UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Marija Radović

KLASIFIKACIJA TEKSTA PREMA SENTIMENTU PRIMENOM METODA MAŠINSKOG UČENJA

master rad

Beograd, 2020.

Mentor:

dr Jelena Graovac, docent
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Mladen Nikolić, docent
Univerzitet u Beogradu, Matematički fakultet

dr Jovana Kovačević, docent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Porodici i dečku

Naslov master rada: Klasifikacija teksta prema sentimentu primenom metoda mašinskog učenja

Rezime: Živimo u vremenu eksponencijalnog rasta količine dostupnih informacija od kojih je većina zapisana u tekstualnom formatu. Prilikom donošenja odluka u poslovnom svetu, ali i u svakodnevnom životu, ljudi se često oslanjaju na mišljenje i iskustva drugih ljudi koji su svoje stavove javno izneli putem društvenih mreža, blogova, foruma i slično. U cilju efikasnijeg dobijanja informacija iz velike količine javno dostupnih informacija, korisno je izvršiti njihovu klasifikaciju prema sentimentu na klase dokumenata sa iskazanim pozitivnim, negativnim ili neutralnim stavom. Rešavanju ovog problema može se pristupiti primenom metoda zasnovanih na leksičkim resursima, primenom metoda mašinskog učenja ili hibridnim pristupom.

Cilj ovog master rada jeste rešavanje problema klasifikacije dokumenata prema sentimentu primenom metoda mašinskog učenja. Metode mašisnkog učenja koje će se koristiti u ovom radu su: metoda potpornih vektora, naivni Bajesov klasifikator, kombinacija metode potpornih vektora i naivnog Bajesovog klasifikatora i potpuno povezane neuronske mreže. Eksperimenti će biti izvršeni nad javno dostupnim korpusima filmskih recenzija na srpskom, engleskom i turskom jeziku. Rezultati eksperimenta će pokazati da je dobijena veća tačnost klasifikacije u odnosu na ranije objavljenje radove nad istim korpusima na engleskom i turskom jeziku.

Ključne reči: NLP, mašinsko učenje, SVM, NB, MLP, BOW, n-gram, TF, TF-IDF, Word2Vec

Sadržaj

1 Uvod **1**

2 Rešavanje problema metodama mašinskog učenja **4**
2.1 Pretprocesiranje teksta 5
2.2 Kreiranje vektorskog modela 7
2.3 Treniranje modela 8
2.4 Evaluacija modela 8
2.5 Poboljšanje modela 11

3 Vektorske reprezentacije teksta **12**
3.1 Vreća reči 13
3.2 N-gramski modeli 14
3.3 Vektor vrsta reči 16
3.4 Vektor pozicija reči u tekstu 18
3.5 TF model 19
3.6 TF-IDF model 19
3.7 Word2Vec model 20

4 Metode mašinskog učenja **23**
4.1 Metod potpornih vektora 24
4.2 Naivni Bajesov klasifikator 27
4.3 Naivni Bajesov klasifikator sa metodom potpornih vektora 29
4.4 Poptuno povezane neuronske mreže 29

5 Implementacija **35**

6 Rezultati **37**
6.1 Korpus na srpskom jeziku 37

6.2 Korpus na engleskom jeziku 39
6.3 Korpus na turskom jeziku 39
6.4 Empirijsko podešavanje i rezultati 40

7 Zaključak **43**

Bibliografija **45**

Glava 1

Uvod

Jedan od najvećih izazova današnjice u IT svetu jeste činjenica da se svakodnevno susrećemo sa velikom količinom dostupnih informacija koje pružaju velike mogućnosti, ali u isto vreme i zagušuju korisnike. Svakog dana stvori se preko 2,5 kvintiliona bajtova podataka sa stalnom tendencijom rasta [17]. Za 2020. godinu očekuje se da će se 1,7 megabajta podataka kreirati svake sekunde po svakoj osobi na planeti Zemlji. Prirodna je potreba da se te informacije nekako obrade i klasifikuju sa ciljem dobijanja skrivenog i korisnog znanja iz ogromne količine dostupnih dokumenata. **Obrada prirodnih jezika** (eng. natural language processing) je disciplina koja se bavi obradom teksta napisanog prirodnim jezikom i veoma je popularna u oblasti računarstva, pre svega zbog činjenice da je najveći deo svih dostupnih informacija zapisan upravo prirodnim jezikom u tekstualnom formatu. Digitalna revolucija je omogućila čuvanje dokumenata u elektronskom obliku, što je naročito značajno iz ugla optimizacije i automatizacije procesa predstavljanja dokumenata, njihove obrade kao i klasifikacije.

Jasno je da dokumenta možemo razvrstati po temi o kojoj pišu. To je instinktivna podela. Takođe, dokumenti se mogu klasifikovati i po autoru teksta, datumu objavljivanja, itd. Međutim, javlja se potreba za drugačijim vidom klasifikacije - **klasifikacijom prema sentimentu**. Svesni smo da društvene mreže predstavljaju deo života jednog modernog čoveka, kome je bitno mišljenje drugih. Ljudi često posećuju sajtove kao što su forumi, blogovi, društvene mreže. Sa jedne strane to im daje mogućnost za iskazivanje svojih ličnih stavova, a sa druge strane to im omogućava pristup mišljenju i stavovima drugih ljudi na neku temu, što može značajno doprineti donošenju ispravnih, ali i pogrešnih odluka u poslovnom svetu i u svakodnevnom životu.

Analiza teksta prema sentimentu se odnosi na obradu i klasifikaciju dokumenata na ona sa pozitivnim, negativnim i neutralnim sadržajem. Podela se može proširiti sa par dodatnih kategorija, kao npr. jako pozitivan ili jako negativan tekst. Možemo govoriti i o analizi na nivou dokumenta, na nivou rečenice ili pak na nivou sintagme. U ovom radu će fokus biti na osnovnom skupu klasifikacije na nivou dokumenta - pozitivni, negativni i neutralni dokumenti. Analiza teksta prema sentimentu se u opštem slučaju može definisati kao interpretacija i klasifikacija emocija. Tako se, na primer, za pozitivne tekstove mogu vezati emocije kao što su sreća, zadovoljstvo, ljubav, itd. Negativnim tekstovima možemo pripisati osećanja kao što su mržnja, bes i slično. U neutralnim tekstovima možemo naći nesigurnost, neodlučnost i druge osobine.

Potreba za analizom teksta prema sentimentu postoji kod različitih vrsta dokumenata. To mogu biti imejlovi, ankete, različite vrste recenzija i slično. Pored toga, nije jasno definisana forma pisanja ovih dokumenata. Informacije na taj način nemaju jasno definisanu strukturu pisanja. Usled kompleksnosti prirodnog jezika, računarska obrada teksta ne može dati stoprocentnu tačnost. Čovek će lako shvatiti ukoliko se u tekstu javlja sarkazam, ironija i slično, dok će računar vrlo verovatno „shvatiti" bukvalno ono što piše. Pored toga, reči nekada gube smisao kada se posmatraju van konteksta. Na primer, posmatrajmo rečenice „Ovaj film **ubija** koliko je dobar" i „U filmu policajac **ubija** nedužnu ženu". U prvoj rečenici reč **ubijati** ima pozitivnu konotaciju, dok u drugoj negativnu. Takođe, tekstovi mogu biti puni žargonskih izraza, višeznačnih reči i slično. Sve ovo čini zadatak analize prema sentimentu izazovnijim.

Ovakva vrsta analize teksta je vremenski skup proces ako je obavlja čovek. Zato ima smisla da se ovaj posao prosledi savremenim računarima. Time se dobija mogućnost korišćenja ovog procesa u realnom vremenu. Zamislimo situaciju da nezadovoljni kupac želi da se osveti proizvođaču tako što će na sajtu ostaviti neki zlonamerni komentar. To bi se moglo detektovati automatskom analizom svih komentara ostavljenih na sajtu u realnom vremenu. Ili, na primer, ukoliko se u svetu dešava neka kriza, to bi se, opet, na vreme moglo detektovati analizom najnovijih tekstualnih postova sa društvenih mreža.

Analiza teksta prema sentimentu ima dosta praktičnih primena. Ona je veoma korisna prilikom donošenja važnih odluka u okviru poslovnih preduzeća, raznih organizacija, ali i za samog pojedinca. Rezultati obrade i klasifikacije teksta prema sentimentu mogu dati ocenu o marketinškoj analizi ili o kvalitetu nekog proizvo-

da, preporuku knjige, filma, proizvoda i drugo. Inspiracija o značaju analize teksta prema sentimentu je dobijena iz [14] i [15].

Ovaj rad je podeljen na sedam delova. Nakon uvodnog poglavlja u kome je predstavljen značaj problema analize teksta prema sentimentu, u okviru drugog poglavlja će biti predstavljen opis celokupnog procesa rešavanja ovog problema - od pretprocesiranja teksta do primene metoda mašinskog učenja. U poglavlju 3 biće detaljno opisani modeli reprezentacije teksta, dok će u poglavlju 4 biti prikazane matematičke osnove metoda mašinskog učenja koje su korišćene u radu. Nakon toga biće reči o samoj implementaciji i rezultatima praktičnog rada, da bi na kraju rada bili sumirani utisci o celokupnom iznetom radu i dobijenim rezultatima.

Glava 2

Rešavanje problema metodama mašinskog učenja

Kao što je pomenuto u prethodnoj glavi, u cilju što efikasnijeg dobijanja znanja iz tekstualnih dokumenata, veoma je korisno prvo izvršiti njihovu klasifikaciju prema sentimentu. Generalno, postoje dva osnovna pristupa u klasifikaciji teksta - **nenadgledano** i **nadgledano učenje**. Kod nenadgledanog učenja nemamo dostupne labelirane podatke, odnosno, algoritam treba sâm da uoči neke zakonitosti u podacima koji su mu dati na ulazu, bez uvida u to šta treba da dobije na izlazu. Pri rešavanju problema primenom ovakvog pristupa obično se zahteva primena leksičkih i semantičkih resursa, kao i nekih drugih eksternih baza znanja. U slučaju nadgledanog učenja, podrazumeva se dostupnost labeliranih podataka na osnovu kojih algoritam treba za podatke na ulazu da „nauči" kako da dobije željeni izlaz. Ideja je da se tako naučeni model primeni na nove ulazne podatke sa ciljem predviđanja izlaza sa što većom preciznošću. Ovaj pristup predstavlja rešavanje problema primenom metoda mašinskog učenja i on će biti centar razmatranja ovog rada. Prilikom pretprocesiranja i pripreme teksta koristiće se metode kao što su stemovanje, lematizacija, uklanjanje stop-reči i slično. Tako pripremljen tekst se prevodi u vektorsku reprezentaciju teksta koji se potom prosleđuje kao ulaz u neki od klasifikatora zasnovanih na metodama mašinskog učenja.

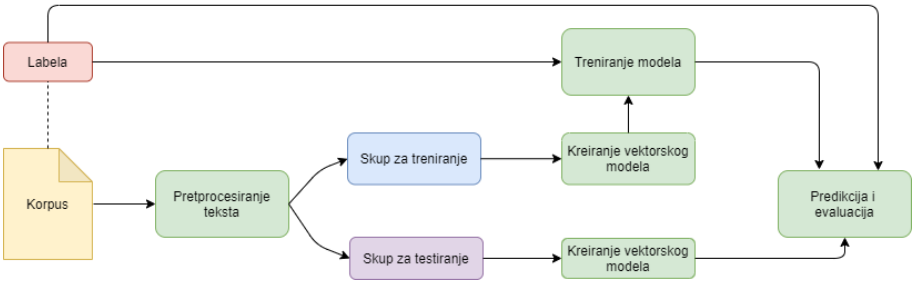
Pojam „veštačka inteligencija" danas se sve češće pominje, a odnosi se na simuliranje ljudske inteligencije pomoću mašina koje su isprogramirane da razmišljaju kao ljudi i oponašaju ljudske postupke. Jedna od primena veštačke inteligencije je **mašinsko učenje**. Vezuje se za program koji se može „naučiti" kako da se ponaša u novim situacijama na osnovu unapred zadatih situacija. Grubo rečeno, naš program

ćemo istrenirati, tj. naučiti koja načela u odlučivanju o polaritetu dokumenta treba da primenjuje. Kako to možemo ovde primeniti? Cilj nam je da napravimo model od dostupnih podataka i da kasnije budemo u stanju da damo sud o novim podacima na osnovu napravljenog modela. Ovakva vrsta učenja podrazumeva snabdevanje algoritma velikom količinom informacija tako da se sâm algoritam može neprekidno prilagođavati i usavršavati. Kao ulaz u klasifikator se prosleđuju labelirani podaci na osnovu kojih algoritam uparuje ono što je bilo na ulazu sa onim što treba da bude na izlazu i na taj način kreira obrasce pomoću kojih će kasnije moći da odlučuje.

Celokupni proces za rešavanje problema obuhvata nekoliko faza koje treba ispoštovati:

- pretprocesiranje teksta,
- kreiranje vektorskog modela,
- treniranje modela,
- evaluacija modela.

U nastavku ćemo detaljnije objasniti svaki od koraka. Biće podrazumevano da imamo labeliran skup podataka. Pomenuti proces se može ilustrovati Slikom 2.1.



Slika 2.1: Proces klasifikacije dokumenata primenom metoda mašinskog učenja

2.1 Pretprocesiranje teksta

Korak pretprocesiranja je važan u celokupnom procesu klasifikacije teksta prema sentimentu i može znatno uticati na tačnost klasifikacije. Pretprocesiranje nije obavezno, ali poboljšava performanse celog procesa. Ulaz u proces pretprocesiranja je sirovi tekst, koji najpre treba obraditi. Neke metode koje se koriste u fazi pretprocesiranja su:

- uklanjanje stop-reči,
- stemovanje,
- lematizacija,
- prevođenje velikih slova u mala,
- uklanjanje šuma,
- normalizacija teksta.

Objasnićemo ukratko pomenute metode. Stop-reči su učestale reči u jeziku, kao npr: „i", „da", „li", itd. u srpskom jeziku ili „a", „the", „is", „are", itd. u engleskom jeziku. Motivacija za uklanjanjem stop-reči leži u tome da uklanjanjem nisko relevantnih reči iz teksta fokus bude na važnim rečima. Procenjuje se da stop-reči čine 20-30% reči (bilo kog) korpusa. Stemovanje reči podrazumeva uklanjanje sufiksa reči, odnosno preslikavanje reči na koren reči. Npr. reči „učim", „učiš", „učimo", „učite" se sve prevode u oblik „uči". Ovaj postupak pomaže u redukovanju problema retkih matrica[1] (eng. sparse matrix). Lematizacija je slična stemovanju, ali za razliku od stemovanja gde se sufiksi samo uklone, lematizacija preslikava reč na kanonski oblik reči. Npr. reči „učim", „učiš", „učimo", „učite" se sve prevode u oblik „učiti". Stemovanjem i lematizacijom se reči koje imaju isto značenje, a zapisane su u drugačijem obliku (npr. drugi rod, padež, lice) prevode u isti oblik. Iako lematizacija donekle pravilnije prevodi reči u odnosu na stemovanje, oba postupka daju slične performanse u smislu tačnosti klasifikacije. Prevođenje velikih slova u mala je postupak koji se često previdi, iako je jednostavan i jako značajan, naročito nad malim skupom podataka. Uklanjanje šuma je zapravo uklanjanje znakova intepunkcije. Znakovi interpunkcije nisu od posebnog značaja za razumevanje polariteta samog teksta i preporučljivo je ukloniti ih. Postupak lematizacije i stemovanja može biti pogrešan ukoliko se ne ukloni šum oko reči. Normalizacija teksta podrazumeva prevođenje datih reči u nestandardnom obliku u kanonski oblik. Ovo je značajno ukoliko je tekst preuzet iz nekih neformalnih izvora kao što su blogovi, forumi, društvene mreže, itd. jer tamo ima puno žargonskih izraza, skraćenica i slično. Preporučljivo je odraditi barem uklanjanje šuma i prevođenje velikih slova u mala. U ovom radu korišćeno je prvih pet metoda. Metode za pretprocesiranje teksta detaljno su objašnjene u [16].

[1]Pojam retkih matrica se odnosi na matrice čije su vrednosti uglavnom nula. Suprotno od retkih matrica su guste matrice (eng. dense matrix). Retke matrice su česta pojava u mašinskom učenju. U narednom poglavlju biće reči o tome kako se dobijaju retke matrice

2.2 Kreiranje vektorskog modela

Kada govorimo o oblasti obrade teksta primenom metoda mašinskog učenja, ulazni korpus bi trebalo preslikati u numeričke vrednosti. Nakon filtriranja dokumenata, potrebno je dokumenta uniformno predstaviti. Predstavljanje dokumenta može biti urađeno na mnogo nacina. Neki od načina su:

- vreća reči,
- vektor unigrama,
- vektor bigrama,
- vektor vrsta reči,
- vektor pozicija reči u tekstu.

Svi ovi modeli obično daju retke matrice. U slučaju vreće reči, model nam daje retku matricu brojeva koja predstavlja vektor pojavljivanja reči u tekstu. U slučaju vektora pozicija dobijamo matricu koja predstavlja vektor reči sa pridruženom oznakom da li se reč nalazi na početku, sredini ili na kraju dokumenta. Detaljnije o vektorskim reprezentacijama modela bavićemo se u narednom poglavlju. Ovakvim vidom predstavljanja dobijamo neke karakteristike, odnosno svojstva teksta (eng. features). Svaka reč ili sintagma[2] će, u zavisnosti od izabranog modela, imati svoju reprezentaciju. U ovoj fazi se kreira i rečnik svih reči ili sintagmi u svim dokumentima, da bi svaka reč ili sintagma imala i svoj jedinstveni identifikator. Ukoliko je izabrani model previše skûp, može se redukovati npr. odabirom najčešćih reči u slučaju vreće reči ili najčešćih n-grama u slučaju n-gramskog modela.

Ovim postupkom je samo odrađena transformacija reči u numeričke vrednosti. Da bi ceo postupak bio pravilno odrađen, najpre ulazni (ne)filtrirani korpus treba podeliti na dva skupa - skup koji će služiti za treniranje (**trening skup**), odnosno za pravljenje modela, tj. klasifikatora i skup koji će služiti za evaluaciju klasifikacionog modela (**testni skup**). Ovde treba voditi računa da se podela na dva skupa uradi nepristrasno, odnosno da skupovi imaju sličnu raspodelu. Ovaj proces se naziva **stratifikacija**. Za nas je značajna **stratifikacija po ciljnoj promenljivoj** koja podrazumeva najpre sortiranje dokumenata po polaritetu (pozitivna, negativna i netrualna dokumenta), a zatim odabir skupova sa definisanim korakom.

[2]Pod sintagmom se misli na n-gram reči, koji će detaljnije biti objašnjen u narednom poglavlju.

Pravljenje vektorskog modela nad trening skupom podrazumeva preslikavanje reči ili sintagmi koje se javljaju u tom skupu u numeričke vrednosti. Nakon pravljenja vektorskog modela nad trening skupom, potrebno je napraviti i vektorski model nad testnim skupom, da bi kasnije mogli izvršiti evaluaciju istreniranog modela. U tom procesu će u obzir doći samo reči koje su se javile i u trening skupu, a reči koje se javljaju samo u testnom skupu se odbacuju. Razlog za to je što model nad trening skupom ne poznaje podatke koje nije imao u trening skupu i novi podaci nemaju svoje predstavnike u postojećem rečniku.

Poželjno je uraditi i **skaliranje** svojstava (eng. feature scaling) da bi opseg numeričkih vrednosti bio jednak za sva svojstva, jer takvi ulazi bolje odgovaraju algoritmima mašinskog učenja. Korišćeno je skaliranje vrednosti na opseg [0, 1]. To praktično znači da je svaka vrednost $x_i \in X$, gde je X trening ili testni skup, transformisana u

$$\frac{x_i - min(X)}{max(X) - min(X)}$$

2.3 Treniranje modela

Pravljenje klasifikacionog modela podrazumeva treniranje modela nad odabranim skupom podataka za treniranje, odnosno snabdevanje algoritma velikom količinom informacija. Vreme treniranja može biti ponekada i izuzetno dugo ako se radi nad velikim skupovima podataka. Vrsta klasifikatora zavisi od izabrane metode za klasifikaciju. U ovom radu su korišćene sledeće metode: metoda potpornih vektora (eng. support vector machine), naivni Bajesov klasifikator (eng. naive Bayes), kombinacija metode potpornih vektora i naivnog Bajesovog klasifikatora, kao i potpuno povezane neuronske mreže (eng. fully connected network). Detaljnije o ovim metodama biće u poglavlju 4.

2.4 Evaluacija modela

Poslednji korak je predikcija i evaluacija istreniranog modela. Skup koji nismo iskoristili služi za proveru tačnosti klasifikacije nad napravljenim modelom. Nad skupom za testiranje izvršićemo predviđanje pripadnosti klase na osnovu istreniranog modela i uporediti sa njegovom stvarnom klasom, što će nam dati ocenu o tome koliko je model zapravo dobro istreniran. Postoje razne mere za evaluaciju klasifikacionog modela. Neke od njih su:

- tačnost klasifikacije,
- preciznost klasifikacije,
- odziv klasifikacije,
- f-mera.

Pretpostavimo da govorimo o binarnoj klasifikaciji dokumenata (klasifikacija na pozitivna i negativna dokumenta). Uvedimo pojmove: stvarno pozitivni primeri (**TP**), stvarno negativni primeri (**TF**), lažno pozitivni (**FP**) i lažno negativni primeri (**FN**). TP predstavlja broj instanci za koje je klasifikator predvideo da pripadaju pozitivnoj klasi, a koje zaista pripadaju toj klasi. TF predstavlja broj instanci za koje je klasifikator predvideo da pripadaju negativnoj klasi, a koje zaista pripadaju toj klasi. FP predstavlja broj instanci za koje je klasifikator predvideo da pripadaju pozitivnoj klasi, a koje zapravo pripadaju negativnoj klasi, a FN predstavlja broj instanci za koje je klasifikator predvideo da pripadaju negativnoj klasi, a koje zapravo pripadaju pozitivnoj klasi. Ovi podaci definišu **matricu konfuzije** (eng. confusion matrix) koja se može prikazati Slikom 2.2. Matrica konfuzije može biti proširena na višeklasnu klasifikaciju. U tom slučaju, element matrice C_{ij} predstavlja broj elemenata koji pripadaju klasi i, a klasifikovani su u klasu j. Jasno je da je model bolji što je više vandijagonalnih vrednosti matrice baš nula.

		stvarna klasa	
		da	ne
predložena klasa	da	stvarno pozitivni	lažno pozitivni
	ne	lažno negativni	stvarno negativni

Slika 2.2: Matrica konfuzije[3]

Sada možemo objasniti vrste ocena modela. **Tačnost klasifikacije** (eng. accuracy) predstavlja udeo tačno klasifikovanih instanci u odnosu na sve klasifikovane

[3]Slika je preuzeta iz [1].

instance i izražava se:

$$acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Preciznost klasifikacije (eng. precision) predstavlja udeo pozitivnih instanci od svih instanci za koje je predviđeno da budu pozitivne, tj.

$$P = \frac{TP}{TP + FP}$$

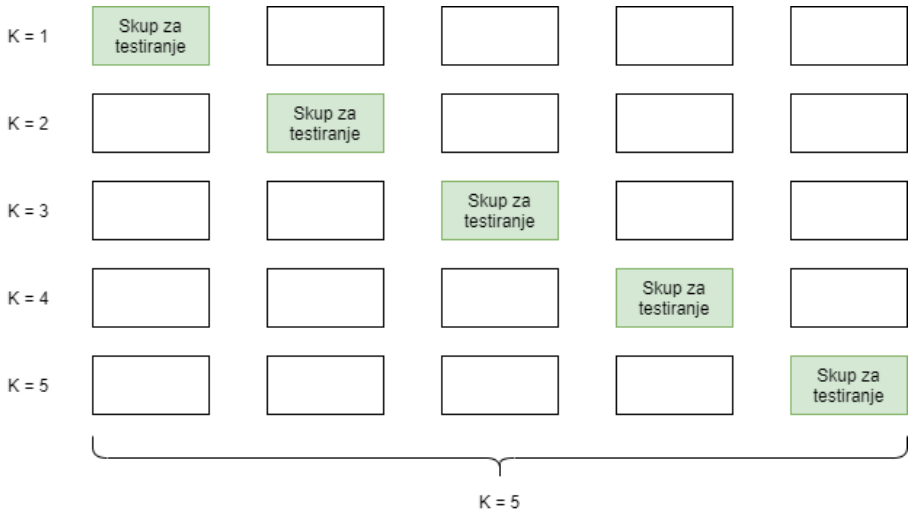
Odziv (eng. recall) je udeo pozitivnih instanci od svih pozitivnih instanci:

$$R = \frac{TP}{TP + FN}$$

F-mera je kombinacija preciznosti i odziva i izražava se kao:

$$F = \frac{2 * P * R}{P + R}$$

Da bi dobili nepristrasnu ocenu modela, može se uraditi **unakrsna validacija** (eng. cross-validation), koja podrazumeva višestruko treniranje i evaluaciju nad trening skupom sa naizmeničnom podelom skupa podataka kako bi dobili što merodavniju ocenu treniranog modela. Za ovaj zadatak korišćena je K-slojna unakrsna validacija (eng. K-fold cross-validation). Ova metoda podrazumeva da se skup za treniranje podeli na K skupova gde će se iterativno trenirati na K-1 skupu, a testirati na preostalom skupu, alternirajući stalno skup za testiranje. K-slojna unakrsna validacija ($K = 5$) je ilustrovana Slikom 2.3.

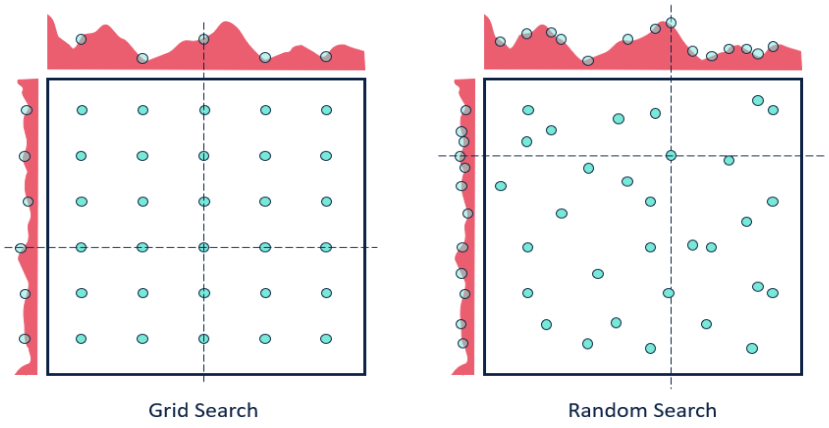


Slika 2.3: K-slojna unakrsna validacija

2.5 Poboljšanje modela

Prilikom treniranja modela može doći do pojave koja se zove preprilagođavanje modela (eng. overfitting), što znači da model previše dobro odgovara treniranim podacima i ne predviđa dobro na nepoznatim podacima. Sa druge strane, može doći i do suprotne pojave, odnosno do nedovoljnog prilagođavanja modela (engl. underfitting), što verovatno znači da je model previše jednostavan i ne daje baš dobru reprezentaciju podataka. Obe ove pojave su posledica bogatstva ili siromaštva modela nad trening skupom.

Da bi model sa izdvojenim karakteristikama što bolje odgovarao izabranom algoritmu za klasifikaciju, poželjno je izvršiti tzv. **mrežnu pretragu** (eng. grid search) za podešavanje metaparametara klasifikatora. Metaparametri su parametri specifični za klasifikator i oni se ne mogu direktno „naučiti" iz podataka, već predstavljaju spoljnu komponentu. Najpre je potrebno napraviti konačnu listu kandidatnih vrednosti za metaparametre za koje želimo da nađemo povoljne vrednosti. Mrežna pretraga radi tako što trenira model i vrši evaluaciju modela nad svim kombinacijama izabranih parametara koristeći K-slojnu unakrsnu validaciju unutar. Izlaz iz mrežne pretrage je klasifikator sa kombinacijom vrednosti metaparametara koja će dati najbolju ocenu. Mrežna pretraga je skûp proces, pa se kao alternativa može primeniti **slučajna mrežna pretraga** (eng. randomized grid search). Razlika ova dva pristupa ilustrovana je Slikom 2.4. U ovom radu korišćen je drugi pristup.



Slika 2.4: Mrežna pretraga i slučajna mrežna pretraga[4]

[4]Slika je preuzeta iz [18].

Glava 3

Vektorske reprezentacije teksta

U ovoj glavi pozabavićemo se detaljnim objašnjenjem nekoliko vrsta vektorskih modela teksta. Tekst nije pogodan za direktnu obradu i algoritmi mašinskog učenja zahtevaju vektore fiksirane dužine na ulazu. Zato je važno odraditi korak enkodiranja teksta u numeričke vrednosti. Ovaj korak je poznat kao **ugrađivanje reči** (eng. word embedding) i često se koristi u obradi prirodnih jezika. Nakon preslikavanja teksta u numeričke vrednosti, svaki dokument biće predstavljen kao retka matrica, odnosno matrica sa velikim brojem nula-vrednosti. Posledica je potreba za velikim računarskim resursima - memorijom i vremenom. Ovo se može poboljšati uvođenjem nekih ograničenja, kao npr. uvođenje maksimalnog broja atributa teksta ili odbacivanje reči koje se ne javljaju više od n puta u tekstu. Uvođenje pretprocesiranja teksta dodatno pomaže uspešnoj obradi i analizi teksta.

Postoji više metoda za preslikavanja teksta u brojeve. Najčešća metoda za predstavljanje teksta je binarna reprezentacija - pridruživanje jedinice ako se reč pojavljuje u tekstu ili nule ako se reč ne pojavljuje u tekstu. Modeli koji su korišćeni u radu, a oslanjaju se na pomenutu metodu su:

- vreća reči,
- vektor unigrama,
- vektor bigrama,
- vektor unigrama i bigrama,
- vektor vrsta reči,
- vektor pozicija reči u tekstu.

Pored osnovne metode za predstavljanje teksta, možemo govoriti i o metodi koja se odnosi na učestalost reči u tekstu, pa su u obzir uzeta još dva modela:

- TF model i
- TF-IDF model.

Još jedan model koji se koristi u radu je Word2Vec model. Ovaj model se oslanja na znanja iz neurosnkih mreža, o čemu će biti reči kasnije.

Svi ovi modeli ne zahtevaju nikakvo lingvističko predznanje. Za sve modele najpre se generiše rečnik i on se generiše samo nad trening skupom. Rečnik podrazumeva skup jedinstvenih reči iz korpusa i eventualno sortiranje tih reči (npr. alfabetsko sortiranje). Logika generisanja rečnika zavisi od izbora vektorskog modela. Pravljenjem rečnika postiže se uniformnost podataka u korpusu. Svaka instanca (reč, n-gram, par) dobija svoj jedinstveni identifikator - poziciju u rečniku. Pored toga, moći ćemo svaki dokument iz ulaznog korpusa da predstavimo kao niz veličine rečnika, a korpus kao dva niza nizova čiji će broj elemenata biti jednak broju dokumenata u trening, odnosno test skupu. Vektorski model koji će se kreirati od testnog skupa (da bi mogli odraditi i evaluaciju klasifikacionog modela) će takođe biti napravljen nad rečnikom skupa za treniranje. Dakle sva dokumenta iz testnog skupa će biti nizovi veličine rečnika izgenerisanog nad trening skupom. Reči koje su se javile u testnom skupu, ali ne i u trening skupu, se odbacuju pri kreiranju modela za testiranje.

3.1 Vreća reči

Kada govorimo o predstavljanju teksta pomoću brojeva, prvi model koji treba objasniti je upravo **vreća reči** (eng. bag of words). Zove se vreća reči zato što se gubi svaka informacija o poziciji reči u tekstu. Iako se u ovom modelu redosled reči, kontekst i semantika zanemaruju, ovakva reprezentacija često daje dobre rezultate. Svaka reč se broji kao jedan atribut teksta. Pokazaćemo na primeru kako se kreira ovaj model. Neka su data sledeća tri dokumenta:

(1) Džon voli da gleda filmove. Meri takođe voli da gleda filmove.
(2) Meri voli da gleda fudbalske utakmice.
(3) Majk ne voli da gleda filmove, ali voli dokumentarne serije.

Kako je fokus na polaritetu dokumenta, a ne rečenice ili sintagme, znakove interpunkcije zanemarujemo. Takođe, sva slova biće prevedena u mala slova. Najpre bi

trebalo izdvojiti sve reči u tekstu. Pored ovoga, vreća reči radi nad „pročišćenim" tekstom, a to podrazumeva uklanjanje stop-reči i lematizaciju. Ulazni tekst sada izgleda ovako:

(1) „džon", „voli", „gleda", „film", „meri", „takođe", „voli", „gleda", „film"
(2) „meri", „voli", „gleda", „fudbalski", „utakmica"
(3) „majk", „ne", „voli", „gleda", „film", „voli", „dokumentaran", „serija"

Potom bi trebalo napraviti rečnik svih reči koje se javljaju u trening skupu. Pretpostavićemo da su prva dva dokumenta iz trening skupa, a poslednji dokument je testni skup. Sortirani rečnik bi izgledao ovako:

Rečnik: [„džon", „film", „fudbalski", „gleda", „meri", „takođe", „utakmica", „voli"]

Svaka reč iz dokumenta će biti preslikana na rečnik, pa dokumenta imaju sledeću reprezentaciju:

(1) BOW 1: [1, 2, 0, 2, 1, 1, 0, 2]
(2) BOW 2: [0, 0, 1, 1, 1, 0, 1, 1]
(3) BOW 3: [0, 1, 0, 1, 0, 0, 0, 2]

Trening skup biće predstavljen kao niz od n elemenata, gde je n broj dokumenata u trening skupu (u ovom primeru $n = 2$), a svaki element će biti niz veličine rečnika (u ovom primeru veličina rečnika je $len(BOWN) = 8$) koji je konstruisan preslikavanjem pojedinačnog dokumenta na izgenerisani rečnik. Testni skup se na sličan način kao i trening skup preslikava u niz od m elemenata, gde je m broj dokumenata u testnom skupu (u ovom primeru $m = 1$). U praksi je veličina rečnika mnogo veća (par hiljada ili miliona).

Vreća reči se koristi u računarskom vidu, obradi prirodnih jezika, Bajesovom filteru za detekciju spam-a, itd. Ova vrsta reprezentacije teksta se može primeniti i u obradi slika i taj model se naziva vreća atributa (eng. bag of features).

3.2 N-gramski modeli

Definicija 1 *Za datu sekvencu tokena $S = (s_1, s_2, ..., s_{N+(n-1)})$, gde su N i n pozitivne celobrojne vrednosti, n-gram sekvence S je podsekvenca uzastopnih tokena dužine n. i-ti n-gram sekvence S je sekvenca $(s_i, s_{i+1}, ..., s_{i+n-1})$[2].*

N-gramski modeli se mogu konstruisati nad rečima, karakterima ili bajtovima. U ovom radu n-grami su konstruisani nad rečima. Najčešće se koriste unigrami ($n = 1$), bigrami ($n = 2$) i trigrami ($n = 3$), ali i njihove kombinacije kao npr. unigrami sa bigramima. Velika razlika n-gramskog modela u odnosu na vreću reči je što ovaj model donekle čuva poredak reči. Takođe, n-gramski modeli rade nad sirovim podacima, dakle nad nefiltriranim korpusom, dok vreća reči radi nad filtriranim korpusom. Jedina tehnika pretprocesiranja koja se ovde koristi je uklanjanje znakova interpunkcije. Uklanjanje stop-reči, kao i lematizacija ili stemovanje se izuzimaju. Zato je često preporučljivo ograničiti veličinu n-gramskog rečnika npr. izdvajanjem najboljih (najčešćih) n-grama u korpusu.

N-gramski modeli su probabilistički modeli koji se zasnivaju na predikciji verovatnoće da se neka reč w pojavi posle neke sekvence reči s, tj. mogu se izraziti kao $P(w|s)$. Da bi se ova verovatnoća izračunala potrebno je u celom korpusu izbrojati koliko se puta naišlo na sekvencu s kao i koliko puta se reč w našla iza date sekvence s. Ovakav pristup se zasniva na Markovljevom modelu[1].

Unigrami su najsličniji modelu vreće reči i grade se na isti način kao i vreća reči. Razlika u odnosu na vreću reči će biti u broju izdvojenih reči, odnosno atributa, zbog nedostatka pretprocesiranja teksta. Unigrami se ne oslanjaju na sekvence koje su se ranije desile.

Bigrami su modeli kod kojih je sekvenca s (u odnosu na koju se traži reč w) samo jedna reč. Ovaj model obuhvata i negaciju, pa se očekuju i bolji rezultati nego kod unigram modela. Reči „volim" i „ne volim" imaju suprotni sentiment i to bigram modeli uzimaju u obzir. Prikazaćemo na primeru kako se gradi ovaj model. Neka imamo opet dokumenata:

(1) Džon voli da gleda filmove. Meri takođe voli da gleda filmove.
(2) Meri voli da gleda fudbalske utakmice.
(3) Majk ne voli da gleda filmove, ali voli dokumentarne serije.

Rečnik sada treba da se generiše nad svim bigramima koji se nalaze u trening skupu, a ne nad svim rečima. Kao i malopre, trening skup će biti prva dva dokumenta.

Rečnik: [(„da", „gleda"), („džon", „voli"), („filmove", „meri"), („fudbalske", „utakmice"), („gleda", „filmove"), („gleda", „fudbalske"), („meri", „takođe"), („meri", „voli"), („takođe", „voli"), („voli", „da")]

[1]Markovljev model je stohastički model koji se odnosi na računanje verovatnoće mogućih događaja uzimajući u obzir trenutni događaj.

Svaki bigram iz dokumenta će biti preslikan na rečnik, pa dokumenta imaju sledeću reprezentaciju:

(1) BIGRAM 1: [2, 1, 1, 0, 2, 0, 1, 0, 1, 2]
(2) BIGRAM 2: [1, 0, 0, 1, 0, 1, 0, 1, 0, 1]
(3) BIGRAM 3: [1, 0, 0, 0, 1, 0, 0, 0, 0, 1]

Na sličan način se konstruišu i kombinacije n-grama, npr. **unigrami sa bigramima**. Rečnik u tom slučaju obuhvata uniju unigrama i bigrama iz trening skupa, a preslikavanje teksta na rečnik se radi kao i do sada.

Kako n-gramski modeli predviđaju pojavljivanje reči nakon sekvence dužine $n-1$, n-gramski modeli se koriste za automatsko završavanje rečenica. Takođe, mogu se koristiti i za automatsku proveru pravopisa i gramatike, kao i za identifikaciju jezika.

3.3 Vektor vrsta reči

Vektor vrsta reči (eng. part of speech) pokušava da pronađe relaciju između reči u tekstu. Reči će se na osnovu definicije i konteksta u tekstu preslikavati u parove (reč, vrsta reči). Prednost u odnosu na vreću reči je što će uglavnom uspešno razdvojiti reči koje se isto pišu, ali koje zavise od konteksta u kom su napisane. Na primer, u rečenici „Ovaj film je **mnogo** dobar" reč **mnogo** će se detektovati kao pridev, a u rečenici „Ovaj film **mnogo** volim da gledam" kao predlog.

Iako je preslikavanje reči u vrste reči za čoveka lako, to nije baš lako preneti na računare, jer u obzir treba uzeti kontekst i smisao reči. Postoje dve glavne tehnike tagovanja (preslikavanja) reči:

- tagovanje zasnovano na pravilima i
- stohastičko tagovanje.

Tagovanje zasnovano na pravilima koristi dostupne informacije iz konteksta da bi se odredila vrsta reči. Višeznačnost se prevazilazi tehnikama kao što je lingvistička analiza reči, posmatranje reči koja prethodi i reči koja sledi, kao i uvođenje skupa pravila koje treba poštovati. Na primer, jedno pravilo može biti: predlozi se uvek nalaze ispred imenica ili imeničkih zamenica, dok prilozi mogu biti samostalni. Pravila zavise od jezika na kom su tekstovi napisani. Uvođenjem ručnih pravila se narušava skalabilnost. Poželjno je pravila graditi direktno nad trening skupom.

Stohastičko tagovanje u praksi daje bolje rezultate. Jedna od heuristika koja se može koristiti je da se odabere najfrekventniji tag od svih tagova za određenu reč iz trening skupa. Međutim, ova tehnika može i ne mora dati dobre rezultate. Drugi način se oslanja na heuristiku kojom su građeni n-grami - računanje verovatnoće da se neka reč w pojavi posle neke sekvence reči s. Ovo znači da će se najreprezentativniji tag naći posmatrajući prethodne $n - 1$ tagove.

Za preslikavanje reči na njihove vrste reči korišćeni su već izgenerisani rečnici sa vrstama reči ili već postojeće biblioteke. Na primeru ćemo pokazati kako se konstruiše vektor vrsta reči, uzimajući u obzir da umemo da odredimo vrstu reči. Neka su date sledeće rečenice:

(1) Džon mnogo voli da gleda akcione filmove. Meri takođe voli da gleda akcione filmove.
(2) Meri jako voli da gleda fudbalske utakmice. Nedavno je pogledala jako dobar film.
(3) Majk ne voli da gleda akcione filmove, ali voli dokumentarne serije.

Poželjno je odraditi korake pretprocesiranja: uklanjanje znakova interpunkcije i stop reči, prevođenje velikih slova u mala, kao i lematizaciju. Pojedinac se sâm može odlučiti za ograničenja nad vektorom vrsta reči. Neke od opcija su odbacivanje rečca i predloga, uzimanje u obzir lica, broja, padeža, roda za prideve i imenice, uzimanje u obzir vrste glagola, itd. Na primeru ćemo samo uraditi prevođenje slova i uklanjanje znakova interpunkcije. Rečnik ćemo izgenerisati tako što ćemo konstruisati sve moguće parove (reč, vrsta reči) nad trening skupom. Neka opet to budu prva dva dokumenta:

Rečnik: [(„akcione”, „pridev”), („da”, „rečca”), („dobar”, „pridev”), („džon”, „imenica”), („film”, „imenica”), („filmove”, „imenica”), („fudbalske”, „pridev”), („gleda”, „glagol”), („jako”, „pridev”), („jako”, „prilog”), („je”, „glagol”), („meri”, „imenica”), („mnogo”, „prilog”), („nedavno”, „prilog”), („pogledala”, „glagol”), („takođe”, „rečca”), („utakmice”, „imenica”), („voli”, „glagol”)]

Svaka reč sa svojom vrstom reči će biti preslikana na rečnik, pa dokumenta imaju sledeću reprezentaciju:

(1) POSTAG 1: [2, 2, 0, 1, 0, 2, 0, 2, 0, 0, 0, 1, 1, 0, 0, 1, 0, 2]
(2) POSTAG 2: [0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1]
(3) POSTAG 3: [1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2]

Vektor vrsta reči se često koristi u obradi prirodnih jezika, npr. za parsiranje, semantičko procesiranje, izvlačenje relevantnih informacija, itd.

3.4 Vektor pozicija reči u tekstu

Još jedan model koji je pokazao dobre rezultate u klasifikaciji dokumenata prema sentimentu je **vektor pozicija reči** u tekstu. Ovaj model vodi računa o tome da li se reč nalazi na početku, sredini ili na kraju dokumenta i često se koristi za sajtove koji imaju mogućnost pretrage.

Kao i do sada, najpre se generiše rečnik. Rečnik u ovom slučaju predstavlja parove reči i njihovih pozicija u tekstu, dakle (reč, pozicija). Pokazaćemo na primeru kako se konstruiše ovaj model. Neka su data dokumenta:

(1) Džon voli da gleda filmove. Meri takođe voli da gleda filmove.
(2) Meri voli da gleda fudbalske utakmice.
(3) Majk ne voli da gleda filmove, ali voli dokumentarne serije.

I ovde je poželjno odraditi korake pretprocesiranja kako bi bolje izvukli ključne reči. Na primeru ćemo uraditi samo konvertovanje slova i uklanjanje znakova interpunkcije. Rečnik nad trening skupom (prva dva dokumenta) izgleda ovako:

Rečnik: [(„da”, „početak”), („da”, „sredina”), („da”, „kraj”), („džon”, „početak”), („filmove”, „sredina”), („filmove”, „kraj”), („fudbalske”, „kraj”), („gleda”, „početak”), („gleda”, „sredina”), („gleda”, „kraj”), („meri”, „početak”), („meri”, „sredina”), („takođe”, „sredina”), („utakmice”, „kraj”), („voli”, „početak”), („voli”, „sredina”)]

Svaka reč sa svojim atributom da li se reč nalazi na početku, sredini ili na kraju teksta će biti preslikana na rečnik, pa dokumenta imaju sledeću reprezentaciju:

(1) POSITION 1: [1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1]
(2) POSITION 2: [0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0]
(3) POSITION 3: [1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0]

Intuicija je da će najbitnije, odnosno ključne stvari koje određuju polaritet dokumenta biti rečene na kraju. Prirodno bi bilo da se reči nakon prevođenja u vektorski model pomnože sa nekim faktorom, gde bi taj faktor bio najmanji za reči na početku teksta i koji bi se povećavao postepeno. U ovom radu množenje faktorom je izostavljeno.

3.5 TF model

TF model (eng. term frequency), za razliku od većine modela, kao metodu za predstavljanje reči ne koristi preslikavanje jedinicom, ako se reč nalazi u tekstu, ili nulom, ako se reč ne nalazi u tekstu. Kao što mu i ime govori, koristi se učestalost reči da bi se oslikala bitnost reči u dokumentu koji se nalazi u korpusu. Što je dokument veći, to je verovatnoća da se neka reč pojavi veća. Kako bi izbegli grešku da određena reč ima veću frekvenciju što je dokument u kom se ona nalazi veći, radi se normalizacija modela. Normalizacija modela zapravo predstavlja deljenje frekvencije reči sa brojem reči u dokumentu. TF model se može kreirati nad vrećom reči, odnosno nad filtriranim korpusom, ali i nad n-gramima. U ovom radu TF model građen je nad vrećom reči.

Kao i obično, generiše se rečnik svih reči u dokumentu. Reč w iz dokumenta d se sada preslikava po sledećoj formuli:

$$tf(w, d) = \frac{broj\ reči\ w\ u\ dokumentu\ d}{broj\ svih\ reči\ u\ dokumentu\ d}$$

Ovaj model daje bolje rezultate ukoliko su stop-reči izuzete iz razmatranja, jer bi inače ove reči imale veliku frekvencijsku vrednost usled činjenice da su to česte reči. Dodatno se preporučuje uklanjanje znakova interpunkcije, kao i odrađivanje koraka stemovanja ili lematizacije.

3.6 TF-IDF model

Da bi uveli TF-IDF model (eng. term frequency - inverse document frequency), potrebno je najpre uvesti neke pojmove. Neka je *df* veličina koja predstavlja broj dokumenata u kojima se određena reč pojavljuje. Pri tome, dovoljno je samo da se reč pominje u dokumentu, ne uzimajući u obzir koliko puta, pa dobijamo sledeću formulu:

$$df(w) = broj\ dokumenata\ u\ kojima\ se\ pojavljuje\ reč\ w$$

Neka je *idf* veličina koja predstavlja broj svih dokumenata u korpusu u odnosu na broj dokumenata u kojima se pojavljuje određena reč. Dobijamo sledeću formulu:

$$idf(w) = \frac{broj\ svih\ dokumenata\ u\ korpusu}{df(w)}$$

Problem kod prethodne formule je to što će za velike korpuse, odnosno za veliki broj dokumenata, vrednost *idf* biti jako velika. Da bi se to izbeglo, koristi se logaritamska

funkcija. Pored toga, ukoliko se pretraga radi za reč koja ne postoji u rečniku, dobićemo deljenje nulom, a to ne smemo da dozvolimo. Iz tog razloga, prethodna formula se malo koriguje:

$$idf(w) = log\left(\frac{broj\ svih\ dokumenata\ u\ korpusu}{df(w)+1}\right)$$

TF-IDF model je zapravo kombinacija prethodno definisanih idf i tf veličina, odnosno:

$$tf{-}idf(w,d) = tf(w,d) * idf(w)$$

I ovaj model se može kreirati nad vrećom reči, odnosno nad filtriranim korpusom, ali i nad n-gramima. U ovom radu TF-IDF model građen je nad vrećom reči. Modeli koji koriste frekvenciju kao metodu predstavljanja teskta su nastali iz potrebe pretrage dokumenata, pronalaska informacija u dokumentima, ali i ekstrakcije ključnih reči iz dokumenata.

3.7 Word2Vec model

Najkompleksniji obrađeni model je upravo **Word2Vec model**, koji koristi znanja iz neuronskih mreža. Detaljno o neuronskim mrežama biće u narednom poglavlju. Word2Vec model koristi dvoslojnu neuronsku mrežu za vektorizaciju teksta. Kao što mu i ime kaže, svaka reč će biti zamenjena nekim vektorom. Intuicija za kreiranje ovog modela je da će slične reči, odnosno reči koje imaju sličan smisao (npr. „serija" i „film"), biti vektorizovane sličnim vektorima. Postoje dva pristupa koji se koriste za generisanje ovog modela:

- pristup neprekidne vreće reči (eng. continuous bag of words) i
- skip-gram pristup.

Prvi pristup podrazumeva korišćenje konteksta da bi se našla ciljna reč, a drugi pristup podrazumeva korišćenje ciljne reči za predikciju konteksta. Skip-gram pristup u praksi daje bolje rezultate za veće korpuse, pa će akcenat biti na njemu. Kreiranje Word2Vec modela se radi tako što najpre pomeramo „prozor reči" kroz dokumenta. Hajde da vidimo to na primeru. Neka je dat dokument:

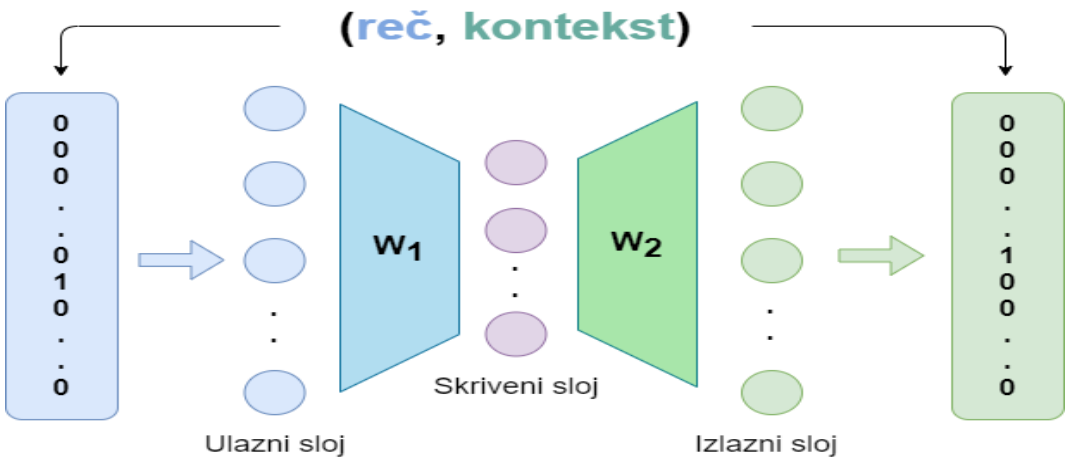
Džon voli da gleda filmove. Meri takođe voli da gleda filmove.

Pretpostavimo da je prozor veličine dva. To znači da ćemo posmatrati reči koje su od ciljne reči udaljene najviše za dva mesta. Parovi koje na taj način dobijamo su parovi oblika (ciljna reč, kontekstna reč), gde je ciljna reč svaka reč iz trening skupa, a kontekstna reč je zapravo reč obuhvaćena prozorom ciljne reči. Reči koje se nalaze na početku ili kraju dokumenta imaće manji broj parova sa odgovarajućom kontekstnom rečju. Parovi su:

Parovi: [(„džon", „voli"), („džon", „da"), („voli", „džon"), („voli", „da"), („voli", „gleda"), („da", „džon"), („da", „voli"), („da", „gleda"), („da", „filmove"), („gleda", „voli"), („gleda", „da"), („gleda", „filmove"), („gleda", „meri"), („filmove", „da"), („filmove", „gleda"), („filmove", „meri"), („filmove", „takođe"), („meri", „gleda"), („meri", „filmove"), („meri", „takođe"), („meri", „voli"), („takođe", „filmove"), („takođe", „meri"), („takođe", „voli"), („takođe", „da"), („voli", „meri"), („voli", „takođe"), („da", „takođe"), („da", „voli"), („da", „gleda"), („da", „filmove"), („gleda", „voli"), („gleda", „da"), („gleda", „filmove"), („filmove", „da"), („filmove", „gleda")]

Rečnik za ovaj model predstavlja sve reči u trening skupu. Za predikciju kontekstne reči koristi se neuronska mreža. Svaki prethodno generisani par će sada biti propušten kroz neuronsku mrežu. Ulaz u skip-gram arhitekturu će biti vektor veličine rečnika, sa jedinicom na mestu koje odgovara ciljnoj reči u rečniku i nulama na ostalim pozicijama. Izlaz će biti vektor iste veličine sa jedinicom na mestu u rečniku koje odgovara kontekstnoj reči i nulama na ostalim pozicijama. Ovaj model između ulaznog i izlaznog sloja ima jedan skriveni sloj koji je obično veličine nekoliko stotina. Treniranje Word2Vec modela prikazano je na Slici 3.1.

Cilj ovog treniranja je da model nauči da kada dobije neku reč da odredi i njen kontekst. Time će model naučiti koje su reči semantički slične. Ovo se dešava iz razloga što se češće zajedno javljaju slične reči, nego one koji nisu slične. Težine grana između slojeva često nemaju smisla za ljudski um, ali šta god model nauči pri treniranju zapisuje se u težine. Kako ulazni vektor ima sve nule i jednu jedinicu, to će nakon množenja sa matricom težina W_1 skriveni vektor sadržati samo težine koje odgovaraju poziciji jedinice u ulaznom vektoru. Svi parovi iz trening skupa biće propušteni kroz neuronsku mrežu i trenirani. Kada propustimo neku reč kroz istreniranu mrežu, dobićemo vektor koji predstavlja reč sa najvećom verovatnoćom da se nalazi u blizini date reči. Dakle, svaka reč će imati svoj ugrađeni vektor skrivenog sloja. Da bi model prilagodili algoritmima mašinskog učenja, svaki dokument će biti predstavljen kao vektor veličine ugrađenog vektora koji će predstavljati uprosečenu



Slika 3.1: Skip-gram pristup

vrednost ugrađenih vektora reči koje su se našle u tom dokumentu.

Prednost ovog modela je pre svega činjenica da se čuva kontekst reči. Pored toga, velika prednost je mala dimenzija vektorskog prostora koji čuva informacije o kontekstu. Teorijski deo objašnjenja Word2Vec modela preuzet je iz [15].

Glava 4

Metode mašinskog učenja

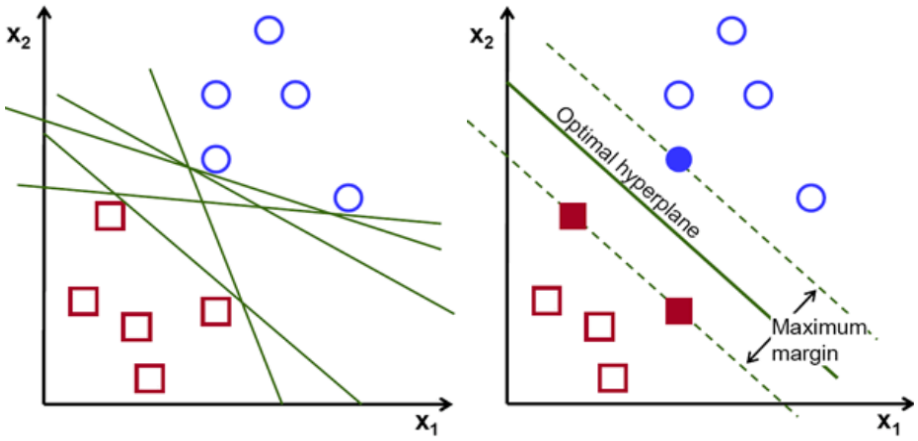
Mašinsko učenje je grana veštačke inteligencije koja se zasniva na ideji da sistem sâm može naučiti obrazac ponašanja iz ulaznih podataka i tako se usavršiti da sâm i odlučuje. U praksi je mašinsko učenje jako primenljivo - za klasifikaciju, regresiju[1], prepoznavanje objekata, obradu slika, autonomnu vožnju, itd. Kada su vektorski modeli generisani, potrebno je takve modele provući kroz algoritme mašinskog učenja da bi ispunili cilj ovog rada, a to je klasifikacija dokumenata. Mašinsko učenje podrazumeva postojanje ulaznih parametara, koji su zapravo **atributi** koji opisuju ono što želimo da obradimo, kao i postojanje izlaznih parametara, tj. **ciljnih promenljivih**. Cilj metoda mašinskog učenja je pronaći vezu između ulaznih i izlaznih parametara, odnosno definisati **model**. Model treba težiti tome da veza između ulaznih i izlaznih parametara bude što opštija i da ima osobinu da sa što manjom greškom predviđa ciljnu promenljivu na osnovu atributa. Možemo definisati funkciju $f(x)$ koja vrednostima atributa x pridružuje vrednosti ciljne promenljive y. Motivacija je težiti osobini $f(x) \approx y$, odnosno da stvarne vrednosti ciljne promenljive budu što bliže aproksimiranim vrednostima. U ovoj glavi pozabavićemo se detaljnim objašnjenjem algoritama mašinskog učenja korišćenih u radu, a to su:

- metod potpornih vektora,
- naivni Bajesov klasifikator,
- naivni Bajesov klasifikator sa metodom potpornih vektora,
- poptuno povezane neuronske mreže.

[1]Regresija je problem predviđanja neprekidne ciljne promenljive. Na primer, može se uočiti veza količine padavina i stepena erozije, širine reke i maksimalnog godišnjeg proticaja, itd.

4.1 Metod potpornih vektora

Metod potpornih vektora (eng. support vector machine, SVM) jedan je od najčešćih algoritama za klasifikaciju. SVM klasifikator razdvaja klase koristeći hiperravni kao granice odlučivanja. Prirodno je da se desi greška pri odlučivanju, a verovatnoća da se ona desi je veća ukoliko je objekat bliži hiperravni. Pojam margina se odnosi na distancu objekata od granice odlučivanja o pripadnosti klasi, pa što je margina veća, to je i mogućnost pojave pomenute greške manja. Zato su od velike koristi modeli koji se zasnivaju na **velikim marginama** (eng. large margin). Cilj SVM algoritma je ne samo pronaći hiperravan koja će razdvajati ulazne vektore u različite klase (leva strana Slike 4.1), već pronaći takvu hiperravan koja će te vektore razdvajati sa maksimalnom marginom (desna strana Slike 4.1). Ako govorimo o dvodimenzionom vektorskom prostoru, onda je hiperravan zapravo linija; ako govorimo o trodimenzionom vektorskom prostoru, onda je hiperravan dvodimenziona ravan, itd.



Slika 4.1: Određivanje hiperravni kod SVM metode[2]

Radi jednostavnosti, za upoznavanje sa SVM metodom, pretpostavićemo da govorimo o binarnoj klasifikaciji. Neka je dato n objekata koje treba klasifikovati $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, gde su $x_i \in R^n$ objekti, a $y_i \in \{-1, 1\}$ indikatori kojoj klasi objekat x_i pripada, za $i = \overline{1, n}$. Cilj je da nađemo maksimalnu marginu hiperravni koja će objekte za koje važi $y_i = 1$ razdvajati od objekata za koje važi

[2]Slika je preuzeta iz [15].

$y_i = -1$. Jednačina hiperravni je:

$$wx + w_0 = 0$$

gde je w vektor normale na hiperravan, a w_0 je slobodni član. Optimalna hiperravan je ona hiperravan koja je maksimalno udaljena od najbližih predstavnika pojedinačne klase. Hiperravni koje su paralelne optimalnoj hiperravni, a dodiruju najbliže predstavnike klase opisuju se sledećim jednačinama:

$$wx + w_0 = c$$

$$wx + w_0 = -c$$

Objekti koji pripadaju ovim dvema hiperravnima čine potporne vektore. Širina margine se dobija iz jednačine za rastojanje tačke od ravni i iznosi $\frac{2}{\|w\|}$. Problem koji sada treba rešiti je maksimizacija dužine margine, tj. minimizacija njenog inverza. Dodatno, treba obezbediti da objekti ne upadnu u prostor margine, pa optimizacioni problem izgleda ovako:

$$\min_{w,w_0} \frac{\|w\|_2}{2}$$
$$y_i(wx_i + w_0) \geq 1, \quad za \quad i = \overline{1,n}$$

U praksi je često nemoguće potpuno linearno odvojiti klase, pa se pribegava uvođenju greške. To podrazumeva da se za svaku trening instancu uvodi ξ_i, za $i = \overline{1,n}$, koja podrazumeva udaljenost instance od hiperravni potpornog vektora odgovarajuće klase, ali sa pogrešne strane. Ovaj model se zasniva na **mekim marginama** (eng. soft margin). Optimizacioni problem sada izgleda ovako:

$$\min_{w,w_0} \frac{\|w\|_2}{2} + C\sum_{i=1}^{n} \xi_i$$

$$y_i(wx_i + w_0) \geq 1 - \xi_i, \quad za \quad i = \overline{1,n}$$

$$\xi_i \geq 0 \quad za \quad i = \overline{1,n}$$

Metaparametar C služi za indikaciju bitnosti greške. Ako je $C = 0$ onda greške nisu važne i mogu biti velike; ako je C veliko onda su greške važne a pravac hiperravni i širina pojasa nisu mnogo važni. Teorijski deo objašnjenja SVM metode većinom je preuzet iz [3].

SVM metoda se, pored klasifikacije, može koristiti i za regresiju. Takođe, može se koristiti za detekciju lica na slikama, gde se delovi slike klasifikuju kao „lice" i

„ne-lice". Može se koristiti u bioinformatici za klasifikaciju proteina i kancera, za prepoznavanje rukopisa, kao i u još mnogim problemima.

Jedan od metaparematara SVM metode su kerneli, koji služe za rešavanje problema linearne razdvojivosti objekata. Objasnićemo ukratko princip kernela.

Kerneli

Kerneli su funkcije koje ulazne vektore preslikavaju u višedimenzioni prostor gde se oni mogu linearno razdvojiti ili barem bolje struktuirati. Pri tome, transformacija vektora iz jednog u drugi prostor nije eksplicitna, već se koristi tzv. kernelski trik. To znači da ako se algoritam može iskazati u terminima skalarnog proizvoda dva vektora, onda je transformacija samo zamena tog skalarnog proizvoda skalarnim proizvodom novog prostora, odnosno kernel funkcijom. Računanje skalarnog proizvoda manjedimenzionog prostora svodi se samo na računanje skalarnog proizvoda višedimenzionog prostora, bez dodatnog računa. Kernel funkcija se obično označava kao:

$$K(x, y) = \langle f(x), f(y) \rangle$$

gde su x i y n-dimenzioni vektori, a f funkcija koja slika iz n-dimenzionog prostora u m-dimenzioni prostor, gde je obično $m >> n$. Kernel funkcija treba da ima sledeće osobine:

- neprekidnost,
- simetričnost,
- postojanje pozitivno (semi)definitne Gram matrice[3].

Pozitivno definitan kernel garantuje da optimizacioni problem bude konveksan, a rešenje jednistveno. Međutim, mnoge kernel funkcije koje nisu striktno pozitivno definitne su pokazale dobre rezultate u praksi. Objasnićemo ukratko par najpoznatijih kernela.

Linearni kernel je najjednostavniji i brži je od ostalih. Iskazuje se formulom:

$$K(x, y) = x^T y$$

[3]**Gram matrica** vektora $a_1, a_2, ..., a_n$ je matrica G svih kombinacija skalarnog proizvoda, tj. $G = \langle a_i, a_j \rangle$, za $i, j = \overline{1, n}$.

Polinomijalni kernel se često koristi u NLP-u. Za polinome stepena d (d je obično 2) polinomijalni kernel je oblika:

$$K(x, y) = (x^T y + c)^d, \quad c = const$$

Problem kod polinomijalnog kernela je numerička nestabilnost. U slučaju da je $x^T y + c < 1$, onda $K(x, y)$ teži nuli za rastuće d, dok za $x^T y + c > 1$, $K(x, y)$ teži beskonačnosti.

Gausov kernel se, pored linernog kernela, takođe često koristi u praksi. Ovaj kernel je oblika:

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{\gamma}\right)$$

gde je γ parametar koji označava širinu Gausovog zvona. Teorijski, ukoliko je γ dovoljno malo, svaki objekat bi mogao biti potporni vektor koji bi se množenjem Gausovog zvona preslikao u tačnu klasifikaciju. Posebna vrsta Gausovog kernela je **RBF** kernel (eng. Radial basis function) koji se dobija zamenom $\gamma = 2\sigma^2$.

4.2 Naivni Bajesov klasifikator

Naivni Bajesov klasifikator je probabilistički algoritam koji se zasniva na **Bajesovoj teoremi**:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

gde formulacija $P(A|B)$ označava verovatnoću da se desi događaj A pod uslovom da se desio događaj B. Ovaj algoritam pretpostavlja da je vrednost nekog atributa u klasi potpuno nezavisna od vrednosti ostalih atributa u toj klasi, pa zato u nazivu i ima reč „naivni". Iako je princip koji se ovde koristi jednostavan, naivni Bajesov klasifikator daje dobre rezultate za velike ulazne korpuse, a pored toga se odlikuje izuzetnom vremenskom efikasnošću. Zbog nerealne pretpostavke da su atributi nezavisni jedni od drugih, drugi klasifikatori (kao npr. SVM) se pokazuju boljim.

U slučaju klasifikacije teksta, prethodna teorema se može zapisati kao:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

gde je C klasa (pozitivno/negativno/neutralno), a X vektor vrednosti atributa, odnosno $X = (x_1, x_2, ..., x_n)$, pa se poslednja jednačina prevodi u:

$$P(C|x_1, x_2, ..., x_n) = \frac{P(x_1|C)P(x_2|C)...P(x_n|C)P(C)}{P(x_1)P(x_2)...P(x_n)}$$

Za sve ulaze imenilac se ne menja, pa se poslednja jednačina može aproksimirati na sledeći način:

$$P(C|x_1, x_2, ..., x_n) \approx P(C) \prod_{i=1}^{n} P(x_i|C)$$

Klasa koja će biti rezultat klasifikacije je ona sa najvećom verovatnoćom:

$$C = argmax_C P(C) \prod_{i=1}^{n} P(x_i|C)$$

Postoje dva glavna tipa naivnog Bajesovog klasifikatora: multinomijalni i Gausov klasifikator. **Multinomijalni naivni Bajesov klasifikator**, odnosno **MNB**, se koristi za problem klasifikacije. Atributi uglavnom koriste celobrojnu numeričku reprezentaciju, ali može se koristiti i frekvencija atributa u tekstu. Verovatnoća da vrednost atributa x_i pripada klasi C može se proceniti metodom glačanja na sledeći način:

$$P(x_i|C) = \frac{N_{C_i} + \alpha}{N_C + \alpha n}$$

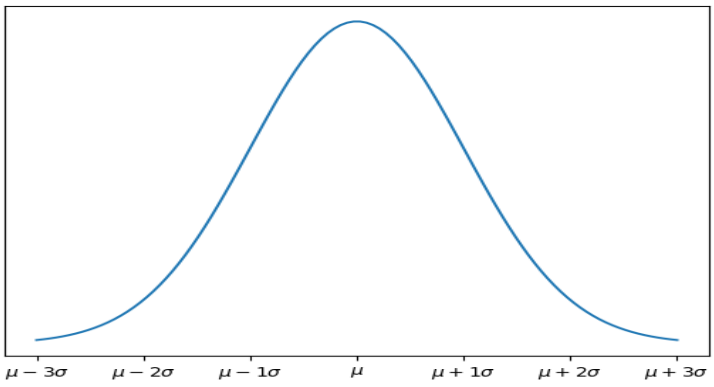
gde je N_{C_i} broj koliko puta se vrednost atributa i pojavila u dokumentu trening skupa koji pripada klasi C, a N_C je ukupan broj atributa klase C. Ukoliko je pojavljivanje neke vrednosti nekog atributa retko, postoji mogućnost da se ta vrednost uopšte i ne nalazi u testnom skupu, pa će verovatnoća te vrednosti biti nula, a samim tim i ceo proizvod sa desne strane prethodne jednačine će biti nula. Upravo se parametar $\alpha \geq 0$ koristi za vrednosti atributa koji se ne pojavljuju u skupu da bi se izbegla verovatnoća koja ima vrednost nula. Za $\alpha = 1$ glačanje je Laplasovo, a za $\alpha < 1$ glačanje je Lidstounovo.

Pretpostavimo da vrednosti atributa imaju normalnu, odnosno Gausovu raspodelu, prikazanu na Slici 4.2. **Gausov naivni klasifikator** se modeluje tako što računa aritmetičku sredinu i standardnu devijaciju vrednosti atributa u okviru svake klase. Verovatnoća da vrednost atributa x_i pripada klasi C može se opisati kao:

$$P(x_i|C) = \frac{1}{2\pi\sigma_C^2} \exp\left(-\frac{(x_i - \mu_C)^2}{2\sigma_C^2}\right)$$

gde je μ_C aritmetička sredina fičera klase C, a σ_C standardna devijacija[4] klase C. Teorijski deo objašnjena Bajesovog klasifikatora preuzet je iz [13].

[4]Standardna devijacija je mera koja govori koliko u proseku elementi skupa odstupaju od aritmetičke sredine skupa i izražava se formulom: $\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2}$.



Slika 4.2: Normalna raspodela

4.3 Naivni Bajesov klasifikator sa metodom potpornih vektora

Naivni Bajesov klasifikator sa metodom potpornih vektora, tj. **NBSVM**, je klasifikator koji kombinuje multinomijalni Bajesov klasifikator i metodu potpornih vektora. MNB služi za računanje težina atributa u vektorskom modelu (najbolje u n-gramskom modelu), a potom se nalazi logaritamski odnos težina za pozitivnu i negativnu klasu, odnosno:

$$p = \alpha + N_{POS_i} \quad i \quad q = \alpha + N_{NEG_i}$$

$$r = log\left(\frac{p/\left\|p\right\|}{q/\left\|q\right\|}\right)$$

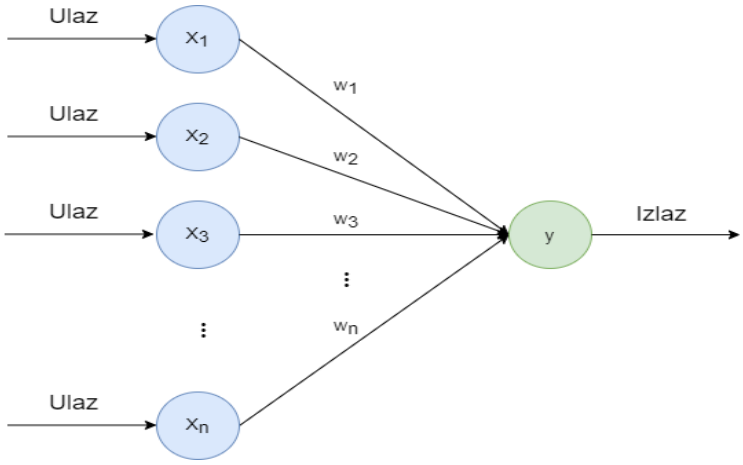
gde je N_{POS_i}/N_{NEG_i} broj koliko puta se atribut i pojavio u dokumentu trening skupa iz pozitivne/negativne klase, a $\alpha \geq 0$ parametar za glačanje. Matrica atributa, tj. vektorska reprezentacija dokumenata se množi dobijenim koeficijentom r, a zatim se model trenira (linearnom) verzijom SVM metode.

4.4 Poptuno povezane neuronske mreže

Najpoznatije metode mašinskog učenja su upravo **neuronske mreže**. Neuronske mreže su nastale kao posledica pokušaja simuliranja biološkog nervnog sistema. Ljudski mozak se sastoji od nervnih ćelija koje obrađuju informacije dobijene od

čula. Nervne ćelije su povezane sa drugim nervnim ćelijama na mestima koje se nazivaju sinapse, pomoću kojih se prenose električni impulsi. Ljudski mozak uči tako što menja jačinu sinaptičke veze između nervnih ćelija zbog ponavljajućeg stimulisanja jednim istim impulsom [1].

Sličan koncept se koristi u mašinskom učenju. Nervne ćelije se simuliraju u računaru. One se međusobno povezuju gradeći mrežu i na taj način podaci mogu da se prenose. Osnovna jedinica veštačke neuronske mreže je **neuron**. On se sastoji od ulaznih i jednog izlaznog čvora. Grane koje povezuju te ulazne čvorove sa izlaznim čvorom imaju težine koje oslikavaju jačinu električnih impulsa u ljudskom mozgu. Kako vizuelno izgleda neuron može se videti na Slici 4.3.



Slika 4.3: Neuron

Cilj neurona je da odredi pripadnost nekog atributa (npr. pripadnost reči, n-grama, para) binarnoj klasi na osnovu ulaznih vrednosti i težina. Pri tome se formira linerna funkcija koja predstavlja sumu proizvoda ulaznih vrednosti i odgovarajućih težina, odnosno,

$$y = \sum_{i=1}^{n} w_i X_i$$

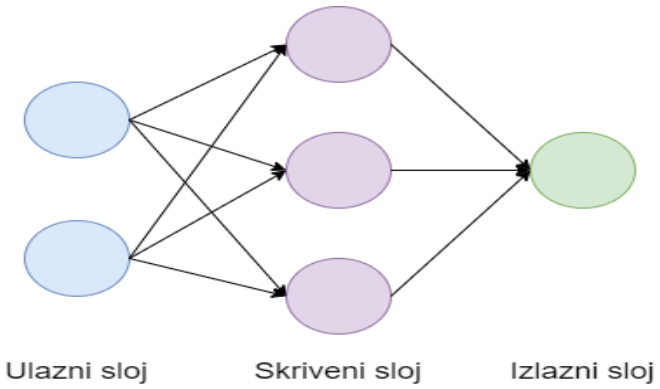
gde je y vrednost izlaznog čvora, X_i ulazni čvor, w_i težina odgovarajuće grane ulaznog čvora, a n broj ulaznih čvorova. Kada se dobije ukupna suma proizvoda čvorova i odgovarajućih grana, onda se primenjuje **aktivaciona funkcija** koja vrednost y slika obično u $\{0, 1\}$ ili $\{-1, 1\}$. Neki primeri aktivacionih funkcija koje se koriste su:

- identitet (bez operacije), tj. $f(x) = x$

- sigmoidna funkcija, tj. $f(x) = \frac{1}{1+e^{-x}}$
- hiperbolički tangens, tj. $f(x) = tanh(x)$
- relu funkcija (eng. rectified linear unit function), tj. $f(x) = max(0, x)$

Težine koje se koriste u neuronu se zapravo iterativno podešavaju sve dok se ne desi da se u nekoj iteraciji nisu promenile. Za inicijalnu vrednost težina obično se uzimaju slučajno odabrane vrednosti.

Za rešavanje kompleksnijih problema, neuroni se udružuju sa drugim neuronima gradeći višeslojne mreže. Svaki neuron jednog sloja šalje svoje izlaze neuronima narednog sloja. Slojevi koje se nalaze između ulaznog i izlaznog sloja nazivaju se skriveni neuronski slojevi. Prikaz višeslojne mreže može se videti na Slici 4.4.



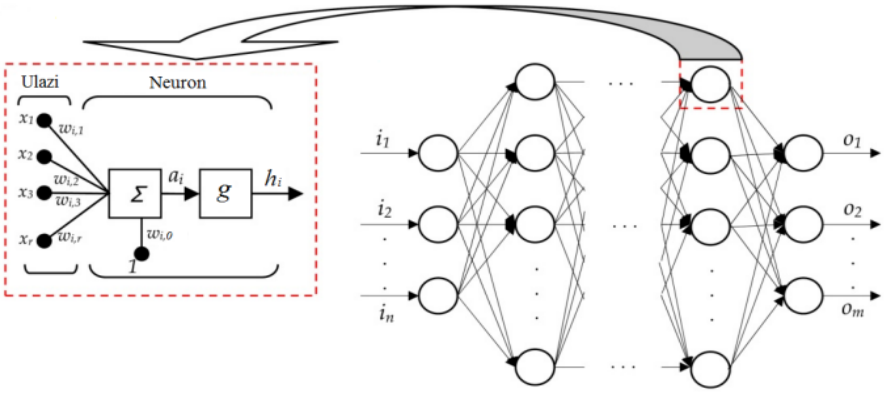
Slika 4.4: Višeslojna neuronska mreža

Neke vrste veštačkih neuronskih mreža (eng. artificial neural network) su:

- potpuno povezane neuronske mreže,
- konvolutivne neuronske mreže,
- rekurentne neuronske mreže.

Pomenuti Word2Vec model pripada potpuno povezanim neuronskim mrežama. Ovde ćemo objasniti samo prvu grupu. **Potpuno povezane mreže** (eng. fully connected network) su neuronske mreže koje se sastoje od najmanje tri sloja (ulaznog, skrivenog i izlaznog), a čvorovi su organizovani tako da se svi čvorovi jednog sloja povezuju sa svim čvorovima narednog sloja, vodeći računa da se ne stvori ciklus. Svaki čvor je linearna kombinacija svojih argumenata, a svi čvorovi, osim izlaznih

čvorova, dodatno imaju nelinearnu aktivacionu funkciju. Prikaz potpuno povezane mreže može se videti na Slici 4.5.



Slika 4.5: Potpuno povezana neuronska mreža

Algoritam koji se zasniva na potpuno povezanim mrežama je **MLP** algoritam (eng. multi layer perceptron). Za dati ulazni vektor x definiše se model:

$$h_0 = x$$

$$h_i = g(W_i h_{i-1} + w_{i0}), \quad i = \overline{1, L}$$

gde je L broj slojeva, W_i matrica čija j-ta vrsta predstavlja vektor vrednosti parametara j-tog neurona u i-tom sloju, w_{i0} je vektor slobodnih članova linearnih kombinacija neurona i-tog sloja. Funkcija g je aktivaciona funkcija.

Za klasifikacioni problem se na izlazni sloj primenjuje funkcija **mekog maksimuma** (eng. softmax) koja je oblika:

$$softmax(x)_i = \frac{exp(x_i)}{\sum_{l=1}^{k} exp(x_l)}$$

gde je x_i predstavlja i-ti element ulaza u softmax funkciju koja odgovara klasi i, a k broj klasa. Na taj način se vrednosti ulaznog vektora transformišu u vrednosti gde će najveća pozitivna vrednost još više odskakati od ostalih. Teorijski deo objašnjenja MLP metode kao i Slika 4.5 preuzeti su iz [3].

Ideja je naučiti mrežu da prepoznaje šablone svake klase, tako da, kada se proslede novi podaci, mreža je u stanju da odredi klasu. Samo „učenje" se odvija u

neuronima menjajući težine nakon procesiranja podataka na osnovu greške koja se dobija upoređivanjem krajnjih dobijenih vrednosti sa očekivanim. Ovo se zove metod **propagacije unazad**. Na osnovu greške koja je izračunata na kraju jedne iteracije, treba se vraćati sloj po sloj unazad računajući prvo parametre izlaznog sloja, pa skrivenog sloja. Propagacija unazad sastoji se od sledećih koraka:

- Inicijalizovati sve težine malim, slučajno odabranim vrednostima.
- Proslediti ulazne podatke mreži i izračunati grešku. Bitno je da je funckija greške diferencijabilna.
- Da bi minimizovali grešku, računaju se gradijenti svih težina. Kako gradijenti predstavljaju pravce najvećeg porasta funkcije, koriste se suprotni pravci gradijenata.
- Prethodna dva koraka se ponavljaju dok god greška ne postane manja od nekog unapred definisanog praga.

Preciznije je reći da MLP trenira koristeći optimizacione algoritme kao što su stohastički gradijentni spust i Adam metoda, a gradijenti se računaju koristeći propagaciju unazad. **Stohastički gradijentni spust** (eng. stochastic gradient descent) se koristi za obučavanje modela sa velikom količinom informacija. Ako se funkcija koja se računa $f(x)$ može predstaviti kao prosek drugih funkcija, tj. ako važi:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

onda je pravilo izračunavanja novog koraka:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Postoje dva načina biranja koraka α_k. Jedan pristup je da je $\alpha_k = \alpha$ za svako i i predstavlja slučaj da su koraci dovoljno veliki da se može dostići rešenje. Drugi pristup podrazumeva ispunjavanje Robin Monroovih uslova:

$$\sum_{k=1}^{\infty} \alpha_k = \infty, \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty$$

i podrazumeva slučaj da su koraci dovoljno mali da niz tačaka α_k kovergira rešenju. **Adam** metoda (eng. adaptive moment estimation) je najčešći algoritam za obučavanje mreža. Koristi ocene prvog i drugog momenta gradijenata date formulama:

$$m_0 = 0, \quad v_0 = 0$$

$$m_{k+1} = \frac{\beta_1 m_k + (1-\beta_1)\nabla f(x_k)}{1-\beta_1^{k+1}}, \quad 0 \leq \beta_1 < 1$$

$$v_{k+1} = \frac{\beta_2 v_k + (1-\beta_2)\nabla f(x_k) \odot \nabla f(x_k)}{1-\beta_2^{k+1}}, \quad 0 \leq \beta_2 < 1$$

Pravilo izračunavanja novog koraka glasi:

$$x_{k+1} = x_k - \alpha_{k+1} \frac{m_{k+1}}{\sqrt{v_{k+1}} + \epsilon}$$

Prednost MLP metode pre svega je efikasnost za nelinearne probleme. Međutim, u zavisnosti od inicijalizacije težina, krajnja tačnost može dosta varirati. Dodatno, vreme izračunavanja može biti jako dugo.

Glava 5

Implementacija

Praktični deo ovog master rada implementiran je u programskom jeziku *Python* i može se naći ovde [11]. Ova implementacija obrađuje 4 različita korpusa sa labeliranim podacima. U pitanju su sledeći korpusi:

- **SerbianPD-3C** - korpus sprskog jezika sa tri klase (pozitivna, negativna i neutralna) koji sadrži po 841 filmsku recenziju za svaku klasu. Korpus se može naći u [8].
- **SerbianPD-2C** - korpus sprskog jezika sa dve klase (pozitivna i negativna) koji sadrži po 841 filmsku recenziju za svaku klasu. Korpus se može naći u [8].
- **CornellPD** - korpus engleskog jezika sa dve klase (pozitivna i negativna) koji sadrži po 1000 filmskih recenzija za svaku klasu. Korpus se može naći u [9].
- **HUMIR** - korpus turskog jezika sa dve klase (pozitivna i negativna) koji sadrži po 26700 filmskih recenzija za svaku klasu, ali je zbog nedovoljno memorijskih resursa uzet slučajno odabran podskup od po 4000 filmskih recenzija za svaku klasu. Korpus se može naći u [10].

Pri pokretanju aplikacije, postoji mogućnost odabira korpusa koji će se obraditi. Svaki korpus se najpre deli na skup za treniranje i skup za testiranje razmeri 80:20, ali postoji mogućnost odabira ove razmere. Prva faza kroz koju prolazi korpus je pretprocesiranje teksta, odnosno uklanjanje znakova interpunkcije, prevođenje velikih slova u mala, uklanjanje stop-reči i stemovanje ili lematizacija. Za srpski jezik je odrađeno stemovanje koristeći ručno stemovani korpus koji se može naći u Github repozitorijumu praktičnog dela. Za engleski i turski jezik odrađena je lematizacija

koristeći *Porter* stemer iz *nltk* biblioteke za engleski i *TurkishStemmer* za turski jezik.

Nad trening i testnim skupom svakog korpusa kreiraju se po dva vektorska modela (trening i testni) za svaki tip pomenutih vektorskih modela u poglavlju 3. Jedini izuzetak je model vrsta reči za turski jezik, koji nije implementiran usled nepostojanja adekvatnog alata za tagovanje turskih reči njihovim vrstama reči, kao i činjenice da je turski korpus naknadno obrađen sa motivacijom za poređenje rezultata nad drugim jezikom, koji nije srpski ili engleski. Za ovaj deo su u velikoj meri korišćene ugrađene metode iz *nltk* i *sklearn* paketa.

Za algoritme mašinskog učenja uglavnom su korišćene metode iz *sklearn* paketa, osim za NBSVM metodu čija se implementacija može naći ovde [12]. Postoji mogućnost pokrenuti aplikaciju sa ili bez podešavanja metaparametara klasifikatora, kao i (ne)računanja validacione greške. Za podešavanje metaparematara klasifikatora korišćena je slučajna mrežna pretraga u kombinaciji sa 5-slojnom unakrsnom validacijom sa sledećim vrednostima:

```
# SVM metoda
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf', 'linear']}
```

```
# NB metoda
param_grid = {'alpha': [0.5, 1.0, 1.5],
              'fit_prior': [True, False]}
```

```
# NBSVM metoda
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'alpha': [0.5, 1.0, 1.5],
              'beta': [0, 0.5, 1]}
```

```
# MLP metoda
param_grid = {'hidden_layers': [(50,50,50), (50,100,50), (100,)],
              'activation': ['tanh', 'relu'],
              'solver': ['sgd', 'adam', 'lbfgs'],
              'alpha': [0.0001, 0.05]}
```

Glava 6

Rezultati

Najpre će biti par reči o razlikama u obradi svakog korpusa u odnosu na objavljene radove, a potom i reči o empirijskom podešavanju i rezultatima. Na sledećim graficima prikazan je odnos tačnosti klasifikacije za svaki model na nivou jednog korpusa.

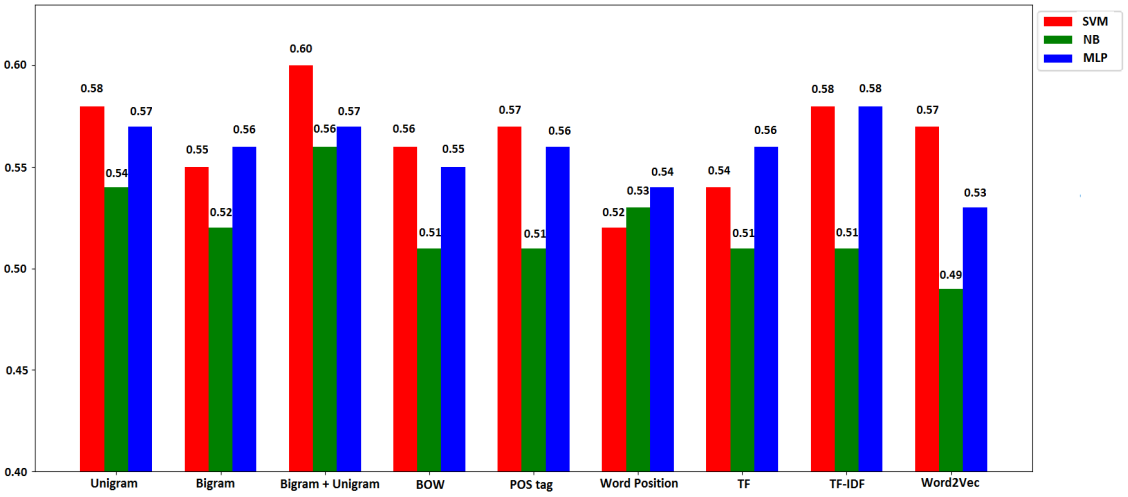
6.1 Korpus na srpskom jeziku

Najveća preciznost (60%) nad korpusom na sprskom jeziku sa pozitivnom, negativnom i neutralnom klasom dobijena je kod bigram-unigram modela sa SVM klasifikatorom. Rezultati kombinacija vektorskih modela sa metodama mašinskog učenja mogu se videti na Slici 6.1. Korišćena implementacija NBSVM metode nije obuhvatala višeklasni slučaj, pa je ovde ta metoda izuzeta. U poređenju sa rezultatima u radu [6], naš pristup daje lošije rezultate (-2.24%). Razlozi će biti objašnjeni u sledećem pasusu.

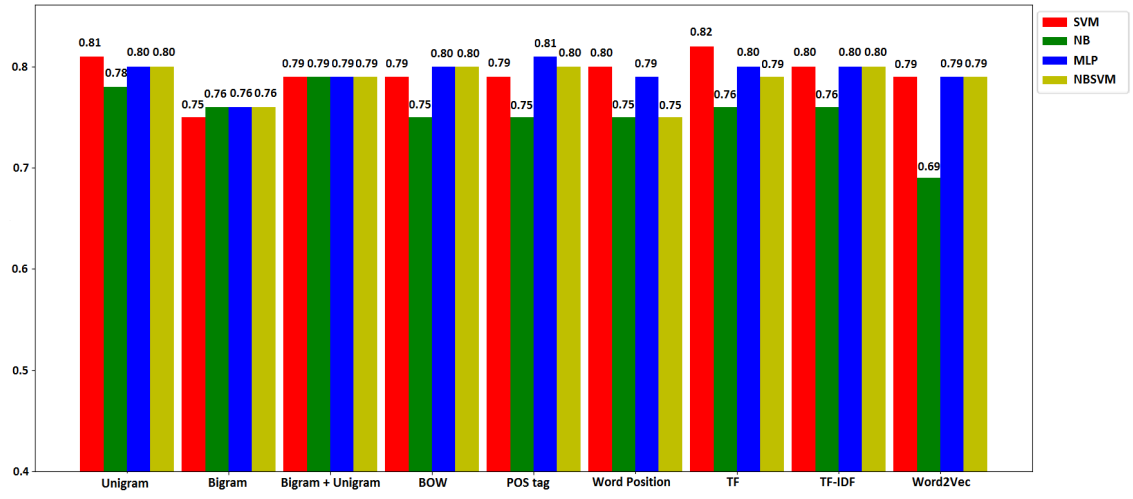
Najveća preciznost (82%) nad korpusom na sprskom jeziku sa pozitivnom i negativnom klasom dobijena je kod TF modela sa SVM metodom. To se može videti na Slici 6.2. U poređenju sa rezultatima u radu [6], naš pristup daje lošije rezultate (-3.55%). U tom radu korišćeni su sledeći principi:

- Odrađena je detaljna analiza optimalnog broja atributa, dok se ovde koristi 100000 najboljih atributa.
- Za preslikavanje sličnih reči se koristi stemovanje, a ovde lematizacija.
- Koristi se WEKA implementacija NBSVM algoritma.

- Koristi se ugnježdeno pretraživanje mreže.
- Nije naveden odnos podele na trening i testni skup.



Slika 6.1: Odnos tačnosti klasifikacije modela za srpski korpus sa tri klase

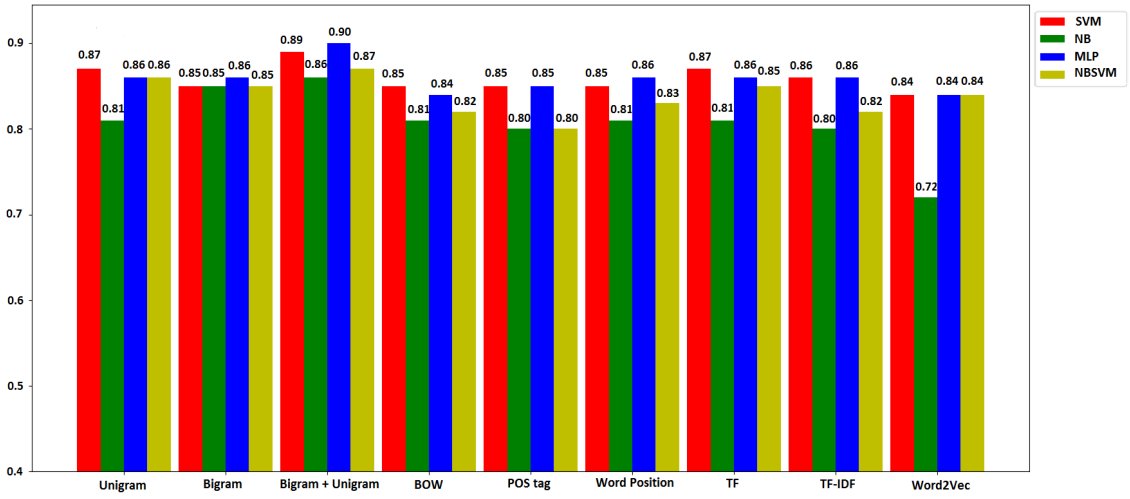


Slika 6.2: Odnos tačnosti klasifikacije modela za srpski korpus sa dve klase

6.2 Korpus na engleskom jeziku

Najveća preciznost (90%) nad korpusom na engleskom jeziku sa pozitivnom i negativnom klasom dobijena kod bigram-unigram modela sa MLP metodom. To se može videti na na Slici 6.3. U poređenju sa rezultatima u radu [5] može se videti da smo dosta bolji (+7.1%). U tom radu korišćeni su sledeći principi:

- Preskače se korak filtriranja teksta izbacivanjem stop-reči.
- Obrađuje se 16165 najboljih atributa, dok se ovde obrađuje 100000 najboljih atributa.
- Ne obrađuje se ceo korpus već podskup od 700 recenzija za svaku klasu.
- Nije naveden odnos podele na trening i testni skup.



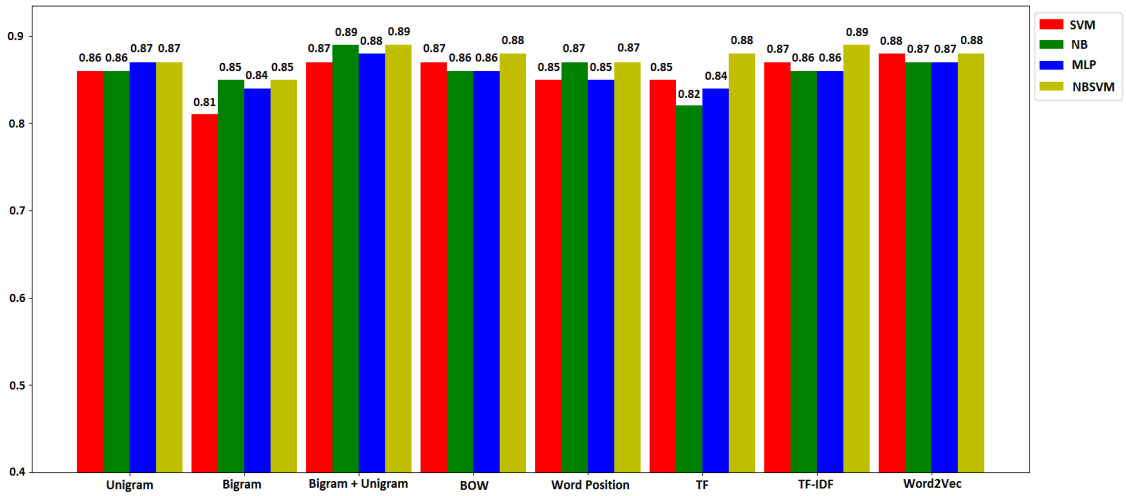
Slika 6.3: Odnos tačnosti klasifikacije modela za engleski korpus sa dve klase

6.3 Korpus na turskom jeziku

Najveća preciznost (89%) nad korpusom na turskom jeziku sa pozitivnom i negativnom klasom dobijena kod bigram-unigram modela sa NB i NBSVM metodom,

kao i kod TF modela sa NBSVM metodom. To se može videti na Slici 6.4. U poređenju sa rezultatima u radu [7], naš pristup daje bolje rezultate (+5.4%). U tom radu korišćeni su sledeći principi:

- Akcenat je bio na pravljenu leksikona prema sentimentu.
- Obrađuje se ceo korpus, a razmera trening i test skupa je 50:50. Iako poređenje nije na potpuno istom skupu, ne očekuju se veća odstupanja.



Slika 6.4: Odnos tačnosti klasifikacije modela za turski korpus sa dve klase

6.4 Empirijsko podešavanje i rezultati

Rezultati iz ovog rada prihvaćeni su za predstavljanje i štampanje u zborniku internacionalne INISTA 2020 konferencije [4]. Da bi se video efekat korišćenja različitih reprezentacija dokumenata i različitih algoritama mašinskog učenja, sprovedeni su eksperimenti nad svim mogućim kombinacijama. Za vektorsku reprezentaciju teksta korišćene su sledeće reprezentacije: vreća reči, bigrami, unigrami, bigrami sa unigramima, vektor vrsta reči, vektor pozicija reči, TF model, TF-IDF model i Word2Vec model. Od metoda mašinskog učenja korišćene su: metoda potpornih vektora, naivni Bajesov klasifikator, kombinacija metode potpornih vektora i naivnog Bajesovog

klasifikatora, kao i potpuno povezane neuronske mreže. Svi korišćeni korpusi su javno dostupni. Svaki korpus je podeljen na skup za treniranje i skup za testiranje u odnosu 80:20. Za podešavanje metaparematara svakog klasifikatora korišćena je slučajna mrežna pretraga u kombinaciji sa 5-slojnom unakrsnom validacijom nad skupom za treniranje. Tako generisani model se primenjivao nad novim skupom podataka (nad skupom za testiranje). Ceo kôd je javno dostupan i u skladu sa tim može služiti kao dobra osnova za dalji razvoj i unapređenje. Iz dobijenih rezultata možemo primetiti da model bigram sa unigramima uglavnom daje najbolje rezultate, što je i za očekivati jer takav model u obzir uzima i negaciju. U tabeli 6.1 koja sledi prikazani su rezultati eksperimenta (u procentima).

Tabela 6.1: Rezultati

Korpus		**SerbMR-3C**	**SerbMR-2C**	**CornellPD**	**HUMIR**
Unigrami	SVM	58	81	87	86
	NB	54	78	81	86
	MLP	57	80	86	87
	NBSVM	-	80	86	87
Bigrami	SVM	55	75	85	81
	NB	52	76	85	85
	MLP	56	76	86	84
	NBSVM	-	76	85	85
Unigrami sa bigramima	SVM	**60**	79	89	87
	NB	56	79	86	**89**
	MLP	57	79	**90**	88
	NBSVM	-	79	87	**89**
BOW	SVM	56	79	85	87
	NB	51	75	81	86
	MLP	55	80	84	88
	NBSVM	-	80	82	88
POS	SVM	57	79	85	-
	NB	51	75	80	-
	MLP	56	81	85	-
	NBSVM	-	80	80	-
Pozicija reči	SVM	52	80	85	85
	NB	53	75	81	87
	MLP	54	79	86	85
	NBSVM	-	75	83	87
BOW-TF	SVM	54	**82**	87	85
	NB	51	76	81	82
	MLP	56	80	86	84
	NBSVM	-	79	85	88
BOW-TF-IDF	SVM	58	80	86	87
	NB	51	76	80	86
	MLP	58	80	86	86
	NBSVM	-	80	82	**89**
Word2Vec	SVM	57	79	84	88
	NB	49	69	72	87
	MLP	53	79	84	87
	NBSVM	-	79	84	88

Glava 7

Zaključak

U današnje vreme susrećemo se sa ubrzanim rastom količine tekstualnih dokumenata koje treba obraditi i razumeti. Shodno ovoj tendenciji, nastala je i potreba za efikasnom obradom i izvlačenjem informacija iz tih dokumenata. Ova potreba, kao i čovekovo neretko oslanjanje na mišljenja drugih ljudi koji su svoja zapažanja podelili na društvenim mrežama, dovela je do nastanka ideje klasifikacije tekstualnih dokumenata prema sentimentu.

Cilj ovog master rada bila je računarska klasifikacija teksta prema sentimentu primenom metoda mašinskog učenja. Ideja rada jeste da se primenom različitih kombinacija vektorske reprezentacije teksta i metoda mašinskog učenja ostvari što bolja preciznost ispravnosti klasifikacije. U ovom radu opisan je kompletan postupak dobijanja informacije o sentimentu teksta. Detaljno su opisane sve korišćene vektorske reprezentacije teksta: vreća reči, bigrami, unigrami, bigrami sa unigramima, vektor vrsta reči, vektor pozicija reči, TF model, TF-IDF model i Word2Vec model. Opisan je teorijski aspekt korišćenih metoda mašinskog učenja: metode potpornih vektora, naivnog Bajesovog klasifikatora, kombinacije metode potpornih vektora i naivnog Bajesovog klasifikatora, kao i potpuno povezanih neuronskih mreža. Praktični deo ovog rada podrazumeva aplikaciju u programskom jeziku *Python* koja obrađuje korpuse dokumenata na srpskom, engleskom i turskom jeziku. Aplikacija predstavlja automatizovanu verziju kompletnog postupka od sirove obrade teksta do dobijanja grafičkog prikaza statistike izabranog korpusa.

Rad predstavlja osnovu za analizu teksta prema sentimentu sa velikim spektrom kombinacija vektorskih reprezentacija teksta i metoda mašinskog učenja. Doprinos rada se ogleda i u poboljšanoj preciznosti klasifikacije teksta prema sentimentu na engleskom i turskom jeziku. Dalja istraživanja mogu biti fokusirana na poboljšanju

preciznosti klasifikacije. Jedan od načina može biti pravljenje skupa ručnih pravila za obradu teksta, koja mogu biti zajednička za sve jezike, ali i specifična za svaki jezik. Takođe, istraživanje se može upotpuniti korišćenjem drugih resursa, kao što je npr. *Word Net*, koji predstavlja bazu semantičkih relacija reči.

Bibliografija

[1] Jelena Graovac. "Prilog metodama klasifikacije teksta: Matematički modeli i primena". *Doktorska disertacija.* Matematički fakultet, Beograd, 2014.

[2] Jelena Graovac. "Text categorization using n-gram based language independent technique". *Naučni rad.* Matematički fakultet, Beograd, 2014.

[3] Mladen Nikolić i Anđelka Zečević. "Mašinsko učenje". *Skripta.* Matematički fakultet, Beograd, 2019.

[4] Jelena Graovac, Marija Radović, and Berna Altınel Girgin. "ML-SPD: Machine Learning based Sentiment Polarity Detection". *International Conference on INnovations in Intelligent SysTems and Applications (INISTA).* Novi Sad, 2020.

[5] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up? Sentiment Classification using Machine Learning Techniques". *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002).* Association for Computational Linguistics, 2002, pp. 79–86. DOI: 10.3115/1118693.1118704. URL: https://www.aclweb.org/anthology/W02-1011.

[6] Vuk Batanović, Boško Nikolić, and Milan Milosavljević. "Reliable Baselines for Sentiment Analysis in Resource-Limited Languages: The Serbian Movie Review Dataset". *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16).* European Language Resources Association (ELRA), 2016, pp. 2688–2696.

[7] Alaettin Ucan, Behzad Naderalvojoud, Ebru Akcapinar Sezer, and Hayri Sever. "SentiWordNet for New Language: Automatic Translation Approach". *12th International Conference on Signal-Image Technology Internet-Based Systems.* 2016.

[8] *Korpus na srpskom jeziku.* https://github.com/vukbatanovic/SerbMR.

[9] *Korpus na engleskom jeziku.* `http://www.cs.cornell.edu/people/pabo/movie-review-data`.

[10] *Korpus na turskom jeziku.* `http://humirapps.cs.hacettepe.edu.tr/tsad.aspx`.

[11] *Github stranica sa praktičnim delom.* `https://github.com/mradovic23/ML-SPD`.

[12] *Github stranica sa NBSVM implementacijom.* `https://github.com/Joshua-Chin/nbsvm`.

[13] *Zvanična dokumentacija Python biblioteke za mašinsko učenje.* `https://scikit-learn.org`.

[14] *MonkeyLearn - platforma za treniranje modela mašinskog učenja.* `https://monkeylearn.com/sentiment-analysis`.

[15] *Towards Data Science - servis koji se bavi obradom podataka.* `https://towardsdatascience.com`.

[16] *KD nuggets - servis koji se bavi obradom podataka.* `https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html`.

[17] *Domo - servis za organizaciju podataka.* `https://www.domo.com/solution/data-never-sleeps-6`.

[18] *Alteryx - servis koji se bavi obradom podataka.* `https://community.alteryx.com`.

Biografija autora

Marija Radović je rođena 23. aprila u Kruševcu. Osnovnu školu „Ivo Lola Ribar" u Aleksandrovcu završila je 2009. kao đak generacije. Srednju školu „Gimnazija" u Kruševcu na prirodno-matematičkom smeru završila je 2013. sa odličnim uspehom. Te godine upisala je Matematički fakultet u Beogradu na smeru „Računarstvo i informatika". Osnovne studije završila je 2018. godine, kada je upisala i master studije na istom smeru. Pri završetku studija počela je da radi u firmi „RT-RK" u automobilskoj industriji. Oblast kojom se bavi je pisanje, testiranje i optimizacija softvera na namenskom čipu za računarski vid, koji je deo sistema za autonomnu vožnju. U slobodno vreme bavi se sportovima kao što su skijanje i planinarenje.