Elektrotehnički fakultet Univerziteta u Beogradu

Katedra za računarsku tehniku i informatiku

Mašinska analiza sentimenta rečenica na srpskom jeziku

Master rad

Mentori:

Prof. dr Boško Nikolić

mr Bojan Furlan

Kandidat:

Nikola Milošević

Beograd, 2012.

Apstrakt

(Srpski)

U ovom radu je opisana implementacija sentiment analizatora za srpski jezik uz pomoć Naive Bayes algoritma mašinskog učenja. Kao deo sentiment analizatora opisan je i razvijen hibridni stemer za srpski jezik koji radi na principima uklanjanja sufiksa iz rečnika, a koji se može koristiti i za druge zadatke procesuiranja srpskog jezika. Analizator sentimenta radi binarnu klasifikaciju teksta u *dve klase - pozitivnu i negativnu*. Takođe, kako bi postigao veću preciznost sposoban je da obradi negacije pomoću dodavanja prefiksa rečima blizu negacije. Sentiment analizator i stemer su razvijeni kao web servis u programskom jeziku PHP, koristeći mySQL bazu podataka, koji sa korisničkom aplikacijom komunicira uz pomoj JSON objekata.

(English)

In this work is described implementation of sentiment analyzer for Serbian language using Naive Bayes algorhitm of machine learning. As a part of sentiment analyzer is described and developed hybrid stemmer for Serbian language that works on principles of suffix stripping and dictionary. Stemmer can also be used for other natural language processing task for Serbian language. Analyzer of sentiment does binary classification of text into two classes - positive and negative. Also, to assure great accuracy it is able to process negation by adding preffix to words near negation. Sentimen analyzer and stemmer are developed as web services in PHP programing language, using mySQL database and are communicating with user application using JSON objects.

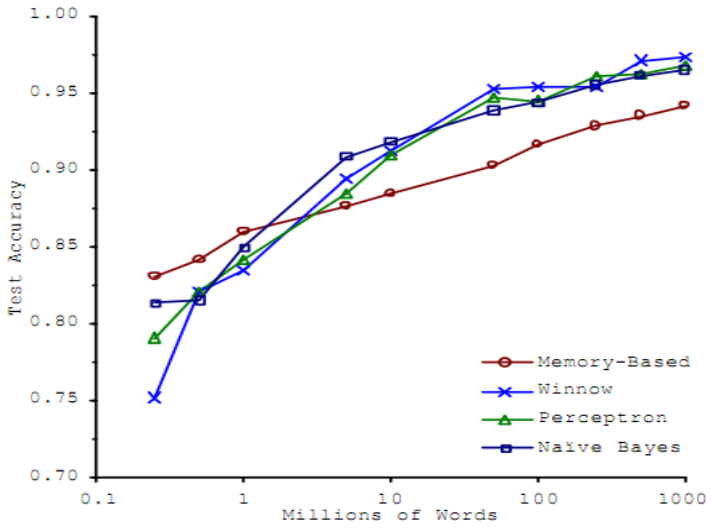
Sadržaj

I. **Uvod**

Klasifikacija teksta predstavlja jedan od osnovnih problema mašinskog procesuiranja čovekovog govora. Metode mašinske klasifikacije teksta se koriste u filterima neželjene pošte (spam filters), u prepoznavanju autora teksta, kao i njegovog pola i godišta, prepoznavanju teme teksta, analizi sentimenta rečenica. Klasifikacija teksta je našla i na široku primenu u pretraživačima interneta.

U ovom radu će biti akcenat stavljen na analizu sentimenta rečenica na srpskom jeziku.

Analiza sentimenta rečenica predstavlja specijalan slučaj problema klasifikacije teksta, jer sadrži samo dve klase. Analizom sentimenta rečenice se određuje da li je rečenica ili neki tekst imao pozitivnu ili negativnu konotaciju. Najčešće se koristi u analizi kvaliteta određenih proizvoda, odnosno, u mašinskom ocenjivanju proizvoda na osnovu komentara korisnika. Ovakav metod koriste naprimer Google Product Search i Bing Shopping. Postoje i aplikacije koje preuzimaju komentare sa Twitter-a ili Facebook-a o određenoj temi i određuju raspoloženje korisnika ovih društvenih mreža prema prikupljenim temama. Za analizu sentimenta se koristi nekoliko imena poput ekstrakcija mišljenja (opinion extraction, opinion mining), analiza subjektivnosti (subjectivity analysis, subjectivity mining, sentiment mining). Analiza sentimenta daje uvid u stavove ljudi o određenoj temi ili prozvodu. Na ovaj način se mogu predvideti rezultati izbora ili položaj proizvoda na tržištu.



Kako bi se rešio problem mašinske klasifikacije teksta upotrebljeni su algritmi supervizovanog mašinskog učenja. Iako postoji nekoliko veoma složenih algoritama supervizovanog mašinskog učenja, najbolje rezultate u analizi sentimenta i uopšte klasifikaciji

čovekovog jezika i govora je dao jedan od najprostijih – Naive Bayes. Ovaj algoritam već pri relativno malom broju primera na kojima je učio uspevao da postigne preciznost od oko 80% tačno predviđenih klasa teksta. Dok pri velikom broju primera, odnosno na trening setu od oko 100 miliona reči procenat tačno pogođenih klasa prelazi 90%.

Mašinska analiza sentimenta prepušta mašini, odnosno računaru labelisanje dokumenata, a nakon toga i sve ostale analize dobijenih labela. Na ovaj način se mogu gotovo u potpunosti automatizovati određene statističke analize. Često se, uz pomoć analize sentimenta, analizira i predviđa položaj određenog proizvoda na tržištu ili ishod izbora.

II. Problem i organizacija rada

Prilikom klasifikovanja teksta potrebno je određene dokumente svrstati u odgovarajuće klase. Problem možemo definisati tako, da ćemo imati dokument $d \in X$ gde je X prostor svih dokumenata i imamo fiksan skup klasa $C=\{c_1,c_2,...,c_n\}$. Tipično, prostor dokumenata X je neki višedimenzionalni prostor, dok su klase definisane od strane čoveka za potrebe aplikacije. Takođe imamo predhodno poznat trening skup D labelisanih dokumenata $<d, c>$, gde je $< d, c > \in X \times C$ [1]. Koristeći algoritam učenja nad trening skupom D, želimo da pronađemo funkciju γ takvu da ona mapira dokumente u klase (uključujući dokumente koji se ne nalaze u trening skupu):

$$\gamma: X \to C$$

Klasifikacija teksta i analiza sentimenta, kao specijalni slučaj klasifikacije, predstavlja problem koji se rešava uz pomoć veštačke inteligencije, odnosno uz pomoć algoritma mašinskog učenja. Kod analize sentimenta definisaćemo dve klase. Dokument može biti ili pozitivnog ili negativnog sentimenta. Na ovaj način suočavamo se sa problemom binarne klasifikacije, koja uz to ima određene jezičke specifičnosti, jer pokušavamo da klasifikujemo dokumente na osnovu osećanja koje nosi tekst na prirodnom jeziku.

Kako se radi o obradi jezika, potrebno je uraditi nekoliko obrada uz pomoć kojih se postižu bolji rezultati već na manjim skupovima trening podataka. Jedna od ovakvih obrada je steming (stemming), odnosno uklanjanje sufiksa fleksija tako da reči rezličite fleksije, a istog značenja nakon obrade imaju isti oblik. Radovi u oblasti steminga postoje od početka 1980., međutim samo nekoliko radova se osvrće na steming srpskog jezika. Nađeni stemeri za srpski jezik nisu se pokazali kao efikasni, a naročito je primećeno da kvalitetan steming srpskog jezika je moguće ostvariti i sa mnogo manjim brojem pravila za uklanjanje sufiksa od broja pravila koje poseduje trenutno postojeći stemer. Zbog toga implementiran je novi stemer.

Cilj rada je da se objasni implementacija analizatora sentimenta za srpski jezik. Međutim, kako analiza sentimenta podrazumeva steming i kako je za potrebe sentiment analize bio razvijen stemer, rad će pokriti i tu oblast. Treća glava rada opisuje proces steming-a, istoriju razvoja stemera i implementaciju stemera za srpski jezik. Četvrta glava se bavi analizom sentimenta i klasifikacijom teksta. U ovoj glavi je opisana upotreba analizatora sentimenta, istorija razvoja, naučni izazovi u ovoj oblasti, proces analize sentimenta, Naive Bayes algoritam mašinskog učenja i obrade potrebne da bi se dobili kvalitetni rezultati sentiment analize. Peta glava opisuje razvijeni web servis koji radi analizu sentimenta i stemovanje za srpski jezik. Šesta glava donosi zaključke, diskusiju i smernice za dalji razvoj kako razvojenog softvera, tako i oblasti analize sentimenta uopšte.

III. Stemmer za srpski jezik

III.1. Definicije i istorija

Prilikom procesuiranja prirodnog jezika potrebno je predhodno obraditi tekst na onaj način, na koji je to najprihvatljivije algoritmima za obradu jezika i na način na koji su ti algoritmi najefikasniji. Obrada teksta, pre izvršenja algoritma za procesuiranje jezika, naziva se *normalizacija*. Postoji nekoliko tehnika za normalizaciju teksta poput normalizacije korišćenjem ekvivalentnih klasa, obrade velikih i malih slova, steming i lematizacija. Kod analize sentimenta rečenica, kao i kod prikupljanja informacija (information retrieval) potrebno je da sve fleksije iste reči imaju pre obrade isti oblik, zbog čega su naročito bitne obrade steming i lematizacija.

Steming (Stemming) predstavlja heuristični proces otklanjanja krajeva reči u nadi da će se to postići u većini slučajeva korektno, a ponekad obuhvata i otklanjanje derivacionih afiksa. Cilj steminga je dobiti za sve fleksije reči sa istim značenjem najmanji zajednički sadržalac. Takođe, potrebno je obezbediti da ne dođe do preteranog otklanjanja sufiksa i na taj način da reč izgubi svoje osnovno značenje (overstemming). Reč koja nastane kao rezultat steminga naziva se stem.

Lematizacija (Lemmatization) se odnosi na svođenje reči na koren korišćenjem rečnika i morfologijskom analizom reči. Cilj lematizacije je da se odklone samo krajevi reči i da se vrati koren ili rečnička forma reči, poznata kao lemma[1].

Steming i lematizacija imaju široku upotrebu u mnogim zadacima obrade prirodnog jezika. Upotrebljavaju se kada je bitno mašinski odrediti semantiku i značenje različitih formi reči. Većina zadataka obrade prirodnog jezika se zasniva na mašinskom prepoznavanju značenja procesuiranog teksta, ali nisu zainteresovani za druge informacije koje daju različite forme reči, poput vremena, lica, konjugacije, deklinacije itd. Zadaci gde se koristi steming i lematizacija su prikupljanje informacija, mašinsko prevođenje, pretraga web-a ili dokumenata, klasifikacija teksta i sentiment analiza, kao specijalni slučaj klasifikacije teksta.

Predhodno je napomenuto da je cilj lematizatora da dobije koren reči pomoću semantičke analize ili rečnika. Koren reči je idealan za dalju obradu u algoritmima procesuiranja jezika. Kako bi se napravio uspešan lematizator potreban je rečnik sa svim formama svih reči nekog jezika. Očigledno, da bi se ovo postiglo, potrebno je mnogo vremena, kao i poprilično velika baza podataka. Jezik se često menja, rađaju se nove reči ili nove fleskije. Iz tog razloga je gotovo nemoguće napraviti idealan lematizator, tako da gotovo ne postoje razlike u performansama između stemera i lematizatora[2]. Kako stemer zahteva razvoj pravila, koje ne

prelaze nekoliko stotina, za jezike sa najvećom fleksijom, mnogo je jednostavnije razviti stemer.

Prvi ikad napisan algoritam za stemovanje je napisan od strane Julie Beth Lovins 1968. godine[3]. Najpopularniji steming algoritam za engleski jezik napisao je Martin Porter 1980. godine. Ovaj stemer je postao de-facto standard za stemovanje Engleskog jezika i Martin Porter je nagrađen Tony Kent Strix nagradom za svoj rad na polju steminga i prikupljanja informacija (information retrieval) 2000. godine. Do 2000. godine gotovo svi radovi koji su se odnosili na steming su razmatrali steming engleskog jezika, pošto je 60% stranica na internetu bilo na engleskom jeziku. Posle 2000. godine javila se potreba za razvojem stemera i za ostale jezike. Martin Porter je 2000. godine objavio rad na snowball radnom okruženju za kreiranje algoritama steminga i nekoliko steming algoritama za različite jezike[4]. Pravila stemovanja za Srpski jezik nisu do sad precizno definisani.

Martin Porter

III.2. Srpski jezik i srodni radovi

Srpski jezik je forma južnoslovenskog jezika (Indo-Evropskog), koji govore uglavnom Srbi. Svi južnoslovenski jezici vode poreklo od Staro Crkveno Slovenskog. Prethodnik srpskog jezika je starosrpski (Srpsko-slovenski), koji ima književnu istoriju od 10. veka.

Srpski jezik je jezik sa visokom fleksijom. Zbog ove karakteristike stemer za srpski jezik mora imati mnogo više pravila od stemera za jezike sa manjom fleksijom, kao što je engleski jezik. Porterov stemer ima 63 pravila, ali ukoliko ga predstavimo u formatu pravila za odklanjanje sufixa dobijamo broj od oko 120 pravila[5]. Stemer za srpski jezik imaće između 3 i 4 puta više pravila za otklanjanje sufiksa od stemera za engleski jezik zbog veće fleksije.

Kao što je predhodno navedeno, Lovins i Porter su napravili prve stemere. U poslednjih 10 godina bili su napravljeni stemeri za rezličite jezike poput nemačkog, francuskog, španskog, portugalskog, italijanskog, rumunskog, holandskog, danskog, švedskog, norveškog, ruskog, finskog, češkog, irskog, mađarskog, turskog, jermenskog, baskijskog, katalonskog koristeći snowball okruženje. Pomoću snowball okruženja napravljeni su stemeri za samo dva slovenska jezika - češki i ruski. Protraga za drugim stemerima slovenskih jezika dala je

ograničene rezultate. Postoji stemer za slovenački jezik[6], kao i za hrvatski jezik. Međutim na SVN repozitorijumu stemera za hrvatski jezik su nađena samo ograničena objašnjenja (http://svn.rot13.org/index.cgi/stem-hr).

Stemer za srpski jezik su napravili Vlade Kešelja i Danka Šipke i objavili u radu „A suffix subsumption-based approach to building stemmers and lemmatizers for highly inflectional languages with sparse resource"[5]. Autori u radu navode da njihov stemer ima preciznost od 79% u zadacima pronalaženja informacija (information retrieval). Istražujući rad stemera Vlade Kešelja i Danka Šipke utvrdio sam da ovaj stemer može biti dosta manuelno poboljšan, kao i da je moguće redukovati broj pravila. Kešelj i Šipka su napravili dva stemera, od kojih optimalni stemer ima 1000 pravila, dok drugi stemer ima 17000 pravila. Pravila stemera su razvijeni na osnovu algoritma nesupervizovanog mašinskog učenja koji je trebao da prepozna sufikse reči na osnovu datog teksta. Pravila dobijena na ovaj način su dobra polazna tačka, međutim algoritam daje i pogrešne i nepostojeće sufikse i pravila za njihovu obradu. Pravila, koje je pronalazio algoritam mašinskog učenja, su često dovodila do preteranog stemovanja (over-stemming), tako da su reči gubila svoj osnovni smisao. Takođe, određene reči su u različitim fleksijama stemovani na različite načine, što ne bi trebalo da bude slučaj sa dobrim stemerima. Utvrđeno je da je moguće redukovati broj pravila na oko 300, što će poboljšati performanse stemera. Takođe poboljšanjem stemera i uklanjanjem pogrešnih pravila i dodavanjem novih znatno je moguće poboljšati preciznost stemera za srpski jezik.

III.3. Algoritam i stemming pravila

Stemer aplikacija je dizajnirana kao web aplikacija. Koristi PHP skriptu i AJAX za interakciju između korisničkog interfejsa i skripte. Aplikacija ima dva tekst polja, jedan za unos teksta, dok drugi služi za prikaz rezultata. Obrada teksta se pokreće pritiskom na dugme „Stem".

Kada je korisnik pritisne dugme „Stem", prva akcija koja će se odigrati je transformisanje srpskih specijalnih karaktera - *č,ć,š,ž,đ* u *cx, cy, sx, zx* i *dx* respektivno. Ova obrada se obavlja na klijentskoj strani pomoću JavaScript-a, radi lakšeg prijema i obrade na serverskoj strani. Kada korisnički interfejs primi odgovor od servera, interfejs će vratiti specijalne karaktere nazad u prvobitnu formu i na taj način ih prikazati korisniku. Nakon ove transformacije klijentska aplikacija kreira POST zahtev sa tekstom koji je potrebno stemovati, kao parametar.

PHP skripta će uraditi nekoliko transformacija pre stemovanja, koje će uprostiti dalje procesuiranje. Skripta će sva velika slova promeniti u mala (case folding). Takođe, dodaće prazan karakter pre i posle znakova interpunkcije. Ovo će učiniti znakove interpunkcije posebnim tokenima. Nakon ovih obrata, izvršiće se tokenizacija, odnosno rečenica će biti pretvorena u niz reči (tokena).

Postoje dva pristupa steming algoritmima. Jedan je aritmetički, dok je drugi rečnički. Da bi se obezbedili dobri rezultati steminga za srpski jezik potrebno je primeniti hibridni pristup. Većinu reči je moguće stemovati uz pomoć algoritma za odkidanje sufiksa i niza pravila. Međutim nepravilne glagole, kao i neke druge nepravilne oblike reči moraju biti stemovane korišćenjem rečnika. Algoritam koji je korišćen, prvo proverava u rečniku da li data reč postoji u tom obliku, a ukoliko postoji, menja celu reč odgovarajućom formom. Ukoliko reč ne postoji u rečniku, algoritam proverava da li sufiks postoji u nizu pravila za otklanjanje sufiksa i ukoliko postoji, otklanja ga ili ga zamenjuje odgovarajućim.

Serverska PHP skripta koristi regularne izraze kako bi pronašla sufikse i otklonila ih, ili zamenila. Stemovi ne bi smeli imati manje od 2 karaktera, jer bi na taj način stemovi postali beskorisni, zbog toga što reč od 2 ili manje karaktera nema jasno značenje.

Sufiks pravila su organizovana kao rečnik u formatu ključ-vrednost. Sufiks je ključ, dok je njegova zamena vrednost u ovoj strukturi podataka. Većina vrednosti zamene su prazni stringovi, međutim kod nekih sufiksa potrebno je zameniti sufiks sa nekim drugim, manjim, kako bi se dobio stem ili osnova reči.

Razvoj pravila za stemovanje za srpski jezik je započet korišćenjem 1000 pravila iz stemera razvijenog od strane Vlade Kešelja i Danka Šipke [5] prilagođenih za PHP. Kako su ova pravila bila kreirana pomoću algoritma mašinskog učenja, kreirana su i neka pogrešna i suvišna pravila, pa je stemer imao previše pravila. Neka od pogrešnih pravila su sadržala karaktere *x* i *y* na svom početku (nisu sadržali ceo specijalni karakter, jer karakteri *x* i *y* se ne mogu naći u srpskom jeziku, već su deo koda za znakove *č,ć,š,đ* i *ž*). S obzirom da algoritam nije bio svestan kodiranja srpskih specijalnih karaktera, ova pravila su bila kreirana. Pravila koja su sadržala ova dva karaktera su bila, u mom radu, ili obrisana ili im je dodat odgovarajući karakter, kako bi se upotpunili.

Prvi korak u kreiranju pravila stemovanja je bilo uklanjanje i ispravljanje pogrešnih i nepotrebnih pravila. Na ovaj način se dobila lista od 180 pravila. Uklanjanje nepotrebnih pravila je bilo rađeno ručno. Uklonjena su pravila koja su počinjala sa *x* ili *y*, kao i pravila duža od 5 karaktera. Određena pravila su identifikovana kao pogrešno formirana, jer bi uklonila više karaktera nego što sufiks sadrži, ili bi uklonio i deo reči koji je predhodnim pravilima definisan za određenu grupu reči kao stem. Indentifikovano je da različite forme iste reči imaju različite stemove. Kako stemer treba da pravi iste stemove za sve forme iste reči, ovaj problem je morao biti rešen brisanjem pravila, koji su do ovoga dovodili i dodavanjem novih pravila, koji nisu bili otkriveni algoritmom mašinskog učenja opisanom u radu Kešelja i Šipke.

Dva pristupa su primenjena prilikom razvoja novih pravila za stemovanje.

Prvi pristup je uključivao konsultovanje gramatike srpskog jezika o formiranju reči i transformacijama reči. Određene grupe reči imaju pravila za formiranje sufiksa u različitim

fleksijama. Ovaj pristup je dosta doprineo dobrom stemovanju reči, međutim primećeno je, da su i dalje neke reči stemovane na pogrešan način.

Drugi pristup je bio da se manuelno identifikuju reči, koje nisu bile dobro stemovane korišćenjem pravila dobijenih na prvi način i kreirati pravila za te forme reči.

Pomoću gramatike srpskog jezika (prvi opisani pristup) su se mogli izdvojiti nastavci za različite padeže. Tako na primer za imenice prve vrste muškog roda nastavci mogu da se preuzmu iz sledeće tabele:

I vrsta	**muški rod jednine**				**muški rod množine**	
padež	*živo*			*neživo*	*živo*	*neživo*
nom.	učenik-Ø	Slavk-o	Pavl-e	prozor-Ø	učenic-i	prozor-i
gen.	učenik-a	Slavk-a	Pavl-a	prozor-a	učenika-a	prozor-a
dat.	učenik-u	Slavk-u	Pavl-u	prozor-u	učenic-ima	prozor-ima
akuz.	učenik-a	Slavk-a	Pavl-a	prozor-Ø	učenik-e	prozor-e
vok.	učenič-e	Slavk-o	Pavl-e	(prozor-e)	učenic-i	(prozor-i)
instr.	učenik-om	Slavk-om	Pavl-om	prozor-om	učenic-ima	prozor-ima
lok.	učenik-u	Slavk-u	Pavl-u	prozor-u	učenic-ima	prozor-ima

Prozor je imenica koja ima za sve padeže koren isti kao nominativ. Na ovaj način se mogu usvojiti za njega nastavci. Međutim komplikacija se pojavljuje kod reči učenik. Ona naime ima različit oblik u vokativu jednine i u nominativu, dativu, vokativu, instrumentalu i lokativu množine. Ona zapravo ima različito jedno slovo pre nastavka. U tom slučaju na nastavke za padeže se dodaje i razlika u reči. Stem učenika u gotovo svim padežima će biti *učeni*, pa će se na taj način pokriti svi svi padežni oblici u svim rodovima. Jedina razlika je nominativ, gde bi sufiks za uklanjanje trebao da bude u tom slučaju slovo k. Međutim ovo se ne radi, jer mnogo reči završava na k, a ne pripadaju prvoj vrsti imenica, pa bi takvo pravilo napravilo veliku štetu tim rečima. Određivanje pravila za imenice poput učenik su sprovedene manuelno (drugi način).

Pošto se imperfekat ne koristi više u modernom srpskom jeziku i zbog toga što su njegove forme mnogo drugačije od korena reči, imperfekat nije podržan u ovom stemeru.

Rečnik reči za zamenu je razmotren kao rešenje samo kada nije bilo drugog prihvatljivog rešenja za određene reči i transformacije. Pošto nepravilni glagoli poput *„biti”*, *„hteti”*, *„jesam”* i *„moći”* imaju različite forme osnove u različitim vremenima i licima, oni moraju da budu stemovani korišćenjem rečnika. Takođe, na još nekim glagolima je iskorišćen rečnik, poput glagola *„naći”*, *„doći”*, *„ići”*, *„otići”*, *„stići”* i *„vući”*. Rečnik ovih glagola je iskorišćen i kao pravila za zamenu sufiksa, pošto postoje mnogi glagoli koji sadrže ove glagole sa određenim prefiksom poput *„pronaći”, „prevući”, „dovući”* itd. Na ovaj način je omogućeno stemovanje glagola, koji imaju različite osnove u različitim vremenima.

III.4. Evaluacija i predlozi za nastavak razvoja

Evaluacija napravljenog stemera za srpski jezik je urađena koristeći tekst od 522 reči iz dnevnih novina „Politika”. Dve metode evaluacije su primenjene.

U prvoj metodi, tekst iz novina je bio manuelno stemovan, nakon čega je taj tekst upoređen sa izlazom stemera nad istim tekstom. Osoba, koja je manuelno stemovala tekst je bila upućena u razvoj stemera i pravila koja su upotrebljena, ali takođe je vodila računa o toma da izlaz bude logičan i da ne bude preteranog (over stemming - da se stemuje više nego što treba i da reč izgubi osnovno značenje) ili premalog (under stemming - da se ne otkloni ceo sufiks) stemovanja.

Druga metoda je obuhvatala mašinsko stemovanje, nakon čega je čovek čitao tekst. Stemovi su evaluirani na osnovu toga, da li je moguće iz stema jasno zaključiti originalno značenje (nema preteranog stemovanja) i da li stem zadovoljava sve morfologijske varijacije lemme (nema premalog stemovanja).

Obe metode stemovanja sadrže izvesnu dozu subjektivnosti. Evaluacija stemera se u većini slučajeva radi na osnovu tačnosti algoritma prikupljanja informacija (information retrieval). S obzirom da tema rada se ne odnosi na zadatak prikupljanja informacija, nije bilo moguće analizirati stemer u tom kontekstu. Rezultati sentiment analize nisu mogli da posluže u analizi stemera, s obzirom da sama sentiment analiza pri idealnim uslovima, sa idealnim stemerom i sa velikim trening skupom (od oko 100 000 reči) ima preciznost nešto veću od 90%. Takođe na tačnost sentiment analize utiču i drugi faktori, kao što je kvalitete trening skupa, broj i učestalost reči u trening skupu. Zbog toga su odabrana ove dve metode. Metod pregledanja stemova su koristili Kešelj i Šipka u evaluaciji svog stemera. Očekivano je da će kombinacija ove dve metode dati objektivne rezultate.

Oba kriterijuma evaluacije su pokazali jako dobre rezultate. U slučaju manuelnog stemovanja, kao što je opisano ranije, problem je u tome što osoba koja stemuje zna pravila stemovanja, ali ipak pokušava da bude neutralna. U ovom slučaju detektovana su 37 različita stema. Ovo znači da je preciznost stemera 92%. S obzirom da je tekst relativno mali, može se predpostaviti da bi bilo u većem tekstu malog odstupanja. Uočeni su problemi kod reči koje poseduju glasovne promene.

Najproblematičnija glasovna promena, koju stemer nije u mogućnosti da obradi je „nepostojano a". Postoji sličan problem i kod drugih glasovnih promena, poput jednačenja suglasnika po zvučnosti, pa tako nije moguće računarski zaključiti da npr. reči *„otac"* i *„oca"* potiču od istog korena. Određena pojavljivanja glasovnih promena je moguće obraditi uz pomoć stemera. Međutim, glasovne promene nastaju prilikom tvorbe reči i potrebno je znati koren reči da bi se detektovala glasovna promena. Stemer obrađuje fleksije reči bez tačnog poznavanja korena reči, zbog čega su glasovne promene problematične za korektnu obradu.

Pravila su kreirana na taj način da pokriju što više reči, a kako se glasovne promene javljaju u manjem broju reči, nije moguće algoritamski napraviti pravilo koje bi podržalo i reči sa glasovnom promenom i reči bez nje. U drugom metodu evaluacije, takođe su primećeni slični rezultati. Oko 90% reči je bilo stemovano na taj način da je originalno značenje moglo biti lako zaključeno, kao i da su ti stemovi pokrivali sve morfologijske transformacije reči. Problem se javio sa kratkim rečima, koji imaju 3-4 karaktera, a koji su nakon stemovanja imali samo dva. Ovi stemovi jesu pokrivali sve morfologijske transformacije, ali originalno značenje se u mnogim slučajevima teško izvlačilo. S obzirom da mnoge od ovih reči se mogu smatrati nepotrebnim u srpskom jeziku (stopwords) i mogu se ukloniti pre naredne obrade, te reči ne bi trebale da prave veliki problem.

Stemer opisan u ovom radu je jedan od najefikasnijih stemera za srpski jezik trenutno. Međutim, postoji mogućnost unapređenja. Postoje forme nekih reči koje nisu korektno stemovane, a koje se ne mogu izbaciti kao stopword-i. Takođe, ostaje problem glasovnih promena. Rešenje tog problema bi moglo biti kreiranje rečnika svih reči koje sadrže glasovne promenu i obrađivati ih zajedno sa nepravilnim glagolima. Međutim ovakav rečnik bi bio izuzetno velik.

Pošto je većina reči stemovana na korektan način (oko 90%), kreiran stemer će verovatno mnogo pomoći u rešavanju problema procesuiranja srpskog jezika. Trenutno ne postoji mnogo aplikacija procesuiranja prirodnog jezika za srpski jezik. Pošto je stemer sastavni deo većine aplikacija procesuiranja prirodnog jezika, postoji verovanje da će ovaj stemer pomoći u daljoj razvoji ove oblasti za srpski jezik.

Zbog toga što je ovaj stemer napravljen kao alat za predprocesuiranje teksta u sentiment analizi, mogu biti potrebna manja prisposobljavanja za druge zadatke procesuiranja jezika poput mašinskog prevođenja, pribavljanja informacija (information retrieval) i pretrage.

IV. Analiza sentimenta

IV.1. Uvod

„Šta drugi ljudi misle" je uvek važna informacija za sve ljude tokom procesa donošenja odluke. Mnogo pre ekspanzije interneta ljudi su tražili savet od prijatelja o različitim proizvodima ili uslugama. Prilikom glasanja na izborima, takođe vode se debate među prijatelima ko će za koga da glasa. Danas je čest izvor ovakvih informacija upravo internet. Internet je omogućio da se više ne oslanjamo samo na informacije koje dobijemo od ograničenog broja poznanika, već informacije možemo da dobijemo od znatno većeg broja ljudi koji su na internetu komentarisali određenu temu. Ne oslanjamo se više samo na ljude koje znamo ili poznate profesijonalce, već i na ljude za koje nikad nismo čuli. I naravno, sve više i više ljudi svoje mišljenje objavljuje i čine ih pristupačnim nepoznatim osobama preko interneta. Velika želja korisnika i zavisnost od online saveta i preporuka je razlog za ogroman interes za nove sisteme koji direktno obrađuju mišljenja. Sa ekspanzijom Web 2.0 platformi, kao što su blogovi, forumi, peer-to-peer mreže i različite socijalne mreže i mediji, korisnici su dobili jak alat za prikupljanje i deljenje iskustava i mišljenja, bilo pozitivnih ili negativnih, o nekom proizvodu ili servisu. Ova mišljenja i iskustva korisnika mogu da imaju enorman uticaj na oblikovanje mišljenja drugih korisnika - kao i na lojalnost brendu i odluku o kupovini. Kompanije mogu da odgovore na korisničke poruke kroz monitoring socijanih mreža i analize, tako što će modifikovati svoje marketinške poruke, bolje pozicionirati brend, razviti bolje produkte i obavljati druge aktivnosti [6].

Postoji ogroman potencijal obrade ovakvih podataka, koji može da obezbedi dodatnu vretnost kako korisnicima tako i firmama koje rade analize javnog mnjenja o nekom proizvodu ili usluzi. Naime, moguće je automatizovano pretražiti i sumirati mišljenja ljudi. Ovo može uštedeti mnogo resursa i vremena prilikom traženja ovakve vrste informacija na internetu. Mišljenja i iskustva korisnika se mogu obraditi na različite načine - mogu se pribavljati određene informacije od interesa (information extraction, information retrieval), određivati sentiment teksta (sentiment analysis), ili vršiti pretraga.

Analiza sentimenta je oblast mašinske obrade prirodnog jezika u kome se za određeni tekst određuje sentiment. Problem određivanja sentimenta rečenica predstavlja specijalni slučaj klasifikacije teksta. Analiza sentimenta je najčešće binarna klasifikacija, u kojoj se određuje da li je tekst pozitivan ili negativan. Međutim analiza sentimenta može imati i više klasa, odnosno nivoa (npr. 1-5 zvezdica ili različiti nivoi pozitivnosti, odnosno negativnosti). Analiza sentimenta ima određene specifične izazove, koji se ne javljaju u klasičnoj klasifikaciji teksta jer se u ovom slučaju radi o emotivnoj klasifikaciji.

Analiza sentimenta se pokazala kao izuzetno korisna za različita istraživanja tržišta, za sisteme za preporuku, obaveštajne sisteme, kao i za sisteme poslovne inteligencije.

IV.2. Istorija i definicije

Oblast analize sentimenta trenutno uživa veliku pažnju istraživačke zajednice, međutim, oblast postoji već duže vreme. Prvi rad u oblasti mašinske sentiment analize se pojavio 1979.[8]. Počev od 2001. godine je počela nagla ekspanzija oblasti, kao i radova na ovu temu, zbog uočenja velikog istraživačkog i komercijalnog potencijala. Faktori koji su uticali na ovu ekspanziju su:

- Unapređenje metoda mašinskog učenja u procesuiranju prirodnog jezika i pribavljanju informacija
- Dostupnost podataka na kojima algoritmi mašinskog učenja mogu da bugu trenirani, zbog velike količine podataka na internetu, kao i specifičnih sajtova sa recenzijama
- Shvatanje fascinantnih intelektualnih izazova i komercijalnih i obaveštajnih upotreba koje oblast nudi.

Iako je mašinska analiza sentimenta uobičajan naziv za ovu oblast, danas proizvođaći, praktičari i medije ovu oblast u povoju zovu različitim imenima kao što su *„brend monitoring”, „buzz monitoring”, „online antropologija”, „analiza tržišnog uticaja”, „conversation mining”, „izviđanje online potrošača”, „analiza i monitoring socijalih medija”, „opinion mining”* kao i već pomenutom *„analizom sentimenta”*.

Fokus oblasti je da reši problem računarske obrade mišljenja, sentimenta i subjektivnosti u tekstu, što je poznato pod imenima opinion mining, analiza sentimenta i analiza subjektivnosti.

Opininon mining je deo oblasti blizak Web pretrazi i prikupljanju informacija. Alat, koji radi opinion mining, procesuira set rezultata pretrage za dati termin, odnosno proizvod, generiše određeni broj atributa proizvoda (kvalitet, odlike) i agregira mišljenja o svakom od atributa (loš, meštovit, dobar)[8].

Sentiment je termin, koji se koristi za automatsku evaluaciju teksta i praćenje predviđanja rasuđivanja. Veliki broj radova stavlja u fokus analize sentimenta specifičnu aplikaciju klasifikovanja polarnosti recenzija (pozitivna ili negativna), što je dovelo do činjenice da neki autori sugerišu da fraza „sentiment analiza” referiše usko samo ovu specifičnu aplikaciju. Međutim, mnogi su konstruisali termin mnogo šire da obuhvata računarsku obradu mišljenja sentimenta i subjektivnosti u tekstu.

Analiza subjektivnosti je termin koji se koristi takođe za aplikacije klasifikacije, u kojima se klasifikuju dokumenti na dve klase po svojoj objektivnosti, odnosno subjektivnosti. Analiza

subjektivnosti se ogleda u analizi teksta, identifikaciji teksta zasnovanog na mišljenjima i razlikovanja istog od objektivnog jezika. Iako je bilo radova na temu analize subjektivnosti, ova tema nije postala glavna tema oblasti, već se fokus oblasti okrenuo analizi sentimenta i opinion mining-u, kao oblastima sa mnogo većim potencijalima upotrebe u realnim okruženjima.

IV.3. Upotreba analize sentimenta

Analizu sentimenta je moguće upotrebiti u mnogim situacijama i obogatiti korisničko iskustvo. Kao prva upotreba se nameću sistemi orijentisani na recenzije i mišljenja o određenom proizvodu, bilo da se radi o nekom filmu, tehničkom uređaju ili o nekom servisu. Sentiment analiza je našla svoju široku upotrebu na web sajtovima orijentisanim na recenzije i mišljenja korisnika. Teme, takođe, ne bi smeli biti ograničene samo na recenzije različitih proizvoda, već mogu uključiti i mišljenja o kandidatima za određenu poziciju, političke probleme i slično. Sumarizacija korisničkih recenzija je važan problem u ovoj oblasti. Takođe jedna od upotrebe sentiment analize, može biti ispravljanje grešaka u korisničkom ocenjivanju (jer postoje slučajvei kada korisnici očito slučajno selektuju nizak rejting iako su njihovi komentari krajnje pozitivni i obrnuto).

Analiza sentimenta i opinion mining sistemi mogu imati važnu ulogu kao tehnologije koje će omogućiti određene funkcionalnosti drugim sistemima. Jedna mogućnost je da se, u sistemima za preporuke (recommendation systems), implementira funkcijonalnost da sistem ne preporučuje stvari koje imaju mnogo negativnih komentara. Detekcija „plamenova" (zapaljivog i agnostičkog jezika) u e–pošti ili drugim tipovima komunikacije je sledeća moguća upotreba binarne klasifikacije i sentiment analize. U sistemima koji prikazuju reklame sa strane na sajtovima, poput *Google Ad Words* korisno je detektovati web stranice čiji sadržaj je senzitivan na postavljenje određenih reklama. Odnosno, bilo bi korisno za sofisticirane sisteme da ne postavljaju reklame o određenom proizvodu, ukoliko se sadržaj stranice izrazito negativno osvrće na taj ili sličan proizvod, kao i postaviti reklamu ukoliko se sadržaj strane pozitivno osvrće na proizvod. Odgovaranje na pitanja, takođe je oblast, gde sentiment analiza može da se pokaže kao korisna. Na primer, pitanja o mišljenju na određenu stvar mogu se posmatrati na drugačiji način. [10].

Oblast sentiment analize i opinion mininga je korisna i za različite tipove obaveštajnih aplikacija. Na primer, poslovna inteligencija je jedan od glavnih faktora korporativnog interesa za ovu oblast. Iako su podaci o konkurentnom proizvodu bitni (poput težine, brzine, izdržljivosti itd), odgovaranje na pitanje pada prodaje zahteva veće fokusiranje na lične utiske korisnika. Često, lični utisci mogu da oblikuju lojalnost određenom brendu i da utiču na donošenje odluka o kupovini, pa je jako bitno za kompanije da analiziraju ovakve podatke u procesu donošenja narednih poslovnih odluka. Sledeća bitna upotreba analize sentimenta i

opinion mining-a je u vladinim ustanovama, koje takođe treba da prate mišljenja građana i da reaguju na njih. Njima od velike pomoći može biti sumarizacija. Takođe mogu se monitorisati izvori negativne i neprijateljske komunikacije i nakon toga na odgovarajući način reagovati rešavanjem problema.

Kao što je dobro poznato, mišljenja su jako važna u politici. Pa tako sentiment analiza može biti upotrebljenja za ispitivanje javnog mnjenja.

Analiza sentimenta je tehnologija koja se pokazala posebno korisna za omogućavanje e–zakonodavstva, omogućujući automatsku analizu mišljenja, koje ljudi objavljuju o određenom pravilu ili regulativi koja je na javnoj raspravi[11].

Interakcija ove oblasti sa socijologijom obećavaju napredak. Mnoga socijološka istraživanja uključuju pitanja poput reakcija javnog mnjenja, koji slojevi društva su kako raspoloženi prema određenoj temi, ko će lakše prihvatiti određenu promenu ili informaciju.

Aplikacije u online marketingu, poslovnoj inteligenciji, obaveštajnom radu vlade, e-zakonodavstvo, ispitivanju javnog mnjenja, kao i socijologiji su pokazali ogroman potencijal iskorišćenosti oblasti, međutim javljaju se i nove oblasti gde će u budućnosti ova tema zauzeti mesto tehnologije koja omogućava mnoge napredne funkcijonalnosti i olakšava rad.

IV.4. Generalni izazovi u implementaciji sentiment analizatora

Tradicijalna klasifikacija teksta treba da klasifikuje dokumente po temama. Može postojati veliki broj mogućih kategorija, kao i definicija kategorija, koje mogu biti zavisne od korisnika ili aplikacije. Za dati problem, možemo da radimo sa ne više od dve klase (binarna klasifikacija), ali takođe i sa hiljadama klasa. Kada se govori o sentiment analizi, radi se o relativno malom broju klasa, često binarnoj kasifikaciji. Međutim, može biti i više klasa (recimo ocene od 1-10).

Klasifikacija teksta se često oslanja na ključne reči, koje se najčešće pominju u određenoj klasi. Kada je reč o analizi sentimenta, nije trivijalan zadatak ni za čoveka da odredi najbolji niz ključnih reči. Međutim ova činjenica ne znači da je problem analize sentimenta teži od kategorizacije dokumentata. Različite tehnike mašinskog učenja bazirane na unigram modelu (posmatraju se reči kao nezavisne) mogu da imaju veću preciznost od čoveka pri odabiru ključnih reči za klasifikaciju[12].

Ipak, nivo tačnosti u zadatku analize sentimenta često nije toliko dobar koliko bi neko očekivao. Postoji nekoliko razloga zašto je ovaj problem težak, iako se klase toliko razlikuju jedna od druge (pozitivna i negativna). Jedna stvar je da postoje rečenice koje imaju jako

pozitivan ili negativan sentiment, međutim nije moguće utvrditi pomoću neke ključne reči taj sentiment. Reči koje se pojaljuju u toj rečenici nisu izrazito pozitivne ili negativne, već se mogu javiti u bilo kom kontekstu. Generalno analiza sentimenta je prilično zavisna od konteksta i domena. Ista rečenica u zavisnosti od konteksta može imati različit sentiment. Na primer rečenica „Pročitajte knjigu", u tekstu recenzije knjige imaće izrazito pozitivan sentiment, međutim u kontekstu recenzije filma negativan sentiment.

Obrada negacija je bitna za analizu sentimenta i mišljenja. Kako se većina algoritama za obradu teksta oslanja na model vreće reči, gde su reči nezavisno posmatrane i analizirane, rečenice kao na primer „Volim ovu knjigu" u „Ne volim ovu knjigu" će biti jako bliske u ovakvim analizama. Ali jedan različit termin, negacija, će učiniti to da ove dve rečenice budu u suprostavljenim klasama. Ovakva situacija, da jedan termin toliko ključno utiče na klasifikaciju, ne postoji u klasičnom pribavljanju informacija (information retrieval). Moguće je obraditi negacije kao drugostepena svojstva, odnosno da se inicijalna rečenica obradi, ignorišući negacije, ali nakon toga da se ona pretvori u reprezentaciju koja bi bila svesna negacije. Alternativno rešenje je da se negacija enkoduje u reči koje je slede, pa tako da algoritmi vreće reči budu svesni razlika između rečenica sa negacijom i rečenica bez negacija. Tako je moguće dodati „*NE*" kao sufiks ili prefiks rečima koje se nalaze blizu negacije[13]. Međutim, postoji problem u rečenicama poput „Ne mogu da verujem kako je ovo dobro". Ova rečenica predstavlja pozitivnu rečenicu, iako sadrži negaciju. Problem negacije je mnogo veći i ukoliko se ozbiljno upušta u rešavanje ovog problema, potrebno je napraviti model detekcije negacija i dela teksta na koji se negacija odnosi. Međutim, ovakvim modelima dobijaju se samo mala poboljšanja i zato nisu praktična.

Sledeći izazov za analizu sentimenta su ironija i sarkazam. Ove stilske figure je teško mašinski detektovati. Čak i ukoliko rečenice ne sadrže ovakve retoričke alate, opet mogu sadržati određene reči koji će biti protumačeni kao ključne reči određene polarnosti, iako zapravo rečenica ima suprotnu polarnost. Primer takve rečenice bi mogao biti „Film izbegava sve klišee i predvidivosti koje se mogu naći u Holivudskim filmovima". Reč „izbegava" u ovom slučaju predstavlja jak neočekivani reverzer sentimenta, koji mnogi analizatori sentimenta neće biti u stanju da obrade na korektan način.

Uprkos mnogim izazovima dosadašnja istraživanja ove teme dala su poprilično dobre rezultate i analiza sentimenta je našla široku primenu i u komercijalnim proizvodima. Tačnost koju može da postigne analiza sentimenta je u rasponu od 80% do oko 95% za veoma sofisticirane sisteme. Zbog pomentutih problema ironije, sarkazma, obrade negacija i raznih drugih jezičkih alata koje otežavaju mašinsku detekciju sentimenta, nije moguće dobiti perfektnu tačnost. Analiza ovih jezičkih alata i načina njihove obrade su i dalje tema mnogih istraživanja.

IV.5. Algoritmi za analizu sentimenta

Kako je analiza sentimenta specijalan slučaj klasifikacije teksta, koriste se algoritmi, koji se koriste i za klasifikaciju teksta. Klasifikacija teksta se rešava uz pomoć algoritama supervizovanog mašinskog učenja.

U supervizovanom mašinskom učenju uvek moraju postojati labelisani trening podaci, na osnovu kojih algoritam uči kako da postupa kada vidi nove podatke, odnosno kako da ih klasifikuje. Svi algoritmi mašinskog učenja imaju dve faze. Prva faza je faza učenja, u kojoj algoritam na osnovu podataka za koje zna odgovor uči kako da klasifikuje. Druga faza je predviđanje. U toj fazi algoritam dobija novi, nepoznati tekst i na osnovu trening podataka i određene analize teksta predviđa kojoj klasi dati dokument pripada.

Nesupervizovano mašinsko učenje podrazumeva algoritam koji nema podatke sa tačnim odgovorima na kojima može da uči, već radi predviđanje bez predhodnog učenja. U slučaju nesupervizovanog učenja, algoritam pokušava da nađe određenu strukturu nelabelisanih podataka i na osnovu toga radi predviđanja.

Sentiment analiza je problem koji se rešava supervizovanim učenjem. Postoji nekoliko algoritama koji se mogu koristiti za analizu sentimenta, od kojih su najčešći Naive Bayes, metoda podržavajućih vektora (support vector machines) i klasifikacija maksimalnom entropijom (entropy classification).

Metoda podržavajućih vektora je algoritam pri kome se od tekstova kreiraju vektori, a na osnovu trening podataka se kreira hiperlinija u *n* dimenzijonom prostoru koja razdvaja klase. Od novih tekstova se takođe kreiraju vektori i u zavisnosti na kojoj strani hiperlinije se vektor nalazi, u tu klasu se svrstava.

Metoda maksimalne entropije koristi multinominalnu logičku regresiju, koja omogućava da bude više od dva ishoda. Algoritam uči iterativno težine osobina, maksimizirajući ih metodom maximum a posteriori. Ovaj način može biti prilično spor zbog iterativnosti, međutim ovaj metod ne predpostavlja nezavisnost osobina, kao što to rade algoritmi vreće reči.

Naive Bayes je prost i brz algoritam zasnovan na modelu vreće reči. U algoritmu Naive Bayes se postavlja hipoteza da su osobine (u slučaju teksta to su reči) nezavisne. Algoritam koristi Bayesovo pravilo. Iako je ovaj algoritam jedan od najprostijih i najprimitivnih algoritama mašinskog učenja, daje poprilično dobre rezultate. Takođe, daje jako dobre podatke već za male količine trening podataka, što nije slučaj za druge algoritme. Ova osobina ga čini najkorišćenijim algoritmom za klasifikaciju teksta.

Kako bi se implementirao analizator sentimenta za srpski jezik odabran je upravo Naive Bayes algoritam, zbog svoje osobine brzog učenja na malim količinama podataka, kao i zbog svoje jednostavnosti i brzine.

IV.6. Bayesovo pravilo i algoritam Naive Bayes

Da bi se napravio klasifikator, potrebno je napraviti odgovarajući model. Odgovarajuća apstrakcija za klasifikator je kondicijonalni model, koji se može zapisati u obliku:

$$p(C|F_1, \dots, F_n)$$

Odnosno, potrebno je izračunati verovatnoću klasne promenljive *C* pod uslovom da su date promenljive $F_1 \dots F_n$. Ovu verovatnoću nije teško izračunati za mali broj promenljivih osobina $(F_1 \dots F_n)$, ali je problematično izračunati za veliki broj promenljivih ili kad promenljiva može imati veliki broj vrednosti. U slučaju klasifikacije teksta, promenljive su zapravo reči, pa ih može biti jako mnogo. Sa tolikim brojem reči nije moguće izračunati ovu verovatnoću na standardan način pa se formula mora reformulisati. Za reformulaciju će biti upotrebljena **Bayesova teorema**:

$$p(C|F_1, \dots, F_n) = \frac{p(C)p(F_1, F_2, \dots, F_n|C)}{p(F_1, \dots, F_n)}$$

U praksi se koristi najčešće samo deljenik, jer delilac ne zavisi od klase *C* i zbog toga što su vrednosti osobina $F_1,\dots,F_n$ data, pa je verovatnoća $p(F_1,\dots,F_n)$ praktično konstantna.

Thomas Bayes (1701 – 1761)

Deljenik je u tom slučaju jednak modelu zajedničke verovatnoće:

$$p(C, F_1, \dots, F_n)$$

Ovo se može napisati u sledećem obliku koristeći lančano pravilo za ponovljene upotrebe definicije uslovne verovatnoće:

$$
\begin{aligned}
& p(C, F_1, \dots, F_n) \\
& \propto p(C)p(F_1, \dots, F_n) \\
& \propto p(C)p(F_1|C)p(F_2, \dots, F_n|C, F_1) \\
& \propto p(C)p(F_1|C)p(F_2|C, F_1)p(F_2, \dots, F_n|C, F_1, F_2) \\
& \propto p(C)p(F_1|C)p(F_2|C, F_1)p(F_3|C, F_1, F_2)p(F_4, \dots, F_n|C, F_1, F_2, F_3) \\
& \propto p(C)p(F_1|C)p(F_2|C, F_1)p(F_3|C, F_1, F_2) \dots p(F_n|C, F_1, F_2, F_3, \dots, F_{n-1})
\end{aligned}
$$

Prilikom klasifikacije u algoritmu Naive Bayes se uvodi predpsotavka uslovne nezavisnosti. Predpostavlja se da su osobine F_1,F_2,...,F_n uslovno nezavisne jedne od drugih. S obzirom da ovo zaista nije slučaj kada se radi o tekstu, to predstavlja naivnu predpostavku, pa je i algoritam dobio ime Naive Bayes. Po predpostavci imamo:

$$p(F_i|C, F_j) = p(F_i|C)$$

za $i \neq j$, pa se model može predstaviti na sledeći način:

$$
\begin{aligned}
& p(C, F_1, \dots, F_n) \propto p(C)p(F_1|C)p(F_2|C)p(F_3|C) \dots p(F_n|C) \\
& p(C, F_1, \dots, F_n) \propto p(C) \prod_{i=1}^{n} p(F_i|C)
\end{aligned}
$$

Dakle, po predpostavci nezavisnoti, uslovna distribucija nad klasom C može se napisati kao:

$$p(C, F_1, \dots, F_n) = \frac{1}{Z} p(C) \prod_{i=1}^{n} p(F_i|C)$$

gde je Z faktor skaliranja zavistan samo od F_1,...,F_n konstanta, ukoliko su vrednosti promenljivih poznate.

U klasifikaciji teksta, algoritam Naive Bayes tekst posmatra kao vreću sa rečima. Predpostavlja se da, za dovoljno dobru klasifikaciju, red reči u tekstu nije bitan.

Naive Bayes algoritam ima dve faze. Prva faza je faza učenja, u kojoj se algoritmu daju podaci, odnosno tekstovi, sa oznakom klase kojoj pripadaju. U ovom delu algoritam računa broj pojavljivanja svake reči u određenoj klasi, kao i koliko ukupno ima dokumenata u svakoj klasi i koliko ima ukupno dokumenata i reči. Na osnovu ovih brojki moguće je izračunati verovatnoće klase, kao i verovatnoće reči u određenoj klasi. Druga faza je faza predviđanja klase, gde se na osnovu brojeva iz trening skupa određuje klasa dokumenta. Određivanje klase se postiže na sledeći način:

Kao što je već pomenuto, potrebno je upotrebiti Bayeovo pravilo i primeniti ga na dokumente i klase:

$$p(c|d) = \frac{p(c)p(d|c)}{p(d)}$$

Gde je *c* klasa dokumenta, dok je *d* dati dokument. Kako bi se odredilo kojoj klasi pripada određeni dokument, potrebno je naći najveću vrednost verovatnoće $p(c_i|d_j)$:

$$\begin{aligned} c_{MAP} &= argmax_{\ c\in C}\ p(c|d) \\ &= argmax_{c\in C}\ \frac{p(c)p(d|c)}{p(d)} \\ &= argmax_{c\in C}\ p(c)p(d|c) \\ &= argmax_{c\in C}\ p(c)p(F_1, F_2, \dots, F_n|c) \end{aligned}$$

U ovom slučaju se delilac ignoriše, jer ne utiče na rezultat. Prilikom traženja maximuma verovatnoće klase, za sve klase delilac će biti verovatnoća pojavljivanja dokumenta. Kako sve klase imaju isti delilac, on se može radi uprošćavanja izuzeti. Verovatnoća pojavljivanja dokumenta *d* u klasi *c* jednaka je verovatnoći pojavljivanja svih reči dokumenta *d* u klasi *c*, što je predstavljeno u poslednjem redu jednačine.

Međutim u slučaju klasifikacije teksta nemamo mogućnost da pratimo verovatnoću celih dokumenata, jer se gotovo nikad neće pojaviti u istom obliku kao u trening skupu. Način na koji se mogu posmatrati novi dokumenti je da se posmatraju verovatnoće reči koje se pojavljuju u dokumentu. Ovaj način obrade dokumenata omogućava predpostavka da su reči i verovatnoće reči potpuno nezavisne. Upršćavanjem ovih izraza uzimajući u obzir predpostavke dolazi se do jednačina:

$$c_{MAP} = argmax_{c\in C}\ p(c) \prod_{i=1}^{n} p(F_i|c)$$

Navedenim formulama je prikazano, kako je moguće odrediti klasu dokumenta, ukoliko znamo verovatnoće pojavljivanja reči u određenoj klasi. Uz pomoć trening skupa moguće je izračunati verovatnoće pojavljivanja reči u određenoj klasi. Potrebno je obezbediti da trening podaci što bolje oslikavaju realnost. Verovatnoće klase i verovatnoće pojavljivanja određene reči u klasi je moguće dobiti uz pomoć frekvencija pojavljivanja klase i reči u datoj klasi u trening skupu na sledeći način:

$$p(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

$$p(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{w\in V} count(w, c_j)}$$

Verovatnoća klase $P(c_j)$ je jednaka broju dokumenata koji su u klasi c_j podeljena sa ukupnim brojem dokumenata u trening skupu. Verovatnoća pojavljivanja reči w_i u klasi c_j je jednaka frekvenciji pojavljivanja reci w_i u klasi c_j podeljena sa ukupnim brojem reči u klasi c_j.

Prilikom ovakvog računanja verovatnoća, javlja se problem, ukoliko se nakon treninga u test podacima pojavi neka nova reč, koja se pre nije pojavljivala u trening podacima. Ovakav slučaj imao bi za posledicu da zbirna verovatnoća $p(c|d)$ bude jednaka 0. Ovakav problem se rešava pomoću Laplasovog poravnanja (Laplace smoothing). Laplasovo poravnanje uvodi predpostavku da se nova reč pojavila i u trening skupu jedan put. Kako se ne bi mnogo poremetile verovatnoće ovakvom pretpostavkom, mora da se broj pojavljivanja svih reči uveća za 1. Formula za izračunavanje verovatnoće reči u klasi glasi:

$$p(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V}(count(w, c) + 1)}$$
$$= \frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}$$

Na ovaj način je postignuto da sve reči imaju određenu verovatnoću veću od 0. Takođe, verovatnoće za određene reči će se smanjiti, međutim, ipak će njihov odnos ostati isti.

Verovatnoće pojavljivanja reči u dokumentima su brojevi između 0 i 1. S obzorom da se radi o tekstu, koji sadrži jako veliki broj reči bez čestih ponavljanja, verovatnoće pojavljivanja reči su bliži 0. Zbog toga što se, ovi relativno mali brojevi, međusobno množe i na taj način se dobijaju još manji brojevi, dolazi u računarima koji klasifikuju tekst, do fenomena koji se zove floating point underflow. Odnosno, računar u nekom trenutku nije više u mogućnosti da čuva u memoriji broj veoma blizak nuli, tj. broj koji ima veliki broj 0 između zareza i decimale različite od 0, pa on biva zaokružen na 0. Na ovaj način bi došlo do greški jer za veće tekstove, većina rezultata bi bila 0. Da bi se ovo izbeglo, potrebno je logaritmovati izraz. Tako se od brojeva bliskih 0, dobijaju brojevi veći od 0, a da se pri tome odnos verovatnoća ne promeni. Pa konačan izraz, koji je korišćen prilikom klasifikacije teksta za verovatnoću pojavljivanja određene reči u nekoj klasi, glasi:

$$p(w_i|c) = \log\left(\frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}\right)$$
$$= \log(count(w_i, c) + 1) - log\left(\left(\sum_{w \in V} count(w, c)\right) + |V|\right)$$

Algoritam Naive Bayes se može napisati i u pseudo jeziku. Sledeće dve funkcije u pseudo jeziku odgovaraju fazama učenja i predviđanja:

TRAINMULTINOMIALNB(**C**,**D**)

1 $V \leftarrow$ EXTRACTVOCABULARY(**D**)
2 $N \leftarrow$ COUNTDOCS(**D**)
3 **for each** $c \in$ **C**
4 **do** $Nc \leftarrow$ COUNTDOCSINCLASS(**D**, c)
5 $prior[c] \leftarrow Nc/N$
6 $textc \leftarrow$ CONCATENATETEXTOFALLDOCSINCLASS(**D**, c)
7 **for each** $t \in V$
8 **do** $Tct \leftarrow$ COUNTTOKENSOFTERM($textc$, t)
9 **for each** $t \in V$
10 **do** $condprob[t][c] \leftarrow \frac{Tct+1}{\sum_{t'}(Tct'+1)}$
11 **return** V, $prior$, $condprob$

APPLYMULTINOMIALNB(**C**,V, $prior$, $condprob$, d)
1 $W \leftarrow$ EXTRACTTOKENSFROMDOC(V, d)
2 **for each** $c \in$ **C**
3 **do** $score[c] \leftarrow \log prior[c]$
4 **for each** $t \in W$
5 **do** $score[c]$ += log $condprob[t][c]$
6 **return** argmax$c\in$ **C** $score[c]$

[1]

Rad algoritma biće demonstriran i primerom na rečenici *„Jovan je dobar dečak.”*. Za ovu rečenicu i trening skup napravićeno sedeće predpostavke:

- Broj dokumenata u pozitivnoj i negativnoj klasi je jednak, odnosno $p(c_{pos}) = p(c_{neg}) = 0.5$
- Broj reči u klasama trening skupa je isti i iznosi 150 000, odnosno $(\sum_{w\in V} count(w,c)) + |V| = 150\,000$
- Pojavljivanje reči iz rečenice su dati u sledećoj tabeli:

Rečenica	Jovan	je	dobar	dečak
Pozitivno	0	17	8	0
Negativno	0	15	1	1

Na osnovu ovih podataka moguće je izračunati da li je rečenica pozitivna ili negativna. Formula koja se primenjuje je:

$$c_{MAP} = argmax_{c\in C}\, p(c) \prod_{i=1}^{n} p(F_i|c)$$

Za pozitivnu klasu vrednost se izračunava na osnovu formule:

$$c_{POS} = p(POS) \prod_{i=1}^{n} p(w_i|POS)$$

S obzirom da je p(POS) = 0.5 i da je

$$p(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V}(count(w, c) + 1)}$$
$$= \frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|} = p(w_i|c) = \log\left(\frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}\right)$$
$$= \log(count(w_i, c) + 1) - log\left(\left(\sum_{w \in V} count(w, c)\right) + |V|\right)$$

Pošto smo uveli predpostavku da je $\left((\sum_{w \in V} count(w, c)) + |V|\right) = 150\,000$, imamo sve promenljive su poznate, pa se može pristupiti računu:

$$c_{POS} = p(POS) \prod_{i=1}^{n} p(w_i|POS)$$
$$= \log\big(p(POS)\big)$$
$$+ \sum_{i=1}^{n} \log(count(w_i, POS) + 1) - log\left(\left(\sum_{w \in V} count(w, POS)\right) + |V|\right)$$
$$= \log\big(p(POS)\big) + \log(count(JOVAN, POS) + 1)$$
$$- log\left(\left(\sum_{w \in V} count(w, POS)\right) + |V|\right) + \log(count(JESAM, POS) + 1)$$
$$- log\left(\left(\sum_{w \in V} count(w, POS)\right) + |V|\right) + \log(count(DOBAR, POS) + 1)$$
$$- log\left(\left(\sum_{w \in V} count(w, POS)\right) + |V|\right) + \log\big(count\big(DEČAK, POS\big) + 1\big)$$
$$- log\left(\left(\sum_{w \in V} count(w, POS)\right) + |V|\right)$$
$$= \log(0.5) + \log(1) - \log(150\,000) + \log(18) - \log(150\,000) + \log(9)$$
$$- \log(150\,000) + \log(1) - \log(150\,000) = -18{,}79588$$

Na ovaj način je dobijena vrednost za pozitivnu klasu, ali da bi algoritam znao kojoj klasi rečenica pripada, mora da izračuna i vrednost za negativnu klasu.

$$
\begin{aligned}
c_{NEG} = p(NEG)\prod_{i=1}^{n} p(w_i|NEG) \\
&= \log\big(p(NEG)\big) \\
&+ \sum_{i=1}^{n} \log(count(w_i, NEG) + 1) - log\left(\left(\sum_{w\in V} count(w, NEG)\right) + |V|\right) \\
&= \log\big(p(NEG)\big) + \log(count(JOVAN, NEG) + 1) \\
&- log\left(\left(\sum_{w\in V} count(w, NEG)\right) + |V|\right) + \log(count(JESAM, NEG) + 1) \\
&- log\left(\left(\sum_{w\in V} count(w, NEG)\right) + |V|\right) + \log(count(DOBAR, NEG) + 1) \\
&- log\left(\left(\sum_{w\in V} count(w, NEG)\right) + |V|\right) + \log\big(count\big(DEČAK, NEG\big) + 1\big) \\
&- log\left(\left(\sum_{w\in V} count(w, NEG)\right) + |V|\right) \\
&= \log(0.5) + \log(1) - \log(150\,000) + \log(16) - \log(150\,000) + \log(2) \\
&- \log(150\,000) + \log(2) - \log(150\,000) = -19{,}199215
\end{aligned}
$$

Pošto vrednost za negativnu klasu (c_{NEG}) manja od vrednosti za pozitivnu klasu (c_{POS}), znači da rečenica pripada pozitivnoj klasi.

U ovom primeru je prikazan hipotetički trening skup, koji je formiran na osnovu predpostavki iz realnog sveta (glagol jesam se pojavljuje sličan broj puta i u pozitivnom i u negaitivnom kontekstu, konkretna lična imena se retko pojavljuju itd.) i na osnovu specifikacija trening skupa (broj reči i broj rečenica u oba konteksta su približno isti).

Kako bi se postigla veća preciznost algoritma pri sentiment analizi, potrebno je uvesti određene dodatne obrade.

Prva obrada koja poboljšava rezultate sentiment analize, ali i mnogih drugih zadataka procesuiranja jezika je svakako steming. Ovo je zapravo prva operacija koja se obavlja u cilju da reči istog značenja, a različitih fleksija budu u algoritmu zajedno poborojani.

Druga obrada je obrada negacija. Upotrebljen model je da se, rečima oko negacije, doda prefiks *„NE_”*. Prefiks se dodaje rečima nakon reči *„ne”* i negacije glagola jesam (nisam) i hteti (neću), sve do prvog znaka interpunkcije. Znak interpunkcije često je mesto gde negacija prestaje da važi i gde se javlja novi kontekst. Iako ovo nije savršen model, jer mogu postojati i

negacije koje se ne odnose na tekst do prvog znaka interpunkcije, već znatno kraće, ovaj model se pokazao kao dovoljno dobar. Mogli bi se koristiti i znatno komplikovaniji modeli, ali se pokazalo da bi doneli neznatna poboljšanja.

Reči, koje ne nose značenje i koje se često pojavljuju u tekstu je potrebno ukloniti. Ovakve reči se nazivaju stop-termini (Stopwords). Stop-termini se uklanjaju kako bi tekst koji se obrađuje sadržao suštinske reči od značaja za analizu sentimenta i kako se usled čestog pojavljivanja ovih reči u različitim kontekstima ne bi loše protumačio. Ovakve reči u srpskom jeziku su na primer: *da, onda, zato, što, ako, na, u, za...*

Naive Bayes algoritam predstavlja veoma brz algoritam mašinskog učenja, pogodan za klasifikaciju teksta koji ima male zahteve što se upotrebe memorije tiče. Robustan je za nerelevantne podatke, jer će se oni međusobno poništavati. Takođe, dobro se pokazao i u domenima, gde postoji veliki broj podjednako relevantnih podataka. Optimalan je ukoliko je tačna predpostavka nezavisnosti podataka. Pokazalo se da je dobra predpostavka nezavisnosti podataka u slučaju klasifikacije teksta, kao i za analizu sentimenta. Prilikom analize sentimenta je lako uočiti neke reči koje će doprineti dobrom klasifikovanju. Reči kao što su na primer *dobar, odličan, fantastičan*, *sjajan* će imati velike frekvencije u klasi pozitivnog sentimenta, dok će im verovatnoća pojavljivanja u dokumentima negativnog sentimenta biti jako blizu 0. Takođe reči poput *loš, grozan, razočaravajuć* će opet imati velike verovatnoće u dokumentima negativnog sentimenta, dok će imati verovatnoće blizu 0 u dokumentima pozivivnog sentimenta.

V. Realizovan softver

Aplikacija za analizu sentimenta je razvijena kao web aplikacija, kao web servis. Komunikacija između servisa i korisničke aplikacije se može ostvariti uz pomoć JSON (JavaScript Object Notation) formata za razmenu. Ideja je da se u kasnijoj fazi razvoja omogući pristup servisu i sa drugih uređaja (desktop aplikacija, mobilnih aplikacija itd.). Na ovaj način, aplikacija je dostupna svim korisnicima interneta. Pored web servisa, koji će pružati interfejs za rad sa aplikacijom, postojaće i web aplikacija, uz pomoć koje će biti moguće rukovanje sa aplikacijom.

Kao razvojni jezik izabran je PHP, zbog toga što predstavlja jedan od najzastupljenijih jezika, koji se izvršavaju na strani servera, na internetu. Takođe, jedan od razloga zašto je izabran baš PHP predstavlja i trenutna cena zakupa hostinga. PHP se takođe pokazao kao jezik koji ima najbolji odnos cene, funkcijonalnosti i lakoće razvoja.

Web aplikacija pruža dva servisa. Jedan je analiza sentimenta teksta, dok je drugi stemming zadatog teksta. Ovi zahtevi se mogu izvesti pomoću polja za unos teksta u web aplikaciji ili pomoću POST zahteva prema servisu iz neke druge aplikacije (mobilne aplikacije ili desktop aplikacije). POST zahtev treba da sadrži parametre u JSON formatu i to:

- Ukoliko se želi dodati rečenica kojom će se upotpuniti training podaci:

  {"Type" : "SentimentLearn","Pass":"MD5 hashovana lozinka i korisničko ime", "Sentence" : "primer teksta", "Sentiment" : "Positive/Negative" }

  Lozinka, prilikom slanja SentimentLearn zahteva, je potrebna kako bi se zabranilo da bilo koji korisnik utiče na algoritam za analizu sentimenta, dodajući pogrešne primere i tako remeteći statističke podatke o pojavljivanju određenih reči u klasama. Zbog toga je ova mogućnost ostavljena samo korisnicima koji znaju lozinku.

  Servis na ovu poruku odgovara POST porukom sa sledećim JSON parametrima:

  {"Type" : "SentimentLearned","Result":"OK\NOK"}

  Rezultat NOK će se slati i ukoliko zbog neke greške ne bude procesuiran tekst za učenje, kao i ukoliko su korisničko ime i lozinka netačni. Pored NOK rezultata, biće takođe i razlog, zbog kog poruka nije korektno procesuirana.

- Ukoliko se želi analizirati određeni tekst format POST parametra bi bio sledeći:

  {"Type" : "SentimentRequest","Sentence" : "Primer teksta ili rečenice"}.

Odgovor na ovaj zahtev bi bio poslat POST odgovor koji bi imao sledeće JSON parameter:

{"Type" : "SentimentResponse", "Result" : "Positive/Negative"}

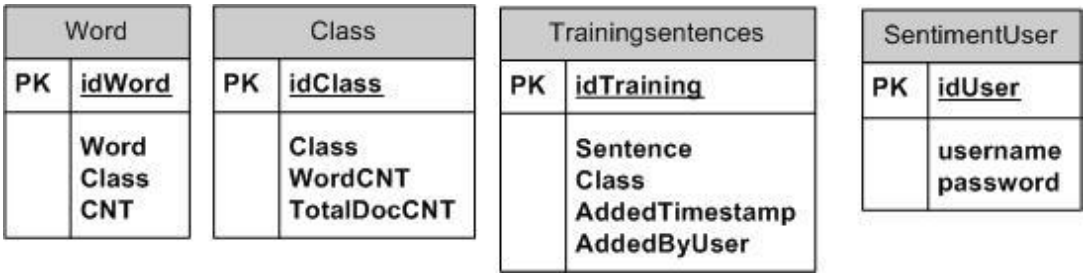
- Ukoliko se želi samo uraditi stemming teksta JSON zahtev bi izgledao na sledeći način:

  {"Type" : "StemmingRequest","Sentence":"Primer rečenice."}

  Na ovaj zahtev server bi odgovorio sledećom porukom:

  {"Type" : "StemmingResponse", "Sentence" : "Stemmovana rečenica"}

Algoritam za učenje će kao svoje skladište koristiti mySQL bazu podataka u koju upisuje i ažurira statističke podatke o rečima, dobijenim prilikom faze učenja Naive Bayes algoritma. Baza sadrži četiri tabele. Model baze je prikazan na slici:



Tabele u bazi aplikacije za analizu sentimenta

Najbitnije tabele su tabele Word i Class u kojima se čuvaju statistički podaci o tekstovima iz trening podataka.

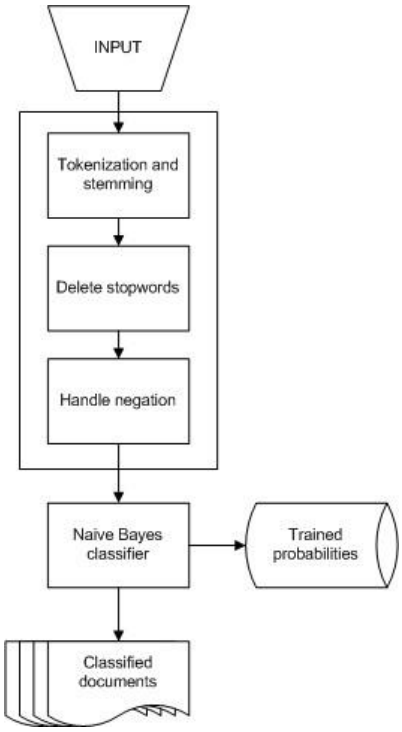
U tabeli Word se čuvaju podaci o broju pojavljivanja određene reči u određenoj klasi. Tabela ima tri atributa: reč, klasu i broj pojavljivanja reči u datoj klasi. U analizi sentimenta reč se može naći u obe klase, pa će i ovde biti dva unosa, jedan za pozitivan sentiment, i jedan za negativan. Brojevi za ključne reči, one koje bitno određuju sentiment bi trebalo bitno da se razlikuju za pozitivan i negativan sentiment. Reči, koje nisu ključne za određivanje sentimenta, bi trebalo u bazi da imaju relativno mali broj pojavljivanja u odnosu na reči koje određuju sentimenta ili da imaju sličan broj pojavljivanja za obe klase.

U tabeli Class se čuvaju podaci o klasi, odnosno broj reči koje su se pojavili u tekstovima određenog sentimenta i broj tekstova tog sentimenta. Takođe u ovoj tabeli se čuva i ukupan broj reči i ukupan broj dokumenata. Ova tabela će imati tri unosa, jedan za pozitivnu klasu, drugi za negativnu i treći unos za sumarne podatke.

Tabela Trainingsentences sadrži podatke o tekstovima koji su deo trening skupa. U ovoj tabeli se nalaze same rečenice nakon njihove obrade (stemming, obrada negacije), kao i podaci o klasi kojoj rečenica pripada, vremenu i datumu kad je rečenica unesena i podaci o korisniku koji je uneo rečenicu.

Poslednja tabela sadrži podatke o korisnicima koji mogu da kreiraju trening podatke. Ova tabela sadrži korisničko ime i lozinku. Lozinka se zbog sigurnosti čuva u hash formatu, odnosno nakon obrade od strane jednostrane funkcije. Takođe, je iskorišćen i salt na lozinci, odnosno nasumični niz karaktera koji dodatno treba da oteža proces otkrivanja lozinke, koji se dodaje lozinci pre obrade od strane jednostrane funkcije.

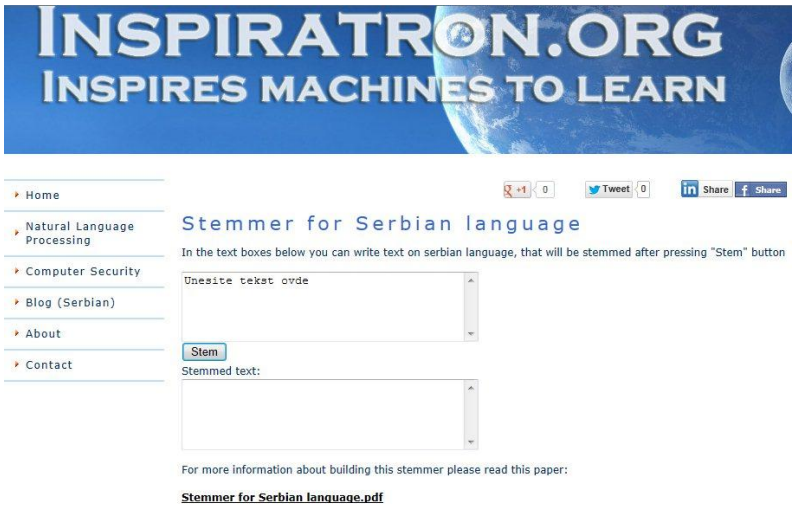
U slučaju da se od web servisa traži steming, rezultat će biti rečenica na kojoj je izvršen steming. Međutim ukoliko se traži analiza sentimenta, nekoliko operacija će biti izvršeni na datoj rečenici. Rečenicu je potrebno stemovati. Tokom steming-a obrađuju se i negacija tako što se na stemovane reči dodaje prefiks *NE_* ukoliko se oni nalaze iza negacije. Takođe, prilikom ove obrade, iz rečenice se izbacuju reči koje ne nose značenje, takozvani stop-termini. Nakon ove obrade, rečenica se pretvara u niz reči i na njoj se primenjuje Naive Bayes algoritam klasifikacije, kako bi se odredila klasa kojoj rečenici pripada, odnosno njen sentiment. Algoritam konsultuje bazu podataka, kako bi pribavio podatke o frekvencijama pojavljivanja određenih reči u klasi i računa verovatnoće da rečenica pripada pozitivnoj, odnosno negativnoj klasi. Upoređuje sračunate verovatnoće i kao rezultat vraća ime klase koja ima veću verovatnoću.



Algoritam obrade zahteva za analizu sentimenta

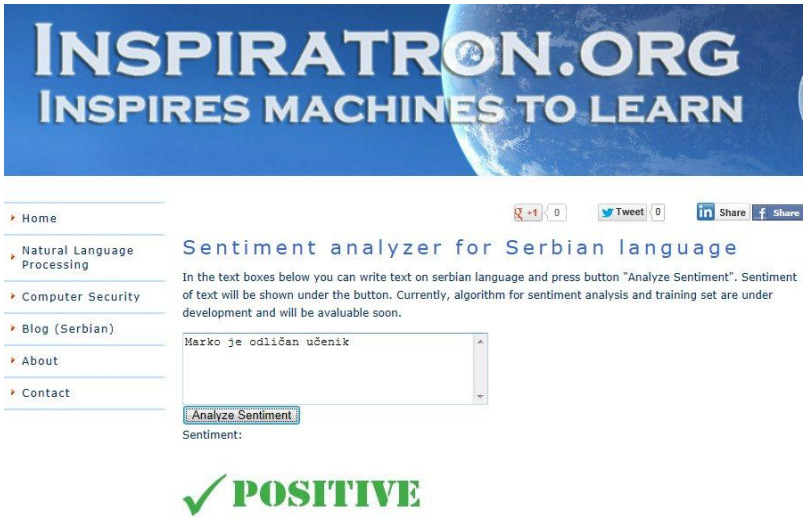
Korisnička web aplikacija služi za prezentaciju mogućnosti web servisa. Radi se o web aplikaciji čija logika je napisana u JavaScript programskom jeziku. Kako web servis omogućava dva servisa - stemovanje i analizu sentimenta, aplikacija ima dve web stranice koje omogućavaju demonstraciju ovih servisa.

Stranica za demonstraciju stemovanja sadrži dva polja za unos teksta i jedno dugme sa oznakom „Stem". Prvo polje za unos teksta namenjeno je da korisnik unese tekst koji želi da bude stemovan. Pritiskom na dugme „Stem" pomoću JavaScript-a i AJAX-a će biti poslat zahtev servisu na obradu i obrađeni tekst će se pojaviti u drugom polju.



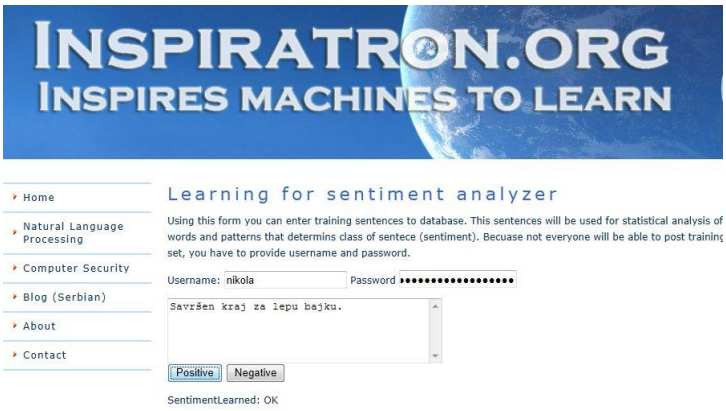
Izgled korisničkog interfejsa stranice za stemovanje

Druga stranica omogućava demonstraciju analizu sentimenta. Sastoji se od polja za unos teksta i dugmeta označenim sa „Analyze sentiment". Potrebno je korisnik da unese tekst u polje i da pritisne dugme, kako bi se tekst poslao na analizu. Rezultat analize će biti prikazan ispod labele na kojoj piše „Sentiment:". Rezultat može biti reč „Positive" ili „Negative".



Izgled korisničkog interfejsa za analizu sentimenta

Pored ove dve stranice za demonstraciju mogućnosti, postoji stranica za unos trening podataka. Ova stranica sadrži dva polja za unos korisničkog imena i lozinke, kao i jedno polje za unos teksta i dva dugmeta označena kao „Positive" i „Negative". Potrebno je da korisnik unese svoje korisničko ime i lozinku, nov tekst, koji alogoritam treba da nauči i uz pomoć dugmeta labeliše sentiment teksta. Pritiskom na dugme, pored labelisanja sentimenta unetog teksta, tekst se šalje serverskoj strani, koja proverava korisničke podatke, radi stemming, obradu negacija, uklanjanje stop-termina, obrađeni tekst unosi u bazu i ažurira statističke podatke. Ukoliko dođe do greške, korisničkoj aplikaciji se vraća poruka o grešci i ona se prikazuje. Ako obrada uspe, prikazuje se poruka o tome da je učenje uspelo. Do stranica za demonstraciju steminga i analize sentimenta je moguće doći pomoću navigacije na web aplikaciji. Međutim stranica za učenje je sakrivena kako bi se pojačala sigurnost dela aplikacije za učenje, jer se ne sme dozvoliti neovlašten unos trening podataka. Trening podatke je moguće dodati u bilo kom trenutku i na taj način poboljšati preciznost algoritma.



Korisnički interfejs stranice za treniranje algoritma

VI. Zaključak

U ovom radu prikazan je jedan način implementacije sentiment analize za srpski jezik.

Analiza sentimenta pomoću Naive Bayes algoritma se pokazala izuzetno dobra za srpski jezik, jer je dala jako kvalitetne rezultate i pri malim skupovima trening podataka. Takođe prikazan je i razvoj stemera za srpski jezik, koji takođe uspešno prevazilazi visoku fleskiju.

Alati koji koriste kao svoj deo ili kao svoju osnovu analizu sentimenta i stemer su raznovrsni. S obzirom da do sada nije implementirano mnogo takvih alata za srpski jezik, uprkos potrebi, ovaj rad će u znatnoj meri olakšati budući razvoj alata zasnovanih na sentiment analizi srpskog jezika.

Pored pristupa implementaciji sentiment analizatora Naive Bayes algoritmom, postoje i drugi pristupi. Mogu se upotrebiti algoritam podržavajućih vektora (support machine vectors), analiza uz pomoć entropije, Vinovljev algoritam (Winnow) i drugi. Naive Bayes se pri drugim jezicima pokazao kao algoritam koji daje jako dobre rezultate. Takođe, on se prevashodno koristi zbog svoje osobine da daje dobre rezultate i pri relativno malim skupovima trening podataka.

Uprkos velikoj tačnosti kako stemera, tako i sentiment analizatora, postoji prostor za unapređenje oba alata, kao i teme za dalja istraživanja. U implementaciji stemera za srpski jezik uočen je problem sa određenim glasovnim promenama. S obzirom da se glasovne promene javljaju prilikom tvorbe reči i predstavljaju nepravilnosti jezika, nije ih moguće obraditi na jednostavan način. Jedan od predloga bi mogao biti uvođenje rečnika za reči sa glasovnim promenama i stemovati na taj način te specijalne slučajeve. Međutim, ovakav rečnik bi bio prilično velik. Analiza sentimenta takođe poseduje potencijal za unapređenja uprkos kvalitetnim rezultatima. Povećanjem trening skupa, vremenom će se povećati i preciznost analizatora. Postoje određene stilske figure u srpskom jeziku koje za sad nije moguće obraditi na korektan način. *Ironija* i *metafora* su primeri takvih stilskih figura.

Ironija je stilska figura koja strukturom jednog sentimenta, zapravo govori u suprotnom sentimentu. Zbog ove osobine ironije, gotovo ju je nemoguće obraditi na korektan način. Kako bi se obradila ironija, morala bi postojati neka semantička analiza rečenice koja bi je detektovala i obradila na određeni način. Obrada stilskih figura, kako bi se dobilo određeno poboljšanje analize sentimenta, je takođe ostavljeno za dalja istraživanja.

Uprkos određenim problemima razvoja stemera i analizatora sentimenta postignuti su veoma zadovoljavajući rezultati u prepoznavanju sentimenta rečenica na srpskom jeziku. Ovi rezultati će uvećanjem trening skupa labelisanih rečenica se dodatno uvećati. Opisani problemi će se manifestovati u otprilike 10% rečenica i verovatno doprineti greškama koji

analizator sentimenta pravi. Ironija i druge stilske figure koje drastično utiču na analizu sentimenta nisu česte u srpskom jeziku. Greške pri stemovanju će stvoriti potrebu za većim trening skupom, jer većim trening skupom sentiment analizator će naučiti i reči, koje nisu korektno stemovane, da tretira na korektan način. Pri dovoljno velikom trening skupu samo stilske figure poput ironije ostaju problem.

Razvijeni stemer i sentiment analizator su implementirani sa ciljem da pomognu i da ojačaju oblast procesuiranje prirodnog srpskog jezika. Ipak srpsko tržište nije malo i postoji potreba za proizvodima koji će iskoristiti prednosti koje pruža procesuiranje prirodnog jezika. Budućnost donosi računare koji će razumeti ljudski izraz, pa je stoga neophodno da računari budu sposobni da razumeju, između ostalih i srpski jezik.

VII. Literatura

[1] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze , *An introduction to information retrieval*, Cambridge University press, 2009.

[2] Eija Airio, *Word normalization and decompounding in mono- and bi- lingual IR systems*, Departement of information studies, Tempere University, Finland, 2006.

[3] Julie Beth Lovins, *Development of stemming alghorthm*, Mechanical translation and computational linguistics, vol. 11, Massachusetts Institute of Technology, 1968.

[4] M. Porter, *Snowball framework*, internet: http://snowball.tartarus.org/ , Oct 21st .2012.

[5] Vlado Kešelj, Danko Šipka, *A suffix subsumption-based approach to building stemmers and lemmatizers for highly inflectional languages with sparse resource*, INFOTHECA, Journal of Informatics and Librarianship, vol. IX, 2008.

[6] Mirko Popović, Peter Willett. *The effectiveness of stemming for natural language access to Slovene textual data*. Journal of the American Society for Information Science, 43(5):384–390, 1992.

[7] J. Zabin and A. Jefferies, *Social media monitoring and analysis: Generating consumer insights from online conversation*, Aberdeen Group Benchmark Report, January 2008.

[8] J. Carbonell, *Subjective Understanding: Computer Models of Belief Systems*, PhD thesis, Yale, 1979.

[9] K. Dave, S. Lawrence, and D. M. Pennock, *Mining the peanut gallery: Opinion extraction and semantic classification of product reviews*, in Proceedings of WWW, pp. 519–528, 2003.

[10] Bo Pang and Lillian Lee *Sentiment analysis and opinion mining*, Foundations and Tredns in Information Retrieval, Volume 2 Issue 1-2, January 2008.

[11] S. Shulman, J. Callan, E. Hovy, and S. Zavestoski, *Language processing technologies for electronic rulemaking: A project highlight*, in *Proceedings of Digital Government Research (dg.o)*, pp. 87–88, 2005.

[12] B. Pang, L. Lee, and S. Vaithyanathan, *Thumbs up? Sentiment classification using machine learning techniques*, in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 79–86, 2002.

[13] S. Das and M. Chen, *Yahoo! for Amazon: Extracting market sentiment from stock message boards*, in Proceedings of the Asia Pacific Finance Association Annual Conference (APFA), 2001.