TakeLab

Laboratorij za analizu teksta i inženjerstvo znanja
Text Analysis and Knowledge Engineering Lab
Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Unska 3, 10000 Zagreb, Hrvatska

SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4753

Otkrivanje poruka neprimjerenog sadržaja u mrežnom razgovoru

Filip Hrenić

Zagreb, lipanj 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 11. ožujka 2016.

ZAVRŠNI ZADATAK br. 4753

Pristupnik: **Filip Hrenić (0036477582)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Otkrivanje poruka neprimjerenog sadržaja u mrežnom razgovoru**

Opis zadatka:

Razvoj interneta doprinio je bržoj i učinkovitijoj komunikaciji, ali je, zbog anonimnosti koju nudi korisnicima, također povećao broj nepoželjnih aktivnosti. Objavljivanje poruka neprimjerenog sadržaja, uključivo uvredljivog i neželjenog sadržaja, najčešći je oblik nepoželjne aktivnosti, koji je moguće spriječiti postupcima za automatsko otkrivanje takvih poruka, posebice postupcima temeljenima na strojnome učenju.
U okviru završnoga rada potrebno je proučiti postupke za klasifikaciju teksta temeljene na nadziranom strojnom učenju te postojeće postupke za otkrivanje teksta neprimjerenog sadržaja. Razraditi model za otkrivanje poruka uvredljivog sadržaja u mrežnom razgovoru više korisnika temeljen na strojnom učenju. Isprobati nekoliko modela strojnog učenja i osmisliti nekoliko značajki za prikaz teksta. Razviti programsku implementaciju modela te provesti iscrpno vrednovanje na odgovarajućem ručno označenom skupu podataka, uključivo analizu značajki, usporedbu s referentnim modelima i statističku obradu rezultata. Radu priložiti izvorni i izvršni kod razvijenog sustava, označene skupove podataka i potrebnu dokumentaciju te citirati korištenu literaturu.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 17. lipnja 2016.

Mentor:

Doc. dr. sc. Jan Šnajder

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:

Prof. dr. sc. Siniša Srbljić

Zahvala tvrtki SofaScore[1] *na ustupljenim podacima*

[1] `http://www.sofascore.com/`

Sadržaj

1. Uvod

Razvoj interneta potaknuo je ubrzani razvoj društvenih mreža koje su se pokazale kao učinkovit način za komunikaciju između ljudi. Svatko je slobodan izraziti se na svoj način te iznijeti svoje mišljenje. No anonimnost korisnika u mrežnim razgovorima može dovesti do neželjenih posljedica. Upravo radi skrivenog identiteta ljudi često ne obraćaju pozornost na svoje objave. Anonimni korisnik može objavom neprimjerenog sadržaja, kao što su to primjerice uvredljive i neželjene poruke, naštetiti ostalim korisnicima. Najčešći primjeri takvog sadržaja su psovke, uvrede, *spam* poruke i tako dalje.

Zbog toga je otkrivanje poruka nepoželjnog sadržaja jedan od najizazovnijih problema u društvenim mrežama. Svakodnevno se razvijaju nove metode kojima bi bilo moguće na efikasan način otkriti takve poruke. Jednostavno pretraživanje sadržaja poruke kako bi se pronašle određene fraze nije dovoljno. No izrada automatiziranog klasifikatora koji bi uspješno raspoznavao sve poruke je praktički nemoguća. Zbog toga se razvijaju postupci koji su većinom bazirani na metodama strojnog učenja. Ideja je dati klasifikatoru skup prethodno označenih *dobrih* i *loših* poruka na kojima će on učiti i moći s određenom točnošću odlučivati o novim porukama te ih klasificirati kao *dobre* ili *loše*.

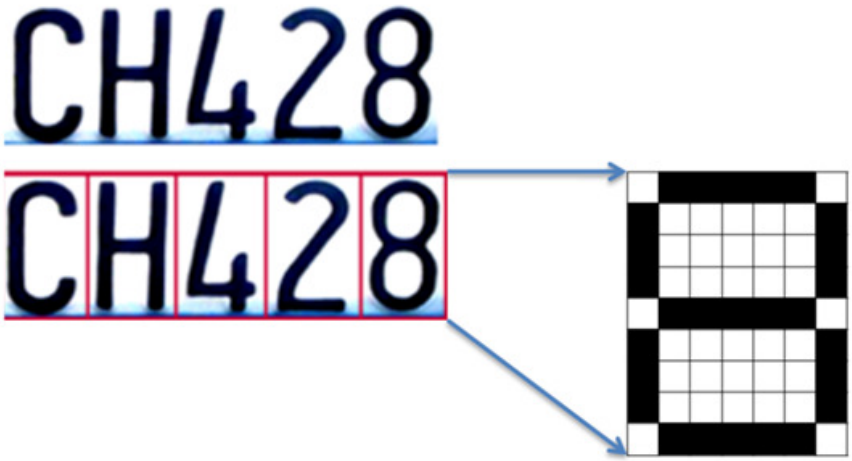
U ovom radu predstavljen je jedan od mogućih načina kako na automatizirani način klasificirati poruke. Poruke su preuzete iz javnog razgovora na internetskoj stranici za objavu sportskih rezultata te su nepoželjne poruke većinom vulgarne ili spam. Sama klasifikacija poruka napravljena je na temelju njihovog sadržaja uporabom raznih algoritama strojnog učenja (aktivno učenje, logistička regresija). Time je omogućeno da se na temelju prethodno označenih poruka za neku novu poruku odredi vjerojatnost da je ista nepoželjna.

Zbog prethodno navedenog problema obrade više jezika odjednom, u ovom radu su se filtrirale samo poruke korisnika koji imaju postavljen engleski jezik kao zadani. Šum ostalih jezika bio je minimalan. Model za klasifikaciju implementiran je pomoću programske knjižice GraphLab [3] te su dobiveni zadovoljavajući rezultati.

2. Klasifikacija

Klasifikacija se pojavljuje u mnogim ljudskim aktivnostima. Ona općenito pokriva bilo kakav kontekst u kojem se donosi odluka na temelju trenutno dostupnih informacija. Postupak klasificiranja je neka formalna metoda kojom donosimo takve odluke u novim situacijama.[1]

U strojnom učenju bismo mogli definirati klasifikaciju kao postupak u kojem određujemo kojoj potkategoriji pripada novi podatak, na temelju njegovih obilježja i skupa poznatih podataka. Često kad govorimo o klasifikaciji govorimo o raspoznavanju uzoraka (engl. *pattern recognition*). Kao primjer možemo uzeti OCR[2] u kojemu se radi klasifikacija teksta na slici, odnosno, na temelju obilježja i izgleda znaka se određuje koje je to slovo.

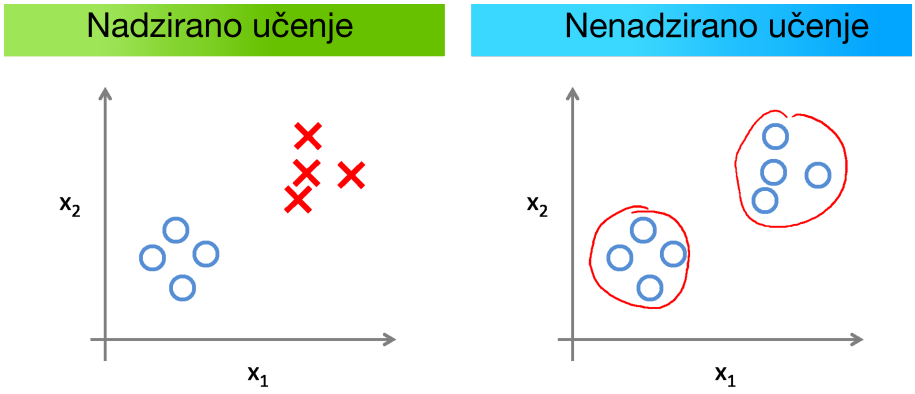


Slika 2.1: Optičko prepoznavanje znakova[3]

Klasifikacija se može podijeliti na dva područja, nadzirano (engl. *supervised*) i nenadzirano (engl. *unsupervised*) učenje. Nadzirano učenje podrazumijeva da imamo podatke koji su označeni (svrstani u klase), pa želimo formirati pravilo kojim bismo novi

[1] Prilagođeno iz Alpaydin [1].
[2] Optičko prepoznavanje znakova (engl. *Optical Character Recognition*).
[3] Preuzeto s `http://www.nedapidentification.com/`.

podatak svrstali u jednu od klasa. Nenadziranim učenjem najčešće želimo neoznačene podatke svrstati u grozdove (engl. *cluster*) s drugim podacima kako bismo uočili njihovu međusobnu povezanost.

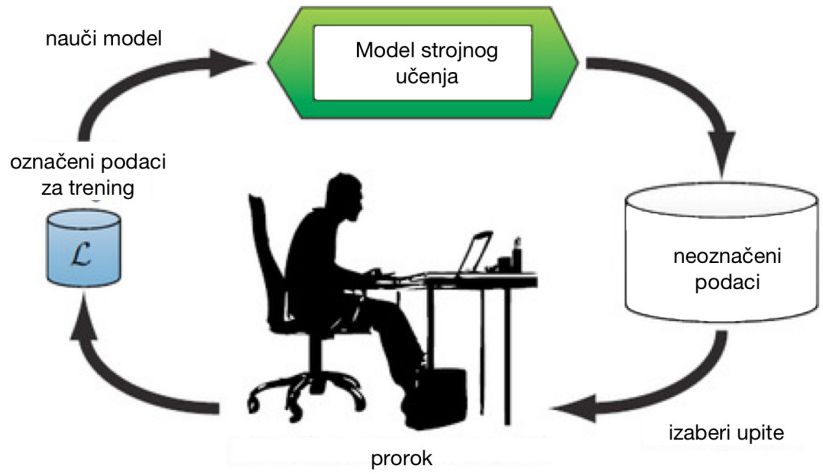


Slika 2.2: Nadzirano i nenadzirano učenje[4]

Polunadzirano (engl. *semi–supervised*) učenje spada između te dvije metode, a obično se radi o maloj količini označenih te puno većoj količini neoznačenih podataka. Takve metode su vrlo korisne kad je teško označiti podatke ili je teško nabaviti označene podatke, dok je pristup neoznačenim podacima lakši.[5]

Aktivno učenje (engl. *active learning*) je specijalni slučaj polunadziranog učenja u kojem algoritam interaktivno ispituje korisnika kako bi označio podatke. Glavna pretpostavka ove metode je da ako dopustimo algoritmu da bira podatke iz kojih uči, ostvarit će bolje rezultate s manje treninga. Više o ovoj metodi može se pronaći u Settles [5].

[4]Preuzeto s `http://oliviaklose.com/`.
[5]Definicija iz Olivier Chapelle [4].



Slika 2.3: Aktivno učenje[6]

Induktivno učenje (engl. *inductive learing*) je metoda kojom iz početnih podataka induktivno saznajemo nove. Nakon toga iz početnih i novih podataka induktivno saznajemo nove, i tako dalje.

[6]Preuzeto iz Settles [5].

3. Logistička regresija

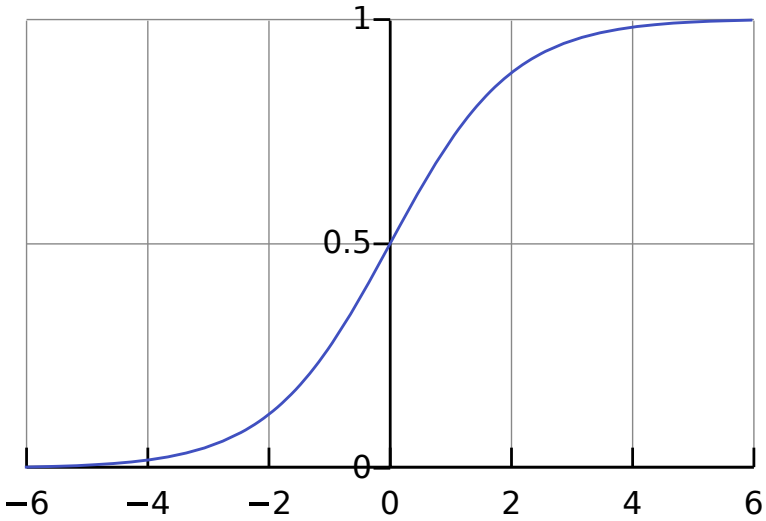
Logistička regresija (engl. *logistic regression*) je metoda za klasifikaciju podataka u diskretne klase. Primjerice, mogli bismo koristiti logističku regresiju kako bismo klasificirali email kao normalan ili *spam*. Detaljnije o logističkoj regresiji može se pronaći u knjizi David W. Hosmer [2]. Daljnji opis ove metode preuzet je i prilagođen iz iste knjige. Model koji je korišten u implementaciji koristi ovu metodu, pa je potrebno definirati na koji način funkcionira logistička regresija.

3.1. Funkcija troška

Svaki podatak je određen vektorom značajki $\mathbf{x}$. Elementi vektora su vrijednosti obilježja tog podatka. Klasifikator je određen vektorom koeficijenata θ te njime možemo procijeniti oznaku novog podatka koristeći

$$h_\theta(\mathbf{x}) = g(\theta^{\mathbf{T}}\mathbf{x})$$
$$g(z) = \frac{1}{1+e^{-z}}$$

gdje je g tzv. logistička funkcija, a h je funkcija klasifikatora koji određuje oznaku vektora. Želimo da klasifikator vraća vjerojatnost da je oznaka podatka 1. Zato treba primijetiti da vrijedi $g(z) \in [0,1] \quad \forall z \in \mathbb{R}$ (vidi sliku 3.1).



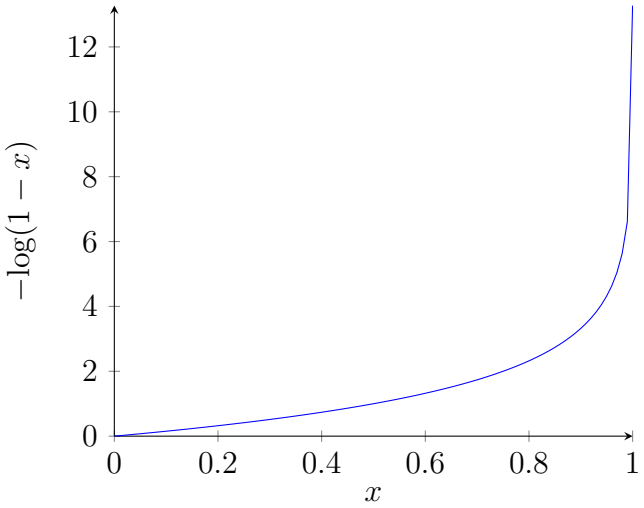
Slika 3.1: Logistička funkcija $g(z)$

Zanimljivo je uočiti da će naš klasifikator svim podacima za koje vrijedi $h\theta(\mathbf{x}) \geq 0.5$ dati oznaku 1, a za $h_\theta(\mathbf{x}) < 0.5$ dati oznaku 0. Odnosno, pogledamo li opet sliku 3.1 uočavamo da će podaci za koje vrijedi $\theta^{\mathbf{T}}\mathbf{x} \geq 0$ biti označeni sa 1, dok će ostali biti označeni sa 0.

Ako su nam podaci za trening klasifikatora $\left\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\right\}$, gdje je $x^{(i)}$ i–ti podatak, $y^{(i)}$ oznaka i–tog podatka, a m broj podataka, onda definiramo funkciju cijene

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log h_\theta(x^{(i)}) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

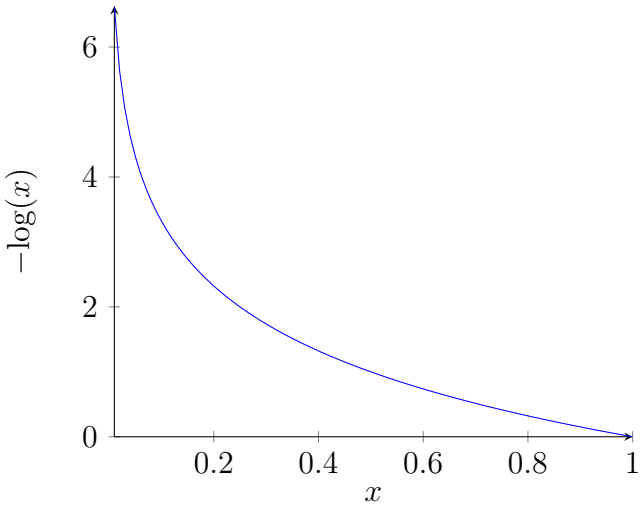
Svaki član sume sastoji se od dva pribrojnika. Pošto je oznaka y binarna, ona može biti ili 0 ili 1. Ako je podatak označen negativno, odnosno ako je $y^{(i)} = 0$, onda prvi pribrojnik postaje 0 te član sume postaje $-\log(1-h_\theta(x^{(i)}))$. Ta funkcija prikazana je na slici 3.2.



Slika 3.2: $f(x) = -\log(1-x)$

Vidimo da ova funkcija za oznaku klasifikatora $h_\theta(x^{(i)}) \approx 0$ daje malen trošak te sve više raste što je oznaka bliža 1.

Dok u suprotnom slučaju, kad je oznaka podatka $y^{(i)} = 1$, onda drugi pribrojnik postaje 0 te član sume postaje $-\log h_\theta(x^{(i)})$. Ta funkcija prikazana je na slici 3.3.



Slika 3.3: $f(x) = -\log(x)$

Vidimo da ova funkcija za oznaku klasifikatora $h_\theta(x^{(i)}) \approx 1$ daje malen trošak te sve više raste što je oznaka bliža 0.

To je upravo ono što smo i htjeli postići ovom funkcijom: da daje veliki trošak za krivo određene oznake.

3.2. Gradijentni spust

Cilj klasifikatora je pronaći takve parametre θ da funkcija $J(\theta)$ ima minimum za dane podatke. Postoje mnogi načini kako naći taj minimum, no razmotrit ćemo algoritam gradijentnog spusta (engl. *gradient descent*). On pronalazi lokalni minimum funkcije na sljedeći način. Parametar θ se u svakoj iteraciji algoritma mijenja

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

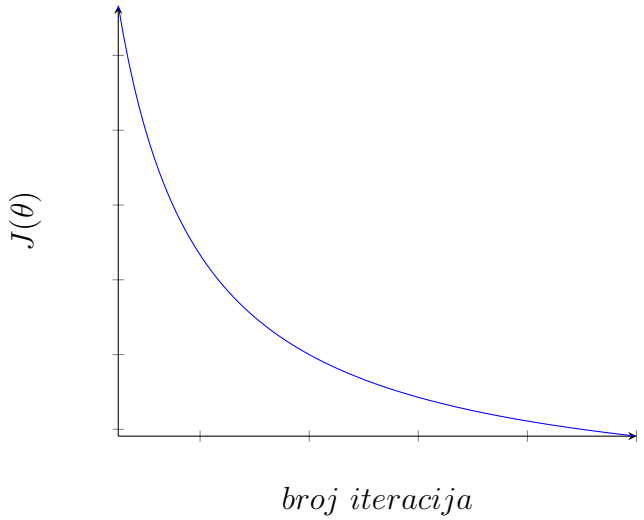
gdje je α stopa učenja (engl. *learning rate*), proizvoljan mali broj koji se odredi na početku. Lako možemo pronaći navedenu derivaciju koja nam treba za promjenu parametra

$$\frac{\partial}{\partial \theta} J(\theta) = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(j)}$$

Algoritam završava kad postane $\Delta J(\theta) < \varepsilon$ za neki ε ili kad se dostigne maksimalni broj iteracija.

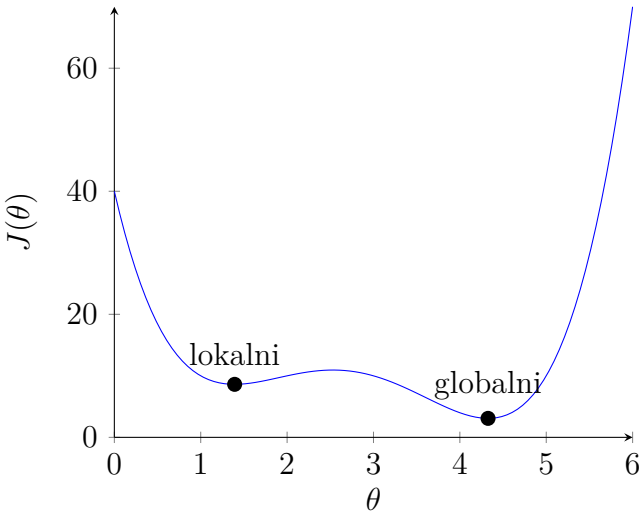
3.3. Mane algoritma

Nažalost, ovaj algoritam ima i mane, kao što je primjerice biranje parametra α. Biranje premalog α dovodi do prespore konvergencije, dok preveliki α može dovesti do toga da funkcija troška nikad ne konvergira. Parametar α treba biti podešen tako da vrijednost funkcije troška pada u svakoj iteraciji (vidi sliku 3.4).



Slika 3.4: Funkcija troška

Drugi veći problem ovog algoritma je zaglavljivanje u lokalnom optimumu umjesto u globalnom (vidi sliku 3.5). No kod logističke regresije funkcija cijene je konveksna te ima samo jedan optimum. Zbog toga se algoritam ne može zaglaviti u lokalnom optimumu.



Slika 3.5: Zaglavljivanje u lokalnom optimumu

4. Baza poruka

4.1. Opis

Za potrebe ovog rada korištene su poruke iz razgovora na internetskoj stranici *SofaScore*[1]koja prikazuje sportske rezultate. Baza poruka je prvotno bila zapisnik događaja u kronološkom poretku. Zapisa je bilo otprilike 3.500.000.

Događaji se mogu podijeliti u dvije skupine: dodana je poruka ili je izvršena akcija nad porukom. Akciju može izvršiti bilo koji korisnik (označiti poruku kao korisnu ili ju prijaviti kao neželjenu) ili ju može izvršiti samo moderator (obrisati poruku ili ju postaviti na neko mjesto gdje ju svi vide).

Zapisnik događaja bio je u formatu *csv* koji je trebalo obraditi da se može iskoristiti za izgradnju klasifikatora. Zbog toga je zapisnik pretvoren u bazu poruka. To je napravljeno pomoću baze podataka MySQL i skripti u Python. Poruke su filtrirane u bazi na temelju zemlje podrijetla.

Zapisnik događaja sadrži zapise u obliku:

- **id**: identifikator akcije,
- **current_user**: korisnik koji je napravio akciju,
- **cmd**: akcija koja je napravljena,
- **text**: ako je akcija *message* onda je ovdje tekst poruke,
- **affecteduser**: ako akcija nije *message* onda je ovdje korisnik kojeg je zahvatila akcija,
- **affectedmessage**: ako akcija nije *message* onda je ovdje poruka nad kojom je učinjena akcija,
- **language**: jezik zemlje iz koje je *current_user* i
- **mcc**: mobile country code.

Primjer zapisa iz zapisnika događaja:

[1] `http://www.sofascore.com`

1454511300856,postgres,message,1 gol please„,in,510

Zapisnik događaja je zatim transformiran u bazu poruka. To znači da se iz kronološkog poretka događaja pretvori u zapise o porukama gdje za svaku poruku znamo koliko je puta koja akcija nad njom izvršena. Nakon toga više ne znamo kontekst poruke. Nije nam poznato koje su joj poruke prethodile, koje akcije su slijedile i slično. No to ni nije bitno jer će sustav poruku klasificirati na temelju njenog sadržaja, ne uzimajući u obzir kontekst. Nakon pretvorbe u bazi se nalaze poruke s pripadnim atributima:

- **id**: identifikator poruke,
- **text**: tekst poruke,
- **language**: jezik zemlje iz koje je *current_user*,
- **mcc**: mobile country code,
- **feature**: da li je poruka izdvojena,
- **report**: broj koliko puta je poruka prijavljena,
- **upvote–message**: broj dobrih glasova za poruku,
- **remove–message**: da li je poruka uklonjena i
- **permaban**: da li je korisnik radi te poruke dobio ban.

Primjer zapisa poruke u bazi

1454513419827,Come on Zavrc score over 2.5!!!,en,311,0,0,0,0,0

4.2. Problemi

Ubrzo je uočeno da korisnici ne označuju nepoželjne poruke konzistentno. To je i očekivano jer su ljudi različiti, ne smetaju svima iste stvari. Ili pak korisnici mogu prijavljivati dobre poruke bez ikakvog valjanog razloga. Zbog toga se poruke ne mogu klasificirati s obzirom na akcije. Ovo je dovelo do ogromnog problema. Akcije koje su izvršene nad porukom trebale su ju i klasificirati no to je sad bilo nemoguće.

Pošto je stranica posjećena od ljudi diljem svijeta, to znači da su poruke na nekoliko desetaka jezika. To je riješeno filtriranjem poruka po zemlji iz koje je korisnik. Filtriranjem su ostavljene samo poruke od korisnika koji imaju postavljen engleski jezik kao zadani. Time naravno nije u potpunosti uklonjen problem, no ostalo je zanemarivo malo poruka na ostalim jezicima. Filtriranjem je nastala baza od 200.000 neoznačenih poruka.

Potrebno je poruke označiti kako bi klasifikacijski model mogao dalje zaključivati o novim porukama te njih klasificirati. No ručno označavanje svih poruka uzelo bi previše vremena te je bilo potrebno pronaći drukčiju metodu.

5. Algoritam

Bilo je potrebno osmisliti algoritam kojim bi na najefikasniji način označio sve poruke. Smišljeni algoritam nastao je u razgovoru s menotorom te je bio spoj samoučenja i aktivnog učenja. Pseudokod algoritma 1 dan je u nastavku.

Algoritam kao ulaz dobije listu označenih primjera i listu neoznačenih. Njegov zadatak je da označi sve neoznačene primjere te ih vrati kao rezultat.

Prvo se lista označenih podjeli na dva dijela, jedan dio služi kao podaci za trening klasifikatora dok drugi dio služi za njegovu validaciju. Nakon toga se za svaki podatak u neoznačenim primjerima odredi vjerojatnost p da oznaka bude 1. Ako je oznaka dovoljno velika, onda se podatak označi sa 1 ili ako je oznaka dovoljno mala, onda se podatak označi sa 0. Za one podatke za koje je klasifikator na granici odluke (vjerojatnost je oko 0.5) se pita korisnika. Sve ostale podatke se doda natrag u listu neoznačenih podataka. Ovaj postupak se ponavlja sve dok ima neoznačenih primjera.

Algorithm 1 Self–training + Active learning

```
1: procedure OZNACI(o, n)                          ▷ o – označeni, n – neoznačeni
2:     t ← 0.1                                                          ▷ prag
3:     while ¬empty(n) do                                    ▷ ima neoznačenih
4:         train, test ← split(o)                              ▷ podjeli nasumično
5:         m ← model(train, test)                  ▷ napravi klasifikacijski model
6:         s ← lista()
7:         for podatak in n do
8:             p ← vjerojatnost(model, podatak)
9:             if p < t then
10:                oznaci(podatak, 0)
11:                dodaj(o, podatak)
12:            else if p > 1.0 − t then
13:                oznaci(podatak, 1)
14:                dodaj(o, podatak)
15:            else if 0.5 − t < p < 0.5 + t then
16:                oznaka = pitajKorisnika(podatak)
17:                oznaci(podatak, oznaka)
18:                dodaj(o, podatak)
19:            else
20:                dodaj(s, podatak)
21:            end if
22:        end for
23:        n ← s
24:    end while
25:    return o                                              ▷ svi su označeni
26: end procedure
```

6. Implementacija

6.1. Označavanje

Ručno je označeno 4000 poruka. Od toga ih je 2000 iskorišteno kao podaci za treniranje, dok je preostalih stavljeno na stranu kako bi se iskoristili za vrednovanje klasifikatora (test podaci). Time je ostalo 200.000 neoznačenih poruka. One su označene uz pomoć algoritma 1. Za to je bilo potrebno otprilike 6–7 sati pošto je za neodlučive primjere model pitao korisnika (autor rada) za oznaku. Sve poruke su bile označene nakon 15–20 iteracija. Broj upita korisniku u svakoj iteraciji padao je od početne prema posljednjoj iteraciji.

6.2. Klasifikacija

Klasifikaciju poruka implementirana je pomoću Pythonove programske knjižnice GraphLab [3] koja pruža mnoge algoritme i strukture podataka vezane uz strojno učenje i analizu teksta. Kao obilježja na temelju kojih se određuje klasa poruke uzet je vektor riječi koje se nalaze u tekstu poruke.

U nastavku je dan isječak koda kojim se gradi klasifikator za dane podatke

```
1 import graphlab
2
3 def ucitaj(naziv):
4     podaci = graphlab.SFrame(naziv)
5     podaci['vr'] = graphlab.text_analytics.count_words(podaci['tekst'])
6     return podaci
7
8 train_podaci = ucitaj('train_podaci.csv')
9 test_podaci  = ucitaj('test_podaci.csv')
10
11 klasifikator = graphlab.logistic_classifier.create(
```

```
12      dataset = train_podaci,
13      features = ['vr'],
14      target = 'oznaka'
15      )
16
17 klasifikator.predict(test_podaci, output_type='probability')
```

buildmodel.py

Ovime smo izgradili klasifikator koji može za novi podatak odrediti vjerojatnost da je njegova oznaka 1, odnosno da je poruka u redu. Funkcija *ucitaj* će učitati podatke naredbom *graphlab.SFrame* iz datoteke *csv* te stvoriti vektor riječi. Taj vektor riječi predstavlja značajke poruke. Možemo ga zamisliti kao mapu koja za svaku riječ (ključ) kao vrijednost ima koliko puta se ta riječ ponavlja u poruci. Pretpostavlja se da učitani podaci imaju vrijednosti *tekst* i *oznaka*. Posljednjom naredbom smo zapravo izgradili klasifikator. Opcije koje su korištene su:

- **dataset**: koje podatke klasifikator koristi za trening,
- **features**: lista značajki koje određuju oznaku podatka i
- **target**: što želimo da model određuje

Ovaj klasifikator može se ugraditi u postojeći sustav za komunikaciju preko interneta te ga iskoristiti kako bi se poruke filtrirale prije nego su objavljene.

7. Rezultati

Validacijom klasifikatora dobiveni su zanimljivi rezultati. Kao set podataka za validaciju uzeti su podaci označeni aktivnim učenjem. Oni su poslužili kako bi se odredio prag koji daje najbolje rezultate nad tim podacima. Kad su određeni svi parametri klasifikatora, on je vrednovan na testnim podacima koji su bili ručno označeni te nisu bili prethodno korišteni ni u jednom koraku označavanja ili klasifikacije.

Dobivaju se različiti rezultati ovisno o pragu odluke klasifikatora. Naime, neka je prag klasifikatora T, te neka on za neki podatak odredi vjerojatnost p da je oznaka tog podatka 1. Onda on o oznaci poruke odlučuje ovisno o tome je li $p < T$. Ako je $p < T$ onda se podatak označi s 0, a inače se označi s 1. Pa prema tome vidimo da ako klasifikator radi deterministički, odnosno ako uvijek za isti podatak daje istu vjerojatnost, onda oznaka podatka ovisi samo o pragu T.

7.1. Grafovi

Da bismo mogli objasniti rezultate, potrebno je prvo objasniti neke pojmove. Svi podaci imaju svoju stvarnu oznaku koja može biti 1 ili 0, odnosno pozitivna ili negativna. Klasifikator će joj odrediti pretpostavljenu oznaku. Na temelju toga možemo podijeliti klasificirane podatke na sljedeće četiri grupe (tablica 7.1): ispravno pozitivni (engl. *true positives*), ispravno negativni (engl. *true negatives*), pogrešno pozitivni (engl. *false positives*) i pogrešno negativni (engl. *false negative*).[1]

Tablica 7.1: Podjela s obzirom na oznaku

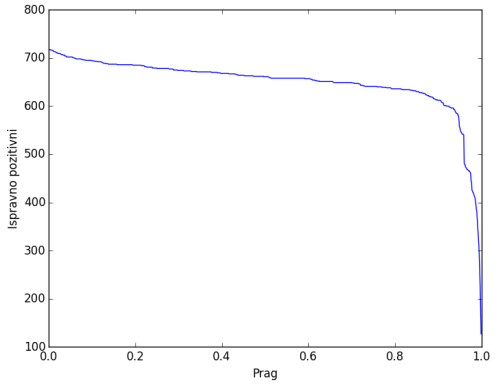
		Pretpostavljena oznaka	
		Pozitivna	**Negativna**
Stvarna oznaka	**Pozitivna**	*Ispravno pozitivni*	*Pogrešno negativni*
	Negativna	*Pogrešno pozitivni*	*Ispravno negativni*

[1] U daljnjem tekstu koriste se oznake **TP**, **TN**, **FP** i **FN** za ispravno pozitivne, ispravno negativne, pogrešno pozitivne i pogrešno negativne.

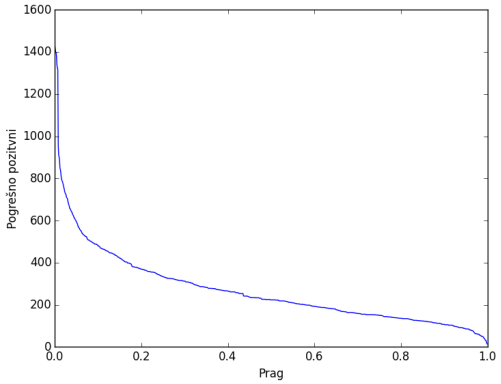
Istinite (engl. *true*) oznake dobivaju oni podaci kojima je točno određena oznaka. Tako primjerice neki podatak spada u *pogrešno negativne* ako mu je pogrešno određena negativna oznaka. To znači da je njegova stvarna oznaka bila pozitivna, no klasifikator ga je označio negativnom oznakom.

U savršenom slučaju klasifikator ne bi uopće imao lažnih oznaka (engl. *false*), no to je u realnosti gotovo nemoguće postići. Kao što je i rečeno, klasifikator određuje oznaku u ovisnosti o pragu. Na sljedećih nekoliko grafova prikazano je kako se kreće broj svake od klasa u ovisnosti o pragu odluke.

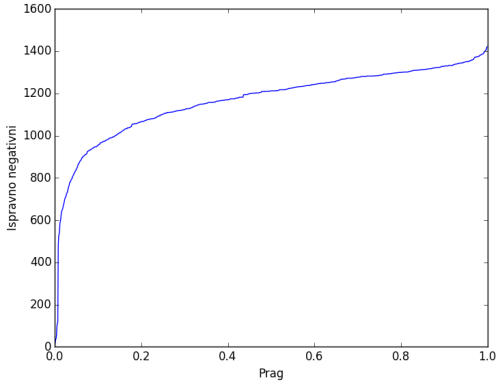
Valja napomenuti da se skup podataka za validaciju sastoji od 719 719 podataka s pozitivnom oznakom te 1421 podatak s negativnom oznakom.



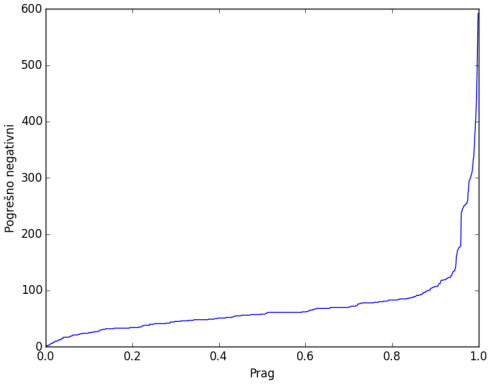
(a) Ispravno pozitivni



(b) Pogrešno pozitivni



(c) Ispravno negativni



(d) Pogrešno negativni

Slika 7.1: Promjena oznaka s obzirom na prag odluke

Iz grafova se jasno vidi da kako raste prag odluke klasifikatora tako raste i broj negativno označenih podataka, a pada broj pozitivno označenih. To je i u skladu s očekivanim pošto je u početku prag 0.0 pa su i svi podaci označeni pozitivno, dok je

na kraju prag 1.0 te su svi podaci označeni negativno.

No ovi grafovi malo govore o učinkovitosti klasifikatora zbog toga što su vrijednosti apsolutne. Bolja mjera učinkovitosti su primjerice odziv (engl. *recall*), specifičnost (engl. *specificity*), točnost (engl. *accuracy*), preciznost (engl. *precision*), F_1 mjera (engl. F_1 *score*) te Matthewsov korelacijski koeficijent (engl. *Matthews correlation coefficient*).

Odziv[2] se odnosi na sposobnost klasifikatora da točno odredi podatke koji su označeni pozitivno. U našem slučaju je to postotak poruka koje klasifikator označi pozitivno od skupa svih poruka koje su pozitivne. Odnosno:

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

Specifičnost se odnosi na sposobnost klasifikatora da točno odredi podatke koji su označeni negativno. U našem slučaju je to postotak poruka koje klasifikator označi negativno od skupa svih poruka koje su negativne. Odnosno:

$$specifičnost = \frac{TN}{TN + FP}$$

Preciznost se odnosi na mjeru klasifikatora da je točno odredio one podatke koje je označio pozitivno. U našem slučaju je to postotak poruka koje klasifikator označi pozitivno od skupa svih poruka koje je označio pozitivno. Odnosno:

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

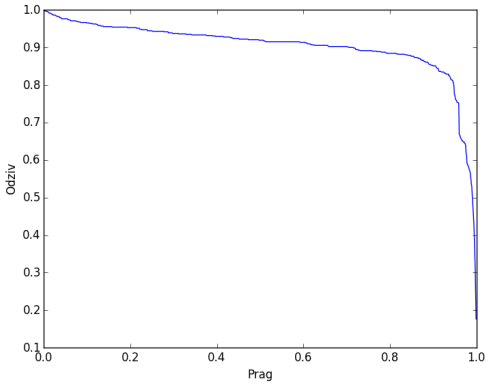
Korisno je za uočiti da krivulja 7.2c raste kako raste prag odluke, a to znači da što više smanjujemo interval za pozitivnu odluku da imamo sve više točno označenih pozitivnih primjera, što je svakako dobar pokazatelj.

Točnost se odnosi na sposobnost klasifikatora da točno odredi oznaku. U našem slučaju je to postotak poruka koje je klasifikator točno označio od skupa svih poruka. Odnosno:

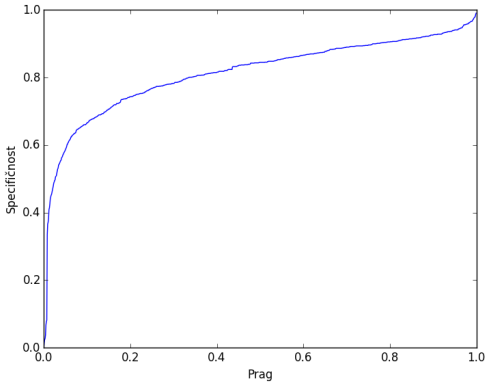
$$točnost = \frac{TP + TN}{TP + FP + TN + FN}$$

Najveća točnost se postiže za prag $T = 0.845$ te iznosi $accuracy = 90.251\%$ (vidi sliku 7.2d).

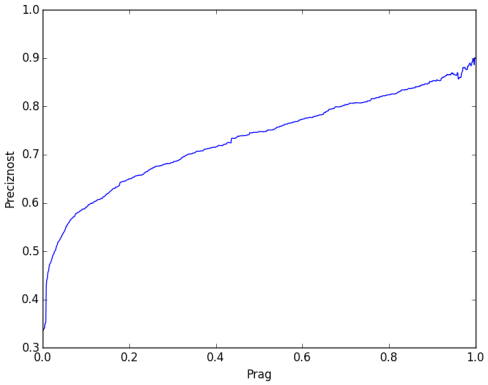
[2]Također poznato kao osjetljivost (engl. *sensitivity*).



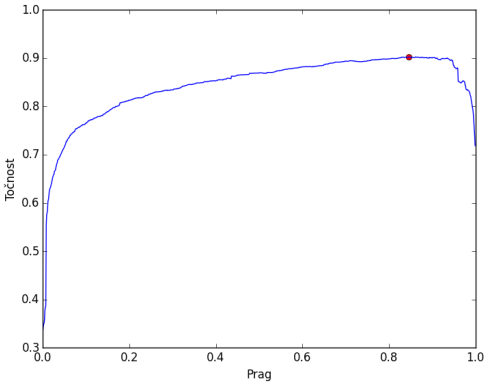
(a) Odziv



(b) Specifičnost



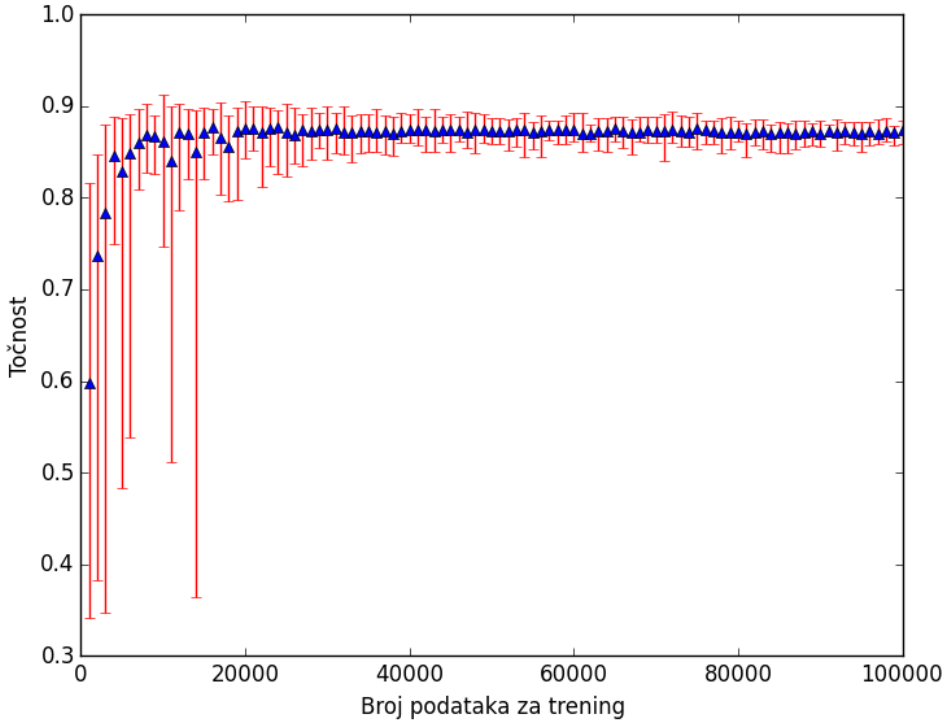
(c) Preciznost



(d) Točnost

Slika 7.2: Promjena mjera klasifikatora s obzirom na prag odluke

Krivulja učenja (engl. *learning curve*) pokazuje kako točnost klasifikacije raste s brojem primjera za učenje. Na slici ispod je prikazana krivulja učenja za izgrađeni klasifikator.



Slika 7.3: Krivulja učenja

Broj primjera se povećava za 1.000 od 0 do 100.000. Za neku veličinu uzorka N uzeto je N slučajno izabranih primjera nad kojima je treniran klasifikator te se nakon toga izračuna točnost nad testnim primjerima. Taj postupak je za svaku veličinu uzorka ponovljen 25 puta. Svaka oznaka na grafu prikazana je s intervalom u kojem su se nalazile točnosti, a oznaka ˆ prikazuje njihovu sredinu. Na grafu se vidi da kad klasifikator dobije više od otprilike 30.000 primjera za učenje, krivulja se može aproksimirati pravcem te točnost postaje više–manje konstantna.

F_1 mjera je mjera točnosti klasifikatora. Njega možemo interpretirati kao težinsku sredinu *preciznosti* i *odziva* klasifikatora. Općenita formula za F_β je

$$\mathrm{F}_\beta = (1 + \beta^2) \cdot \frac{preciznost \cdot odziv}{\beta^2 \cdot preciznost + odziv}$$

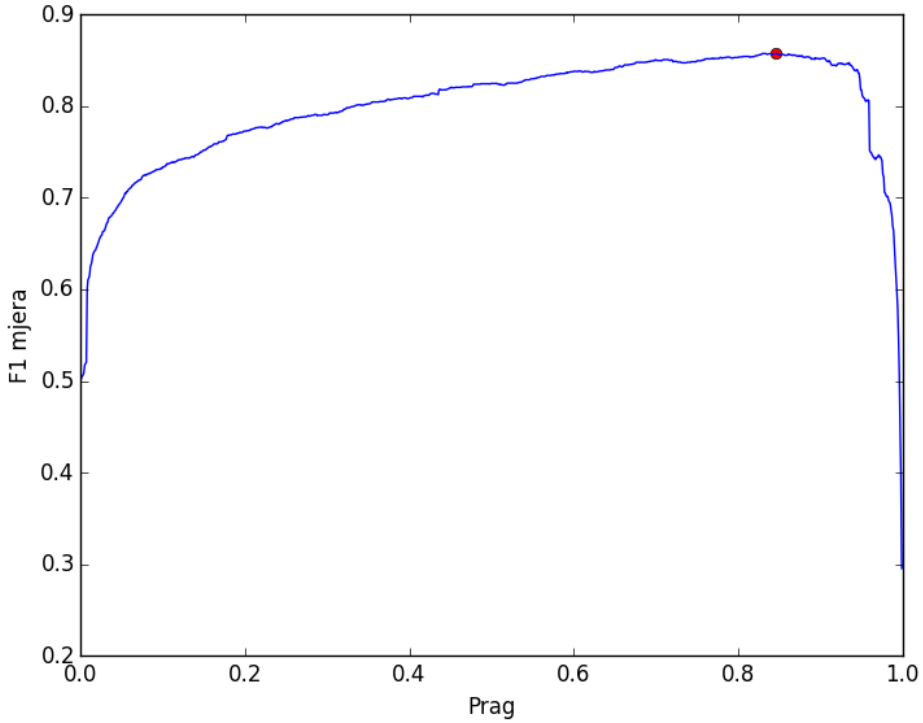
Pa za $\beta = 1$ prelazi u

$$\mathrm{F}_1 = 2 \cdot \frac{preciznost \cdot odziv}{preciznost + odziv}$$

Što se može zapisati i kao

$$\mathrm{F}_1 = 2 \cdot \frac{TP}{2 \cdot TP + FP + FN}$$

Pošto je ta mjera težinska sredina druge dvije mjere, važno je napomenuti da njezinu vrijednost ne možemo tumačiti kao neki postotak podataka. Također, maksimalni F_1 se postiže za prag $T = 0.845$ te iznosi $\mathrm{F}_1 = 0.85753$ što je vrlo visok rezultat.

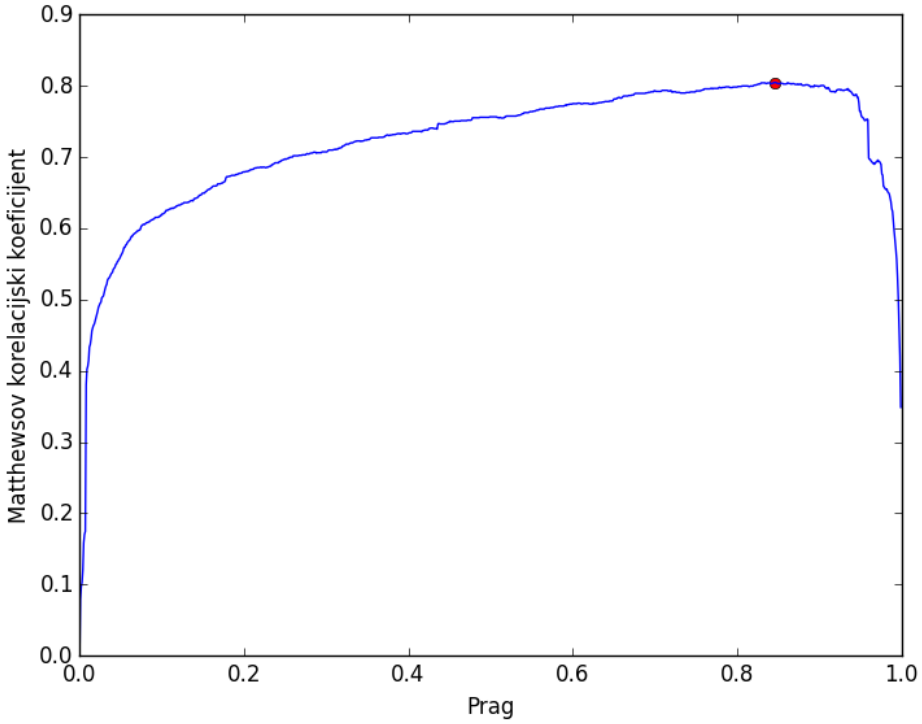


Slika 7.4: F1 rezultat

Matthewsov korelacijski koeficijent se generalno smatra kao uravnotežena mjera koja uzima u obzir točne i netočne pozitivne i negativne primjere te se može koristiti i kad su klase različitih veličina. MCC je u stvari korelacijski koeficijent između stvarnih i određenih oznaka te je u intervalu $[-1, 1]$. Savršeni klasifikator bi imao $MCC = 1$ što bi značilo savršenu korelaciju između stvarnih i određenih oznaka. Možemo ga izraziti kao:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$$

Najveći MCC se postiže za prag $T = 0.845$ te iznosi $MCC = 0.80454$.



Slika 7.5: Matthewsov korelacijski koeficijent

7.2. Testni podaci

Klasifikator je kao optimalni prag izabrao $T = 0.845$ te je isproban na testnim podacima koji su se sastojali od 663 poruke označene s 1 (*true*) i 1337 poruke označene s 0 (*false*). Dobiveni rezultati prikazani su u tablicama 7.2 i 7.3

Tablica 7.2: Oznake klasifikatora

Ispravno pozitivni	569
Pogrešno pozitivni	120
Ispravno negativni	1217
Pogrešno negativni	94

Tablica 7.3: Vrednovanje klasifikatora nad testnim podacima

Odziv	Specifičnost	Preciznost	Točnost	F_1	MCC
85.822%	91.025%	82.583%	89.300%	0.84172	0.78648

8. Zaključak

Tema ovog završnog rada bila je implementacija modela koji može klasificirati poruke mrežnog razgovora u dvije klase: *primjerene* i *neprimjerene*. To je ostvareno pomoću implementacije logističkog klasifikatora te prethodno označenog skupa podataka (poruka). Rezultati su se pokazali vrlo dobri s obzirom na visoke postotke točnog klasificiranja podataka.

Rad je rezultirao uspješnom implementacijom modela za klasificiranje, ali može poslužiti i kao osnova za daljnji rad. Model ima i neke nedostatke kao naprimjer to što kod klasifikacije koristi samo sadržaj poruke, što ponekad nije dovoljno za klasifikaciju same poruke. U budućem radu mogle bi se dodati i druge značajke poruka kao što su kontekst poruke (poruke koje su prethodile i poruke koje su slijedile, odnosno, cijeli razgovor) ili statistika koja bi opisivala kakve poruke taj korisnik uobičajeno objavljuje i slično.

Literatura

[1] Ethem Alpaydin. *Introduction to Machine Learning.* The MIT Press, 2009.

[2] Stanley Lemshow David W. Hosmer. *Applied Logistic Regression.* A Wiley-Interscience Publication, 2000.

[3] GraphLab. GraphLab Create™. `https://dato.com/products/create/open_source.html`. [version 1.8.5].

[4] Alexander Zien Olivier Chapelle, Bernhard Scholkopf. *Semi-Supervised Learning.* The MIT Press, 2006.

[5] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Otkrivanje poruka neprimjerenog sadržaja u mrežnom razgovoru

Sažetak

U današnje vrijeme sve je više ljudi koji komuniciraju putem interneta. Anonimnost korisnika nerijetko dovodi do nepoželjnog ponašanja tijekom razgovora. Da bi se to spriječilo, koriste se razne metode kojima bi se detektirale neprimjerene poruke. Da bi se ubrzao proces označavanja korišteno je aktivno učenje. Implementirana je metoda logističke klasifikacije te se može jednostavno uključiti u postojeći sustav. Dobiveni klasifikator postiže preciznost od 82.583% nad testnim podacima.

Ključne riječi: strojno učenje, obrada prirodnog jezika, logistička regresija, aktivno učenje, nadzirano učenje, klasifikacija, mrežni razgovor

Detection of inappropriate messages in network conversations

Abstract

More and more people nowadays use internet for communication. User anonymity often leads to undesirable behaviour during conversations. In order to put a stop to it, various methods are used to detect such messages. Active learning has been used to speed up data marking. Implementation of logistic regression achieved precision of 82.583% on test data and can easily be incorporated with existing systems.

Keywords: machine learning, natural language processing, logistic regression, active learning, semi supervised learning, classification, online chat