Klasifikacija tekstova temeljena

na skupovima riječi

Sadržaj

[1. UVOD 3](#_Toc143023756)

[2. MODEL ZBIRKI ZNAČAJKI 4](#_Toc143023757)

[3. PROGRAM 5](#_Toc143023758)

[4. METODE KLASIFIKACIJE 7](#_Toc143023759)

[4.1. N-torke 7](#_Toc143023760)

[4.2. Uklanjanje nebitnih riječi 8](#_Toc143023761)

[4.3. Korijenski algoritmi 8](#_Toc143023762)

[4.3.1. Prefiks-sufiks algoritam 8](#_Toc143023763)

[4.3.2. nenazvani korijenski algoritam (koristeći bazu podataka hrvatskih imenica i glagola) 9](#_Toc143023764)

[4.4. „tf-idf“ 10](#_Toc143023765)

[5. KOMBINACIJE METODA 11](#_Toc143023766)

[6. ZAKLJUČAK 12](#_Toc143023767)

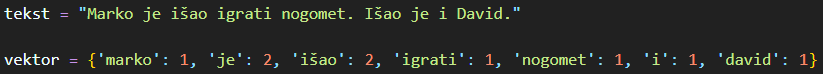
1. UVOD

Model zbirki značajki se koristi u strojnom učenju.

Mmodelu zbirki riječi, gdje su značajke skupovi riječi se najčešće koristi kod klasifikacije teksta Koristi se u Naive Bayes spam filtering...

1. MODEL ZBIRKI ZNAČAJKI

Model zbirki značajki je način izvlačenja značajki iz nekog sadržaja s ciljem sažimanja i oblikovanja tog sadržaja. U kontekstu ovog rada značajke su riječi te njihova frekvencija pojavljivanja, a sadržaj je tekst. Model ne očuvaje redoslijed riječi što bi se moglo zaključiti iz njegovog engleskog naziva: „Bag of words model“ - riječi su „smještene u vreću“, tj. gubi se njihov poredak. Umjesto pojedinačnih riječi može se brojati i učestalost skupa susjednih riječi, takozvanih tokena ili n-torki. Tokeni i njihov broj ponavljanja za svaki pojedinačni tekst spremljeni su u strukturu podataka zvanu rječnik, gdje su ključevi tokeni a vrijednosti broj ponavljanja tog tokena. Kod modela zbirki značajki taj rječnik se zove vektor(primjer vektora slika).



1. PROGRAM

Cilj programa je sa što većom preciznosti klasificirati uneseni tekst, tj. odrediti kojoj od zadanih tema taj tekst najviše odgovara. To radi tako da uspoređuje riječi već poznatih tekstova kojima su teme već određene s riječima teksta kojega unese korisnik. Kroz par distinktivnih metoda klasifikacije pokušano je maksimizirati preciznost.

Program razlikuje šest kategorija tema članka: filmovi/serije, glazba, gospodarstvo, politika, sport i tehnologija. IZABRANE SU TAKO DA SE NE PREKLAPOAJJ DA JE MOGUCE Svaka kategorija ima svoju tekstualnu datoteku koja sadrži 20 poveznica koje su ručno unesene gledajući kako je koji članak kategoriziran na web-stranici časopisa.

~~Program klasificira temu zadanog teksta (ručno unesenog ili dohvaćenog putem poveznice) u šest različitih kategorija~~

Jer je korišten programski jezik Python, ovaj cijeli proces je nešto sporiji kad bi ga usporedili s nekim drugim, ali zato ima puno jednostavniju sintaksu te je lakše rukovati s *string-ovima*.

Program je exportan te se može pokrenuti na Windows operacijskim sustavim. Može se preuzeti s Githuba – link- kako skinut...

* 1. Obrada članaka

Za stvaranje vektora klasifikatora korišteni su internet članci dnevnog časopisa „24sata“ te „Novi list“, U početku je bilo planirano da program može primiti članke iz više časopisa, no bilo je teško ukloniti irelevantne dijelove web-stranice jer svaki časopis ima svoj dizajn *HTML* koda. Zbog tog razloga uzeta su samo dva jer je kod njih odvajanje teksta članka bilo nešto jednostavnije. Jer ti irelevantni dijelovi web-stranice sadrže riječi/rečenice koje nisu povezane s kontekstom samog članka, bilo ih je potrebno ukloniti kako ne bi uništili vektor te kategorije. Primjeri takvih dijelova su izbornici na vrhu s nazivima kategorija, nepovezani predloženi članci i reklame duž i informacije o kompaniji na dnu web-stranice.

Program također priznaje HTML kod Wikipedija čije se stranice također mogu koristiti za vektore klasifikatore, no primarni im je cilj biti za korisnike.

Riješenje nije dugoročno jer u slučaju da se dizajn *HTML* koda članka časopisa „24sata“, „Novi list“ ili „Wikipedija“ promijeni, sav tekst koji se vidi na stranici bez obzira na povezanost s temom članka, će biti unesen u vektor. To vrijedi i ako je unesen članak s nekog drugog časopisa(KRATKA). To posebno dolazi kod izražaja kod metoda koje su usredotočene na tokene čija je učestalost pojavljivanja manja. (slika sa parserom i bez parsera) Na slikama su prikazani sortirani vektor

Poveznice, bile one unesene od strane korisnika ili uzete iz tekstualnih datoteka kategorija, se obrađuju tako da se prvo preuzme „HTML“ kod web-stranice kojoj ta poveznica vodi.

Ako se radi o poveznici sa 24sata, Novog lista ili Wikipedije traži se *<div>* oznaka sa specifičnom *class* vrijednosti koja je ručno pronađena te je drugačija za svaku domenu stranica. Unutar parova te oznake - *<div>* i *</div>* traže se drugačije oznake koje označavaju paragrafe (*<p>*). Sve tri od tih stranica u te oznake stavljaju tekst vezan uz članak, no taj tekst ima razne modifikatore za boldanje, italic...

Uklanaju se oznake *<a>*, *<span>*, *<b>*, *<i>*, *<sup>*, *<strong>*, *<div>*, ali ne i tekst unutar njih. U *HTML* prezentacijskom jeziku postoje različiti kodovi koji zamijenjuju znakove koji bi mogli poremetiti interpretaciju koda. Takvi i njemu slični KODOVI prevedeni se u prave znakove (slika) Rezultat je čitljivi tekst isključivo vezan uz samu temu članka. Takvom tekstu ili ako korisnik za klasifikaciju nije unio poveznicu nego tekst se sva slova pretvaraju u mala slova te su rečenice podjeljene u svoje redove, koristeći interpukcijske znakove kao separatore.

Ako poveznice nije iz jedne od tih stranica(ponvaljanje) Iz tog koda se, osim već spomenutih oznaka, uklanjaju *<style>* i *<script>* *HTML* oznake (eng. tag) (i svi znakovi izmedu njih) te pronađeni zakomentirani kod, te je preostali proces isti. OSIM ŠTO OSTAJU NEBITNE RIJECI ALI TO SAM VEC REKAO GORE

To se dalje obrađuje ovisno o metodi klasifikacije koja se koristi. Iz tih metoda nastaju vektori te ako se radi o vektorima vokabulara iste kategorije oni se povezuju unijom(primjer slika).

Usporedba vektora i izračun preciznosti

Načini uspoređivanja vektora klasifikatora i vektora koji se klasificira, tj. način određivanja preciznosti je .... „ifidf“(poveznica prema poglavlju) ima svoj način jer je ...

Nakon izračuna preciznosti klasifikacije, automatski se izradi tekstualna datoteka(slika tekstualne datoteke) koja sadrži koja su preklapanja i koliko ih je – za svaku kategoriju. Pregledavanjem sadržaja datoteke može se zaključiti zašto program odluči da je jedna kategorija zastupljenija od druge, te će biti korištena za objašnjavanje dobivenih rezultata kroz rad.

1. METODE KLASIFIKACIJE

U nastavku su navedene metode s kojima je pokušano povećati preciznost klasifikacije. Za uspredbu preciznosti metoda klasifikacije svakom metodom je klasificiran po jedan hrvatski Wikipedija članak iz svake od ukupno šest kategorija te su rezultati prikazani u tablicama. Za kategoriju sport je izabran članak Luke Modrića, za gospodarstvo „Turizam u Hrvatskoj“, za politiku članak Zorana Milanovića...(poveznice za te članke)

* 1. N-torke

N-torka je konačan niz n objekata, a ovdje ona predstavlja niz susjednih riječi s ciljem očuvanja dodatnih informacija iz članka. Za n-torke pretpostavka je da će što je veći *n* biti manja preciznost zbog jako malog broja matches. Najjednostavnija i potencijalno najgora metoda je 1-torka, u prijevodu samo jedna riječ. Jer preostale metode u osnovi koriste jednu riječ, pravilno je reći da im prethodi se koriste n-torkom. ~~Uklonjene su sve „riječi“ koje se ne sastoje isključivo od znakova abecede.~~

~~1-torka je u prijevodu samo jedna riječ i predstavlja najjednostavniju i potencijalno najgoru? od metoda klasifikacije.~~

(objašnjenje tablice)

(tablica sa rezultatima)

2-torka, dvojka

(objašnjenje tablice)

(tablica sa rezultatima)

3-torka, trojka

(objašnjenje tablice)

(tablica sa rezultatima)

4-torka, četvorka?

(objašnjenje tablice)

(tablica sa rezultatima)

Preostale metode se oslanjaju na 1-torku, a n-torke se opet pojavljuju kod kombinacija metoda(poveznica na poglavlje?)

* 1. Uklanjanje nebitnih riječi

Pod nebitnim riječima smatraju se riječi iz kojih ne bih mogli razaznati o čemu se piše u nekom tekstu, tj. koja je njegova tema. Za uklanjanje takvih riječi napravljena je tekstualna datoteku koja sadrži što više zamjenica, brojeva, priloga, prijedloga, veznika, čestica i usklika. Tekstualna datoteka je ručno pregledana te su izostavljene riječi koje bi ipak mogle imati kontekstualnu važnost. Jedan primjer je riječ „oko“ koja je i prijedlog ali i imenica. Pretpostavka je da ova metode neće imali preveliki utjecaj na preciznost klasifikacije jer će dva dovoljno duga teksta različitih tema vjerojatno imati slične „nebitne riječi“, koje bi se u vektorima „poništile“.

(objašnjenje tablice)

(tablica sa rezultatima)

* 1. Automatsko određivanje korijena riječi

Ova metoda pokušava algoritmima povezati riječi koje nemaju isti oblik, jer su u drugom padežu ili glagolskom vremenu, ali imaju isto značenje. Mnogi od pronađenih algoritama su fokusirani na engleski jezik te nisam našao dovoljno materijala da bi ih mogao prilagoditi hrvatskome jeziku.

* + 1. Prefiks-sufiks algoritam

Uklanjanje sufiksa je puno jednostavnije u engleskom jeziku zbog njihovog malenog broja, uvelike zbog nedostatka padeža te načina oblikovanja glagolskih vremena(ok?, usporedi s hrvatskim i ostalim jezicima). Sufiksi imenica, pridjeva i glagola skupljeni su u jednu tekstualnu datoteku te ako su pronađeni na kraju bilo koje riječi, uklonjeni su. Na isti način, lista prefiksa pronađena na Wikipediji(izvor), prebačena je u svoju tekstualnu datoteku. Prefiksi i sufiksi su sortirani po duljini kako bi se osiguralo da se prvo ukloni najduži mogući dio riječi. (primjer?) uklonjeni su koristeci ove python metode

(objašnjenje tablice)

(tablica sa rezultatima)

Kod sufiks-prefiks algoritma neophodno je prvo uklanjanje nebitnih riječi zbog njihove kratke duljine. Tokeni stvoreni iz veznika pa, te, ni, ali... postaju potpuno beznačajni kad im se uklone sufiksi a, e te i – česti sufiksi deklinacije većine imenica. Iako uklanjanje nebitnih riječi pomaže u preciznosti klasifikacije, ne rješava u potpunosti problem preagresivnog uklanjanja sufiksa.

(objašnjenje tablice)

(tablica sa rezultatima)

Sufiks „ama“ se dodaje na korijen imenica ženskog roda u dativu, lokativu i instrumentalu množine (e-sklonidba) te ga napravljena tekstualna datoteka sufiksa sadrži. Problem stvaraju riječi koje završavaju na „ama“ ali nisu u navedenim padežima. „Mama“, „jama“, „slama“, „tama“ i „dama“ su riječi u nominativu jednine te uklanjanjem sufiksa „ama“ dobivamo jednoslovne tokene koji su potpuno neupotrebljivi. Ovo je samo jedan od mnogih primjera preagresivnog uklanjanja sufiksa. Način na koji bi se ovaj problem mogao rješiti je da se uklanjanje sufiksa dopusti jedino ako je duljina sufiksa (broj slova sufiksa) manja ili jednaka polovici duljine cijele riječi (broja slova cijele riječi). Dodatno, „korijenovanje“ bi se mogao zabraniti ako je token koji nastane prekratak – u sljedećoj tablici(poveznica do tablice) prikazani su rezultati klasifikacije token je prekratak ako je duljine tri slova ili manje. U sljedećoj tablici(pointer) prikazani su rezultati klasifikacije

(objašnjenje tablice)

(tablica sa rezultatima)

* + 1. nenazvani korijenski algoritam (koristeći bazu podataka hrvatskih imenica i glagola)

BAZE BLA BLA

Umjesto da se ručno pokušavaju ukloniti prefiksi i sufiksi, lakše bi bilo naći već postojeću bazu podataka koja sadrži sve moguće oblike svih riječi te njihov osnovni oblik.

„croDict“ (poveznica) je web-stranica koja služi kao prevoditelj riječi s engleskog i njemačkog na hrvatski. Osim toga sadrži i popise imenica te glagola na tim jezicima, uključujući i njihove deklinacije, tj. konjugacije gdje je to primjenjivo. U polje za pretraživanje se upiše tražena riječ i izabere se je li ta riječ imenica ili glagol. Jer želimo da osnovni oblik dohvati program, ne možemo se korititi elementima sučelja web-stranice. Srećom, osim polja za pretraživanje i opcije za vrstu riječi, moguće je stvari definirati u samoj poveznici.(slika poveznice stranice). Jer program ne zna odredriti vrstu riječi potrebno je isprobati obje opcije. U slučaju da traženu riječ ne prepoznaje, na vrhu web-stranice se ispiše „Der gesuchte Begriff konnte nicht gefunden werden“ što je prevedno s njemačkog „Traženi pojam nije pronađen“. Nakon slanja zahtjeva i nakon primitka HTML-a, ako program nije pronašao taj error izraz, pokušava dohvatiti riječ iznad tablica koja predstavlja osnovni oblik unesene riječi – nominativ jednine za imenice te infinitiv za glagole.(slika sto stanica pokazuje ako je successful)

(To je bila originalna ideja, no IGNORE MAYBE) kako bi se olakšao load cro dict servera, umanjila šansa zabrane pristupa stranici zbog prevelikog broja zahtjeva te ubrzala klasifikacija odlučeno je da će se rezultat te upita stranici spremiti lokalno u json datoteku, gdje je ključ riječ koja je unesena a vrijednost je osnovni oblike te riječi. Sada, ako je ta riječ već bila tražena, neće se ponovno slati zahtjev nego će se uzeti iz rječnika. Za još veće ubrzanje programa i olakšanje zahtjeva (možda koliko puta ovdje 7 puta?? ne jer imaju iste nastavke neki padezi) , napravljeno je da program ne sprema osnovni oblik riječi samo za traženu riječ, nego da spremi sve moguće oblike riječi te da ih sve poveže.

(Parsiranje je bilo nešto teže jer...??)

Rezultat je metoda koja može spojiti sve oblike velikog broja imenica i glagola. (showcase kako spaja rijeci nepravilne deklinacije imenica ronilac i vrabac i jos neki ako nadem)

Ako za zadanu riječ web-stranica nije mogla pronaći valjanu riječ, ta zadana riječ je lokalno spremljena u tekstualnu datoteku, kako bi se spriječilo ponovno slanje zahtjeva stranici. Te riječi su ili ignorirane ili ne u sljedećim tablicama.

Rezultati za metodu s neprodnadenim i bez nepronadenih riječi

(objašnjenje tablice)

(tablica sa rezultatima)

(objašnjenje tablice)

(tablica sa rezultatima)

Pronadeno i na githubu ali jako lose formatirano

„Wječnik“ (poveznica) je internet rječnik zasnovan na dobrovoljnim dodacima i izmjenama korisnika te je „sestrinski projekt Wikipedije“(citat). Sadrži više od deset tisuća riječi (citat) – većinom imenice. Velik broj web-stranica tih riječi uopće ne sadrži njihovu deklinaciju te uspoređujući to s toliko (izvor) riječi koju hrvatski jezik zapravo ima. Iako taj rječnik sadrži samo frakciju pravog broja rječi ne bi ga bilo loše implemenirati. Problem je što...

* 1. „tf-idf“

što je...

Jer metoda „tf-idf“ prebacuje važnost s tokena koji se ponavljaju puno na tokene koji se ponavljaju malo nebitne riječi koje se nalaze u headeru i footeru će poremetiti preciznost...

(objašnjenje tablice)

(tablica sa rezultatima)

1. KOMBINACIJE METODA

(najbolje kombinacije te njihove tablice s rezultatima)

1. ZAKLJUČAK