
zamenjava števil	**54,8**	**53,9**	**55,6**	**42,9**
odstranitev ločil	**56,5**	**53,3**	**58,0**	**44,1**
SKUPAJ	**58,3**	**54,7**	**59,0**	**44,9**

Tabela 5.9: Vpliv posameznih načinov predobdelave besedila na uspešnost klasifikacije. S krepko pisavo so za posamezno metodo strojnega učenja označeni tisti načini predobdelave, ki so boljši od osnove.

Oglejmo si še najboljšo konfiguracijo, ki smo jo dobili s predobdelavo besedila (tabela 5.10). Konfiguracija vključuje Pottsov tokenizator, lematizacijo besedila ter večino specifičnih načinov predobdelave besedila. Za razliko od osnovne konfiguracije se je tukaj najbolje izkazal klasifikator MNB. S predobdelavo smo pridobili dobrih 8% pri CA, število unigramov smo zmanjšali na 9198, kar je več kot 50% manj kot pri osnovni konfiguraciji. Podoben učinek je imela predobdelava tudi na mero F_1. Pri CA in meri F_1 so pridobile vse metode strojnega učenja. Z rezultati smo pokazali, da ima predobdelava na klasifikacijo sentimenta velik vpliv.

Klasifikator	CA	mera F_1			
		pos	*neg*	*neu*	povp.
osnova (LR)	54,5	57,6	51,5	54,3	54,5
LR	61,8	66,8	58,6	60,1	61,8
SVM	59,7	64,6	55,9	58,4	59,6
MNB	**63,2**	67,3	64,2	57,6	**63,0**
BNB	48,9	57,5	44,4	36,4	46,1

Tabela 5.10: Izbira najboljše konfiguracije po predobdelavi besedila.

5.3 Priprava značilk

V prejšnjem poglavju smo merili vpliv predobdelave besedila na uspešnost klasifikacije na modelu, kjer so bile značilke predstavljene z vrečo besed. V tem si ogledamo vpliv izločanja, izbire ter uravnoteženja značilk. Preizkusili bomo klasifikacijo z uporabo višjih *N-gramov*, pogledali ali lahko z uporabo slovarjev sentimenta izboljšamo rezultate klasifikacije, ugotavljali vpliv uteži ter si ogledali rezultate izbire najboljših značilk z uporabo metode *Hi-kvadrat*.

Najprej smo preizkusili kombinacijo različnih *N-gramov* (tabela 5.11) saj smo predpostavljali, da bi lahko zajetje širšega konteksta (negacija, fraze ipd.) koristilo pri klasifikaciji. Izkazalo se je, da je kombinacija različnih *N-gramov* prinesla uspeh le v primeru klasifikatorja SVM, v ostalih primerih so bile vrednosti podobne modelu z unigrami. Opazen je še padec rezultatov klasifikatorja BNB. Glede na velik padec CA pri precej višjem številu značilk bi lahko sklepali, da je klasifikator zelo občutljiv. Naj omenimo, da smo evalvirali višje *N-grame* tudi z omejitvijo števila značilk, pri čemer smo značilke omejili s pogostostjo pojavitve v korpusu, vendar se uspešnost klasifikacije ni bistveno spremenila.

V poglavju 4.2.1 smo predstavili slovarje sentimenta, ki so osnova za naslednji eksperiment. Želeli smo izmeriti, ali lahko z vključevanjem leksikalnih virov izboljšamo rezultate klasifikacije. Za eksperiment smo uporabili naslednje leksikalne vire: slovar sentimenta, ki je nastal v okviru te diplomske na-

Model	# značilk	LR	SVM	MNB	BNB
unigrami	9198	**61,8**	59,7	**63,2**	**48,9**
unigrami + bigrami	53499	61,2	**60,1**	59,0	43,3
bigrami	44301	51,0	49,5	51,6	38,3
uni + bi + trigrami	118274	60,6	59,4	56,4	39,4

Tabela 5.11: Izločanje značilk z uporabo različnih stopenj N-gramov.

loge (KSS), slovar sentimenta M. Volčanšek (GIS) ter Wordnetova razširitev s sentimentnimi ocenami, leksikon SentiWordnet (SWN). Rezultati so predstavljeni v tabeli 5.12. Najboljše se je obnesel slovar, ki smo ga izdelali v okviru te diplomske naloge. Z njim smo dosegli zaznavno izboljšanje klasifikacije pri vseh metodah strojnega učenja, v primeru klasifikatorja SVM za 1,3%. S slovarjem GIS smo dobili mešane rezultate. Najslabše se je odrezal slovar SWN. Razloge za to gre lahko iskati v dejstvu, da SWN različne pomene iste besede točkuje z različnimi sentimentnimi ocenami. Za učinkovito izrabo slovarja SWN bi bilo potrebno vključiti sistem razdvoumljanja večpomenskih besed, tako da bi dobili sinset s pravim pomenom in posledično pravilnejšo sentimentno oceno, vendar slednje že presega okvire tega dela. Tudi z vključitvijo vseh slovarjev skupaj v povprečju nismo uspeli bistveno izboljšati rezultatov slovarja KSS. Sklenemo lahko, da leksikalni viri pozitivno vplivajo na analizo sentimenta z uporabo metod strojnega učenja.

Model	LR	SVM	MNB	BNB
unigrami	61,8	59,7	63,2	48,9
unigrami + KSS	**62,9**	**60,6**	64,5	49,8
unigrami + GIS	61,5	59,5	64,4	49,4
unigrami + SWN	61,0	59,8	63,4	49,2
SKUPAJ	62,2	60,5	**65,2**	**50,3**

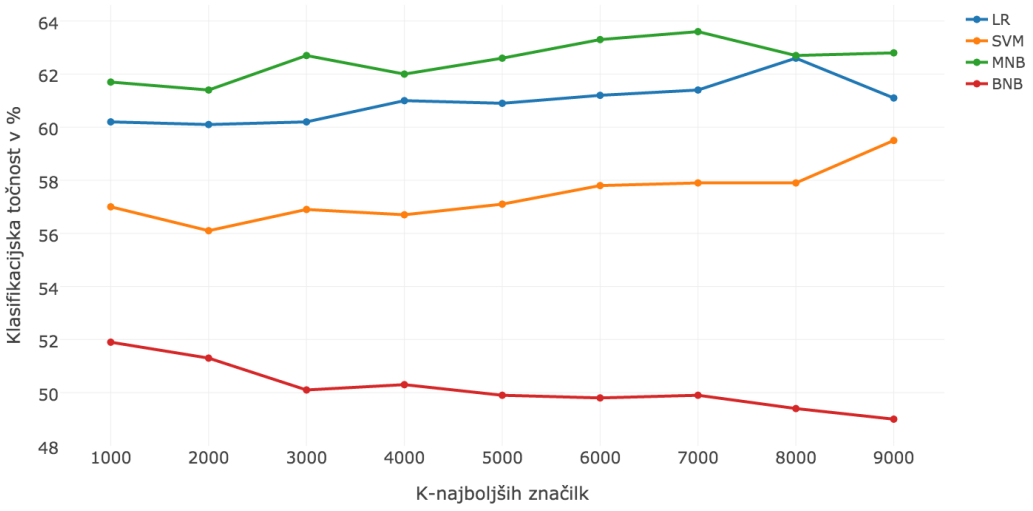
Tabela 5.12: Vpliv slovarjev sentimenta na uspešnost klasifikacije.

Nadalje nas je zanimal vpliv načina vektorizacije značilk. Pri tem smo preizkusili dvoje uteži, ki se pri klasifikaciji besedil največ uporabljajo ter preprostejšo vektorizacijo s štetjem in prisotnostjo značilk. Pang in sod. [17] so v raziskavi primerjali model na osnovi štetja ter prisotnosti značilk. V okviru klasifikacije sentimenta opisov filmov so najboljše rezultate dosegli z modelom prisotnosti unigramov. Smailović [13] je na primeru klasifikacije objav na mikroblogih sklenila, da se preprostejša utež *TF* obnese boljše od uteži *TF-IDF*. V tabeli 5.13 so prikazani rezultati različnih načinov vektorizacije značilk, kjer vidimo, da ima uporabljen način uteževanja opazen vpliv na posamezne metode strojnega učenja. Z uporabo uteži *TF-IDF* se je SVM približal ostalima klasifikatorjema. Klasifikator BNB smo izpustili zaradi definicije Bernoullijevega modela, ki ne upošteva števila pojavitev dogodka in so zato vrednosti med različnimi načini identične.

Vektorizacija značilk	LR	SVM	MNB
prisotnost (dvojiška vrednost)	**62,9**	60,2	62,1
pogostost (štetje)	61,8	59,7	**63,2**
utež TF	58,7	61,0	56,6
utež TF-IDF	61,7	**62,6**	61,3

Tabela 5.13: Primerjava načinov vektorizacije značilk.

Izbira značilk z metodo *Hi-kvadrat* je predstavljala osnovo za zadnji eksperiment. Želeli smo izmeriti, kakšen vpliv na klasifikacijo ima metoda *Hi-kvadrat* pri različno velikih K, kjer K predstavlja število najboljših značilk. Za K smo izbrali vrednosti med 1000 in 9000, s korakom po 1000 na modelu z unigrami. Rezultati so prikazani na sliki 5.1. Z metodo *Hi-kvadrat* smo pri BNB pridobili 3% CA. Pri drugih klasifikatorjih so bile izboljšave bistveno manj opazne. Kot zanimivost naj navedemo, da ima metoda pozitiven učinek tudi na klasifikacijo z višjimi *N-grami*. V navezi z bigrami smo pri BNB pridobili še dodaten odstotek pri CA. Očitno *Hi-kvadrat* dobro identificira širši kontekst pridobljen z bigrami.



Slika 5.1: Izbira značilk z metodo *Hi-kvadrat*.

5.4 Končni klasifikacijski model

S serijo eksperimentov smo merili vpliv predobdelave ter priprave značilk na uspešnost klasifikacije. V tabeli 5.14 so najboljše konfiguracije za posamezne metode strojnega učenja. Pri klasifikaciji uporabniških komentarjev v slovenskem jeziku se je najbolje obnesel klasifikator MNB (5.15). Glede na osnovno konfiguracijo smo uspeli uspešnost klasifikacije dvigniti za dobrih 10%. V primerjavi z vsakokratno izbiro večinskega razreda, pa je prirastek več kot 30%. Z ozirom na to, da smo delali na splošnem modelu klasifikacije uporabniških komentarjev, smo z rezultatom zadovoljni.

Na začetku poglavja smo omenili, da bomo najboljšo konfiguracijo preizkusili tudi na neuravnoteženemu korpusu. V slednjem je distribucija komentarjev po razredih naslednja: nevtralnih je 588, pozitivnih 898 ter negativnih 3291 komentarjev. Močno prevladujejo negativno nastrojeni komentarji. Če bi vse komentarje klasificirali v večinski razred, bi dobili 68,9% CA. V tabeli 5.16 so rezultati modela, ki smo ga zgradili na neuravnoteženem korpusu. CA se je precej izboljšala. Mera F_1 je visoka pri negativnih komentarjih in nizka

Konfiguracija	LR	SVM	MNB	BNB
izločanje značilk	unigrami	unigrami	unigrami	unigrami, bigrami
vektorizacija značilk	TF-IDF	TF-IDF	pogostost	TF-IDF
leksikalni viri	KSS	KSS, GIS, SWN	KSS, GIS, SWN	KSS, GIS
Hi-kvadrat (K-naj.)	8000	/	/	1000

Tabela 5.14: Najboljše konfiguracije po posameznih metodah strojnega učenja.

Klasifikator	CA	mera F_1			
		pos	*neg*	*neu*	povp.
osnova	54,6	59,0	55,2	48,8	54,3
LR	63,6	68,1	61,3	61,6	63,7
SVM	63,2	69,0	62,1	58,6	63,2
MNB	**65,5**	68,6	66,8	60,6	**65,3**
BNB	60,1	65,0	56,7	58,4	60,0

Tabela 5.15: Izbira najboljše metode strojnega učenja za klasifikacijo uporabniških komentarjev v slovenskem jeziku.

pri nevtralnih. Povprečna natančnost je bila 65,7%, povprečen priklic 55,5%. Glede na to, da gre pri uporabniških komentarjih po večini za subjektivno besedilo, se nizek priklic pri nevtralnem razredu ne zdi toliko problematičen.

CA	mera F_1			
	pos	*neg*	*neu*	povp.
76,2	60,0	85,4	29,4	58,3

Tabela 5.16: Uspešnost klasifikacije najboljše konfiguracije na neuravnoteženem korpusu.

Poglavje 6

Sklepne ugotovitve

V diplomskem delu smo obravnavali področje analize sentimenta s poudarkom na strojnem učenju. Seznanili smo se z razlogi, zakaj je to področje v zadnjem času med raziskovalci tako popularno ter zakaj podjetja vanj vlagajo veliko denarja. Za praktični cilj diplomske naloge smo si postavili izdelavo celovitega orodja za analizo sentimenta. Z orodjem smo zgradili slovar slovenskih sentimentnih besed ter označen korpus uporabniških komentarjev v slovenskem jeziku. Slovar in korpus sta med glavnimi rezultati tega dela. Na podlagi obstoječih raziskav sentimenta smo izbrali nekaj najpogosteje uporabljenih metod strojnega učenja ter jih preizkusili na označenem korpusu uporabniških komentarjev. S serijo eksperimentov smo ovrednotili različne metode predobdelave uporabniških komentarjev in različne klasifikatorje. Rezultati eksperimentov potrjujejo našo hipotezo, da je mogoče s predobdelavo besedila bistveno izboljšati klasifikacijo. Prav tako smo potrdili hipotezo, da lahko slovarji sentimenta pozitivno vplivajo na klasifikacijo neformalnih besedil.

Tako slovar sentimenta, korpus označenih komentarjev kot tudi sama klasifikacija niso brez pomankljivosti. Pri slovarju sentimenta lahko izpostavimo temeljno težavo tovrstnih slovarjev in sicer neupoštevanje konteksta uporabe. V različnih kontekstih ima lahko beseda drugačno sentimentno oceno. Druga pomankljivost se tiče načina izdelave slovarja. Ker smo slovar izdelali s preva-

janjem, bi bile dobrodošle dopolnitve z besedami in frazami, ki so specifične slovenskim besedilom. Jezik je živ organizem, zato bi bilo potrebno slovar tudi stalno dopolnjevati. Kar se korpusa komentarjev tiče, bi lahko še razširili nabor tem, ki smo jih izbrali za označevanje. Dosegli smo sicer zadovoljivo stopnjo strinjanja med označevalci, ki pa bi jo lahko z nekaj spremembami procesa označevanja še izboljšali, denimo sprotno spremljanje označevanja v začetni fazi ter odprava morebitnih nejasnosti skupaj z označevalci. Zaradi časovne stiske smo prezrli oblikoslovno označevanje, ki se v raziskavah o analizi sentimenta pogosto uporablja. Kot morebitne izboljšave klasifikacije lahko omenimo tudi vključitev konteksta povezav, ki se pojavijo v komentarjih ter kontekst novice, na katero se posamezen komentar nanaša. Izkoristili bi lahko tudi podporo v knjižnici scikit-learn, ki poišče optimalne vrednosti parametrov za klasifikacijo.

Analizi sentimenta v praktično-uporabne namene se v tem delu nismo posvetili, smo pa pripravili dobro osnovo za naprej. Ideje za nadaljne delo gre tako iskati predvsem v razširitvi orodja z analitskim delom, ki bi izkoristil vgrajeno podporo za luščenje komentarjev iz različnih spletnih portalov, jih klasificiral ter rezultate uporabil za, denimo, zaznavanje trendov, kot je padanje ali rast podpore opazovane politične stranke. Za nadaljne raziskave je zanimiv tudi izdelan slovar sentimentnih besed, sploh ker je tovrstnih virov v slovenskem jeziku malo. Slovar bi lahko uporabili kot osnovo za leksikalno analizo sentimenta. Zanimivo bi bilo tudi videti, kako se odreže pri klasifikaciji formalnih besedil.

Literatura

[1] M. Verlič, "Hibridni pristop za zaznavo elementov subjektivnosti v besedilnih tokovih", Doktorska disertacija, Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru, 2009.

[2] T. Bhuiyan, Y. Xu, A. Jøsang, "State-of-the-Art Review on Opinion Mining from Online Customers' Feedback' ", *Proceedings of the 9th Asia-Pacific Complex Systems Conference*, Chuo University, Tokyo, 2009.

[3] B. Liu. *Sentiment analysis and opinion mining.* Morgan & Claypool Publishers., 2012.

[4] L. Liu, M. Tamer Özsu. "Encyclopedia of Database Systems". V: B. Liu *Opinion Mining.* Springer US, pp. 1986–1990, 2009.

[5] L. Perković, "Primerjava Web 2.0 in Web 3.0", Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2011.

[6] B. Pang, L. Lee, "Opinion mining and sentiment analysis", *Foundations and Trends in Information Retrieval*, letnik 2, izdaja 1-2, 2008.

[7] G. Petz, M. Karpowicz, H. Fürschuß, A. Auinger, V. Stříteský, A. Holzinger. "Opinion Mining on the Web 2.0 – Characteristics of User Generated Content and Their Impacts". V: A. Holzinger, G. Pasi *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data.* Springer Berlin Heidelberg, pp. 35-46, 2013.

[8] T. Günther, "Sentiment Analysis of Microblogs", Thesis for the degree of Master in Language Technology, University of Gothenburg, 2013.

[9] B. Škoda, "Rudarjenje razpoloženja na komentarjih rtvslo.si", Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2013.

[10] R. Martinc, "Merjenje sentimenta na družabnem omrežju Twitter: izdelava orodja ter evaluacija", Magistrsko delo, Fakulteta za družbene vede, Univerza v Ljubljani, 2013.

[11] M. Volčanšek, "Leksikalna analiza razpoloženja za slovenska besedila", Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2015.

[12] M. Martinc, "Učinkovito procesiranje naravnega jezika s Pythonom", Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2015.

[13] J. Smailović, "Analiza sentimenta v tokovih kratkih spletnih sporočil", Doktorska disertacija, Mednarodna podiplomska šola Jožefa Stefana, Ljubljana, 2014.

[14] STA (2012) Kako merimo politični indeks kandidatov? [Online]. Dosegljivo: http://www.24ur.com/novice/slovenija/kako-merimo-politicni-indeks-kandidatov.html). [Dostopano 4. 1. 2016].

[15] Gama System® PerceptionAnalytics [Online]. Dosegljivo: http://www.gama-system.si/sl/produkti/gama-system-perceptionanalytics/. [Dostopano 4. 1. 2016].

[16] A. Ozgür, "Supervised and unsupervised machine learning techniques for text document categorization", Master Thesis, Boğaziçi University, Beşiktaş/İstanbul, 2004.

[17] B. Pang, L. Lee, S. Vaithyanathan. "Thumbs Up?: Sentiment Classification Using Machine Learning Techniques". V: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*. Association for Computational Linguistics, Stroudsburg, pp. 79–86, 2002.

[18] R. Munro (2013) Why is sentiment analysis hard? [Online]. Dosegljivo: http://idibon.com/why-is-sentiment-analysis-hard/. [Dostopano 21. 1. 2016].

[19] P. Trinkle, "An Introduction to Unsupervised Document Classification", University of Maryland Baltimore County, 2009.

[20] S. Bird, E. Klein, E. Loper. *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

[21] R. Bosch, "Sentiment Analysis: Incremental learning to build domain models", Master Thesis, Universitat Pompeu Fabra, Barcelona, 2004.

[22] A. McCallum, K. Nigam. "A comparison of event models for Naive Bayes text classification". V: *Proceedings AAAI-98 Workshop on Learning for Text Categorization*. AAAI Press, pp. 41–48, 1998.

[23] V. Vryniotis (2013) Machine Learning Tutorial: The Max Entropy Text Classifier. [Online]. Dosegljivo: http://blog.datumbox.com/machine-learning-tutorial-the-max-entropy-text-classifier/. [Dostopano 25. 1. 2016].

[24] A. Go, B. Richa, H. Lei. "Twitter Sentiment Classification using Distant Supervision". V: *CS224N Project Report*. Stanford 1, 2009.

[25] C. Potts (2011) Sentiment Symposium Tutorial: Tokenizing. [Online]. Dosegljivo: http://sentiment.christopherpotts.net/tokenizing.html/. [Dostopano 27. 1. 2016].

[26] M. Juršič, "Implementacija učinkovitega sistema za gradnjo, uporabo in evaluacijo lematizatorjev tipa RDR", Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2007.

[27] C.D. Manning, P. Raghavan, H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[28] Y. Yang, J.O. Pedersen. "A Comparative Study on Feature Selection in Text Categorization". V: *Proceedings of the Fourteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., pp. 412–420, 1997.

[29] B. Agarwal, N. Mittal. *Prominent Feature Extraction for Sentiment Analysis*. Springer, 2014.

[30] Python (programming language). [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Python_*programming*_*language*. [Dostopano 26. 12. 2015].

[31] J. Jackson. Python bumps off Java as top learning language. [Online]. Dosegljivo: http://www.computerworld.com/article/2489732/it-skills-training/python-bumps-off-java-as-top-learning-language.html. [Dostopano 26. 12. 2015].

[32] N. H. Tollervey. *Python in Education*. O'Reilly Media Inc., 2015.

[33] JavaScript. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/JavaScript. [Dostopano 26. 12. 2015].

[34] What is JavaScript? [Online]. Dosegljivo: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript. [Dostopano 26. 12. 2015].

[35] Natural Language Toolkit. [Online]. Dosegljivo: http://www.nltk.org. [Dostopano 29. 12. 2015].

[36] scikit-learn: Machine Learning in Python. [Online]. Dosegljivo: http://scikit-learn.org/stable/. [Dostopano 28. 12. 2015].

[37] M. Jurišič (2013) LemmaGen. [Online]. Dosegljivo: http://lemmatise.ijs.si. [Dostopano 20. 12. 2015].

[38] J. Virag (2013) Lemmagen slovenian lemmatizer available for python. [Online]. Dosegljivo: https://www.virag.si/2013/11/lemmagen-available-for-python/. [Dostopano 20. 12. 2015].

[39] Beautiful Soup. [Online]. Dosegljivo: http://www.crummy.com/software/BeautifulSoup/. [Dostopano 2. 1. 2016].

[40] API Client Library for Python. [Online]. Dosegljivo: https://developers.google.com/apiclientlibrary/python/start/get_started. [Dostopano 2. 1. 2016].

[41] Ajax (programming). [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Ajax_(programming). [Dostopano 27. 12. 2015].

[42] What is AJAX? [Online]. Dosegljivo: http://www.tutorialspoint.com/ajax/what_is_ajax.htm. [Dostopano 27. 12. 2015].

[43] Bootstrap (front-end framework). [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework). [Dostopano 27. 12. 2015].

[44] Računalništvo v oblaku. [Online]. Dosegljivo: https://sl.wikipedia.org/wiki/Računalništvo_v_oblaku. [Dostopano 28. 12. 2015].

[45] Cloud computing definition. [Online]. Dosegljivo: http://searchcloudcomputing.techtarget.com/definition/cloud-computing. [Dostopano 28. 12. 2015].

[46] J. Demojzes, "Tehnologija za masovni zajem in analizo podatkov iz družbenih omrežij ter njihova vizualizacija v realnem času", Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2013.

[47] SQL Azure. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/SQL_Azure. [Dostopano 28. 12. 2015].

[48] Introducing Visual Studio. [Online]. Dosegljivo: https://msdn.microsoft.com/en-us/library/fx6bk1f4(v=vs.90).aspx. [Dostopano 24. 12. 2015].

[49] Python Tools for Visual Studio. [Online]. Dosegljivo: https://visualstudiogallery.msdn.microsoft.com/9ea113de-a009-46cd-99f5-65ef0595f937. [Dostopano 24. 12. 2015].

[50] Use SQL Server Management Studio. [Online]. Dosegljivo: https://msdn.microsoft.com/en-us/library/ms174173.aspx. [Dostopano 23. 12. 2015].

[51] Team Foundation Server. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Team_Foundation_Server. [Dostopano 24. 12. 2015].

[52] M. Tilley (2012) Infinite Scrolling for AngularJS. [Online]. Dosegljivo: http://sroze.github.io/ngInfiniteScroll/images/diagram.png?1361649002. [Dostopano 9. 2. 2016].

[53] B. Liu, M. Hu (2004) Opinion Mining, Sentiment Analysis, and Opinion Spam Detection. [Online]. Dosegljivo: https://www.cs.uic.edu/ liub/FBS/sentiment-analysis.html#lexicon. [Dostopano 15. 2. 2016].

[54] R. Hurwitz (2002) General Inquirer. [Online]. Dosegljivo: http://www.wjh.harvard.edu/ inquirer/Home.html. [Dostopano 15. 2. 2016].

[55] MPQA Subjectivity Lexicon. [Online]. Dosegljivo: http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/. [Dostopano 15. 2. 2016].

[56] T. Wilson, J. Wiebe, P. Hoffmann. "Recognizing Contextual Polarity in Phrase-level Sentiment Analysis". V: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, USA, pp. 347–354, 2009.

[57] A. Esuli, F. Sebastiani (2010) SentiWordNet. [Online]. Dosegljivo: http://sentiwordnet.isti.cnr.it/. [Dostopano 15. 2. 2016].

[58] C. Potts (2011) Sentiment Symposium Tutorial: Lexicons. [Online]. Dosegljivo: http://sentiment.christopherpotts.net/lexicons.html. [Dostopano 11. 2. 2016].

[59] Cohen's kappa. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Cohen%27s_kappa. [Dostopano 19. 2. 2016].

[60] Fleiss' kappa. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Fleiss%27_kappa. [Dostopano 19. 2. 2016].

[61] I. Kononenko, M. Robnik-Šikonja. *Inteligentni sistemi*. Založba FE in FRI, 2010.

[62] A. Descoins (2013) Why accuracy alone is a bad measure for classification tasks, and what we can do about it. [Online]. Dosegljivo:

http://blog.tryolabs.com/2013/03/25/why-accuracy-alone-bad-measure-classification-tasks-and-what-we-can-do-about-it/. [Dostopano 25. 2. 2016].