Sustav za davanje kontekstualiziranih preporuka na temelju rudarenja teksta

Kamenjaš, Lena

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Lena Kamenjaš

SUSTAV ZA DAVANJE KONTEKSTUALIZIRANIH PREPORUKA NA TEMELJU RUDARENJA TEKSTA

Diplomski rad

Voditelj rada:
prof. dr. sc. Luka Grubišić

Zagreb, studeni 2018.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

Većina ljudskog znanja pohranjena je u nestrukturiranom, tekstualnom obliku. Obzirom na neprestan rast digitalnih podataka (procjenjuje se da se količina pohranjenih podataka udvostručuje svakih 20 mjeseci [1]) javlja se potreba za stvaranjem sustava za analizu i pretraživanje teksta. Takvi sustavi omogućavaju crpljenje informacija iz velikih količina nestrukturiranih podataka. Ovisno o potrebama, sustav može klasificirati dokumente, pretraživati ih, generirati ključne riječi ili sažetak za određeni dokument.

Razlika između rudarenja teksta i rudarenja ostalih tipova podataka je što su u većini ostalih tipova informacije koje se traže implicitne, tj. skrivene ljudskom oku, ili čak nepoznate te je glavni zadatak otkriti ih. Kod rudarenja teksta, informacija je jasno vidljiva i raspoznatljiva - ljudima. No, ručna obrada svakog dokumenta iziskuje previše vremena te dolazi do potrebe za automatiziranom obradom.

Tema ovog rada je izrada mrežne aplikacije s poboljšanim pretraživanjem hrvatskih zakona, na način da se za svaki zakon odrede njegove ključne riječi te po njima pretražuje. Prikupljeni dokumenti su zakoni, javno dostupni, objavljeni u Narodnim novinama[2] tijekom 2015. i 2016. godine.

Za bilo koji proces računalne obrade teksta je tekst potrebno strukturirati na neki način prihvatljiviji računalu, čime se bavi disciplina obrade prirodnog jezika[3] i o čemu se priča u prvom poglavlju. Zatim se općenito priča o otkrivanju ključnih riječi i o dva algoritma za otkrivanje ključnih riječi. Zatim se ukratko navode i objašnjavaju tehnologije i alati korišteni za izradu aplikacije. Naposlijetku, objašnjava se proces implementacije.

[1] Preuzeto iz [6]

[2] Narodne novine, dioničko društvo za izdavanje i tiskanje službenoga lista Republike Hrvatske, službenih i drugih tiskanica te za trgovanje školskim i uredskim priborom

[3] engl. *Natural Language Processing - NLP*

Poglavlje 1

Obrada prirodnog jezika

Obrada prirodnog jezika je disciplina koja pruža računalnu podršku kad je početni ulaz[1] problema tekst pisan prirodnim jezikom. Područja u kojima dolazi do potrebe za obradom prirodnog jezika su npr. odgovaranje na pitanja, ekstrakcija informacija, sentimentalna analiza, strojno prevođenje, označavanje dijelova teksta[2], prepoznavanje imenovanih entiteta[3].

Na početku razvijanja ove discipline, sustavi za obradu jezika su se formirali tako da bi se ručno napisao skup pravila (npr. pisanje gramatika ili heurističkih pravila), no to se nije pokazalo dovoljno robusnim rješenjem, obzirom na višeznačnost jezika. Pristupanjem ovom problemu kroz strojno učenje (oko 1990. godine) postigao se značajan napredak na područjima sentimentalne analize, određivanja smisla riječi, parsiranja teksta, strojnog prevođenja i ekstrakcije informacija, no još uvijek postoje područja gdje je višeznačnost jezika teško nadvladati: odgovaranje na pitanja, ekstrakcija sažetka, parafraziranje, dijalog.

Svaki zadatak obrade prirodnog jezika uključuje sljedeće dvije komponente:

1. segmentiranje[4] riječi u tekstu - *token* je niz znakova (tekstualnih, numeričkih, punktuacijskih) koji prema pravilima danog jezika definira smislenu cjelinu, primjerice *vrata* (imenica), *20.* (redni broj), . (punktuacija). Tekst se promatra kao niz *tokena* i cilj ga je raščlaniti na sve *tokene* koje sadrži.

2. normalizacija riječi - nakon segmentacije riječi, riječi je potrebno normalizirati. Normalizacija može biti npr. na razini oblika riječi, fonetička normalizacija, ispravljanje pravopisnih grešaka.

[1] engl. *input*
[2] engl. *part of speech tagging*
[3] engl. *named entity recognition*
[4] engl. *tokenization*

Tijekom godina razvijaju se razni alati za obradu teksta, od kojih su neki javno dostupni, npr. *General architecture for text engineering* - GATE[5] i *The Stanford Natural Language Processing Group*[6]. Obzirom da obrada teksta pisanog prirodnim jezikom uvelike ovisi o jeziku na kojem je tekst pisan, nije moguće koristiti bilo koji alat za bilo koji jezik. Oba prethodna alata su izvrsna za obradu tekstova na engleskom jeziku, no nijedan se ne može primijeniti na tekst pisan na hrvatskom jeziku zbog različitosti između engleskog i hrvatskog jezika. Primjerice, već kod segmentacije riječi za datume, redne ili rimske brojeve oba će alata pogrešno segmentirati riječi, jer u engleskom jeziku niti formati datuma niti redni brojevi ne uključuju točku. Tako bi se izraz ”20. siječnja” netočno segmentirao u tri *tokena*: ”20”, ”.”, ”siječnja”, a trebao bi se segmentirati u dva: ”20.”, ”siječnja”.

1.1 Projekt ReLDI

ReLDI (engl. *Regional Linguistic Data Initiative*) je međunarodni institucionalni projekt, financiran u okviru programa SCOPES (engl. *Scientific Co-operation between Eastern Europe and Switzerland*) švicarske nacionalne zaklade za znanost. Cilj projekta je razviti resurse i alate za obradu južnoslavenskih jezika: slovenski, srpski i hrvatski. Razvijeni alati[7] su:

- obnova dijakritičkih znakova
- označavanje dijelova teksta - označavanje se radi prema specifikaciji zadanoj na `http://nl.ijs.si/ME/V5/msd/html/msd-hr.html`. Radi se o jednom obliku normalizacije riječi, točnije normalizacije oblika riječi zvanog lematizacija - postupak u kojem je potrebno odrediti lemu[8] svakog tokena. Ovim postupkom se npr. riječi *je*, *su*, *bilo* sve svode na jednu - *biti*. Više detalja se može pronaći u [2].
- parsiranje ovisnosti (analiza rečenice)
- prepoznavanje imenovanih entiteta

Navedeni alati mogu se upotrijebiti na mrežnoj aplikaciji `http://clarin.si/services/web/` ili korištenjem mrežnog servisa preko Python biblioteke `https://pypi.org/project/reldi/`.

[5] dostupno na `https://gate.ac.uk/`

[6] dostupno na `https://nlp.stanford.edu/`

[7] preuzeto s `https://github.com/clarinsi/reldi-lib`

[8] lema - kanonski (leksikografski) oblik promjenljivih vrsta riječi (infinitiv za glagole, muški rod jednine za pridjeve, itd.), riječ svedena na osnovni oblik

Poglavlje 2

Otkrivanje ključnih riječi

Otkrivanje ključnih riječi se može opisati kao proces čiji je cilj identificirati izraze (riječi, kombinacija riječi) koji najbolje opisuju temu zadanog dokumenta. Predstavlja važan problem u područjima rudarenja teksta, dohvata informacija [1] i obrade prirodnog teksta. Neke od primjena su kod pretraživača, indeksiranja baza podataka znanstvenih radova, uspoređivanje sličnosti dokumenata i klasifikacija dokumenata.

Metode otkrivanja ključnih riječi se mogu ugrubo podijeliti na ***izvlačenje*** *ključne riječi*[2] i na ***dodjelu*** *ključne riječi*[3]. Metode mogu biti:

- nadzirane - podaci su u obliku $(ulaz, izlaz) = (x, y)$, treba naći preslikavanje $\tilde{y} = f(x)$
- nenadzirane - podaci su neoznačeni, ciljna vrijednost nije unaprijed poznata, treba pronaći pravilnost u podacima
- polunadzirane - većina je podataka neoznačena

2.1 Izvlačenje ključne riječi

Glavna karakteristika ovih metoda je da su sve ključne riječi izvučene iz teksta koji se analizirao. Načini na koji se evaluira vrijednost svake riječi variraju; može se, primjerice, gledati frekvencija riječi (broj svih pojava zadane riječi u tekstu), lokacija riječi u odnosu na početak/kraj dokumenta, lokacija riječi u odnosu na početak/kraj paragrafa. Problemu se također može pristupiti preko grafova, gdje su izrazi (riječ, kombinacija riječi) vrhovi, a bridovi relacija koja izražava povezanost vrhova.

[1] engl. *Information Retrieval - IR*
[2] *engl. keyword extraction*
[3] engl. *keyword assignment*

Tipičan algoritam za izvlačenje ključnih riječi dan je sljedećim pseudokodom:

Algoritam 1: Izvlačenje ključne riječi

```
ulaz : n: broj ključnih riječi, d: dokument
izlaz: ključne riječi
kandidati ← nadjiKandidateZaDokument (d);
za kandidat ∈ kandidati radi
    svojstva ← nadjiSvojstvaKandidata (kandidat);
    evaluirajVrijednostKandidata (kandidat, svojstva);
sortirajSilaznoPoVrijednosti (kandidati);
kljucneRijeci = kandidati [1 : n];
```

Parametar n je uglavnom unaprijed određen, no postoje i implementacije gdje se dinamički određuje, ovisno o duljini teksta, npr. u [3].

Nalaženje kandidata za dokument je zapravo obrada prirodnog jezika, o čemu se govorilo u prethodnom poglavlju.

Iz pseudokoda se vidi da je algoritam dosta fleksibilan - mogu se koristiti nadzirani, polunadzirani i nenadzirani pristupi, ovisno o problemu, no lagano se mogu integrirati i drugi algoritmi koji se koriste za pretraživanje, kao što su [4] i [1].

Kao glavni nedostatak kod metoda izvlačenja ključnih riječi navodi se to što je ključna riječ *samo* izvađena iz teksta - rječnik ključnih riječi se tako svodi isključivo na rječnik tog dokumenta, što dovodi do toga da bi unutar skupa dokumenata pronađene ključne riječi bile nestandardizirane, što bi moglo dovesti do nekonzistentnosti.

2.2 Dodjeljivanje ključne riječi

Razlika između izvlačenja i dodjeljivanja ključne riječi je *odakle* riječ dolazi - kod dodjeljivanja, ključna riječ se bira iz unaprijed zadanog rječnika ili rječnika sinonima. Na taj način se za ključnu riječ može pojaviti i pojam koji ne postoji u dokumentu, ali je na neki način povezan s njim, primjerice preko sinonimije ili antonimije. Iako u nekim situacijama postojanje rječnika dolazi kao poboljšanje, ukoliko oni nisu dobro definirani, u smislu da ne pokrivaju dovoljno dobro teme svih dokumenata, kvalitetni kandidati za ključne riječi mogu biti odbačeni, čime se smanjuje kvaliteta odabranih ključnih riječi.

Tipičan algoritam za dodavanje ključnih riječi bi se mogao gledati kao nadopunjenje algoritma za izvlačenje - u pseudokodu 1 bi se na kraj dodala još jedna linija:

```
filtrirajPronadjeneKljucneRijeciPoRjecniku(kljucneRijeci);
```

Nakon pronađenih ključnih riječi iz teksta, prihvatile bi se samo one koje nađu poveznicu u rječniku - npr. postoji identična riječ, postoji sinonim/antonim.

Dodjeljivanje ključnih riječi se najčešće koristi s nadziranim pristupom - kod kategorizacije gdje su kategorije unaprijed poznate ili kod skupa dokumenata gdje postoji skup za treniranje sa već označenim ključnim riječima. U oba slučaja se može formirati rječnik koji dobro odgovara traženom cilju.

Poglavlje 3

Algoritmi za izvlačenje ključne riječi

U ovom poglavlju se predstavljaju dva algoritma za izvlačenje ključnih riječi. U prvoj sekciji se detaljnije priča o TextRank algoritmu, a u drugoj o tf-idf algoritmu.

3.1 TextRank algoritam

TextRank algoritam je grafovski algoritam rangiranja, nastao po uzoru na PageRank algoritam [4]. Osnovna je ideja algoritma svaki tekstualni dokument pretvoriti u graf, koji daje cjelovitiju informaciju o dokumentu nego kad bi se gledala svaka riječ zasebno.

Neovisno o specifičnostima problema, primjena grafovskih algoritama rangiranja na dokumente pisane prirodnim jezikom se sastoji od sljedećih koraka:

1. identificiraj tekstualne jedinice, ovisno o problemu, dodaj ih kao vrhove u graf
2. identificiraj veze između vrhova u grafu i po tim vezama konstruiraj bridove u grafu, koji ovisno o problemu mogu biti usmjereni/neusmjereni te imati težinu
3. iteriraj algoritam do konvergencije
4. sortiraj vrhove po njihovoj vrijednosti i izvrši selekciju prikladnih vrhova

3.1.1 TextRank model

TextRank model zasniva se na modelu dokumenta opisanog grafom, gdje su vrhovi tekstualne jedinice (riječi, izrazi) i gdje je cilj odrediti koliko je koji vrh važan u grafu. Ta se informacija za svaki vrh računa rekurzivno gledajući cijeli graf.

Osnovna je ideja preuzeta iz [4] i zasniva se na pojmu glasanja gdje vrhovi međusobno glasaju jedan za drugog. Važnost, odnosno vrijednost vrha, je proporcionalna s brojem

glasova koje ima. Također, ne nose svi glasovi istu težinu - što je vrijednost vrha koji daje glas veća, veća je i težina njegovog glasa.

Predstavljaju se sljedeće definicije:

Definicija 3.1.1. *Graf je uređen par (V, E) pri čemu je V skup vrhova, a E skup bridova. Skup bridova E je podskup svih dvočlanih podskupova od V. Za vrhove $A, B \in V$ kažemo da su susjedni ako je $\{A, B\} \in E$.*
Vrh A i brid e su incidentni ako je $A \in e$, tj. ako postoji e takav da $e = \{A, B\}$ za neki B.

Definicija 3.1.2. *Usmjereni graf je graf $G = (V, E)$ gdje E ima dodatno svojstvo*

$$(v, w) \in E \implies (w, v) \notin A.$$

Definicija 3.1.3. *Težinski graf je par (G, ω) gdje je $G = (V, E)$ graf, a $\omega : E \to \mathbb{R}_0^+$ neka funkcija koju nazivamo težinska funkcija.*

$$\textit{Broj } \sum_{e \in E} w(e) \textit{ nazivamo } \textbf{\textit{težinom grafa}}.$$

Definicija 3.1.4. *Podgraf grafa $G = (V, E)$ je graf čiji su vrhovi elementi skupa V, a bridovi elementi skupa E.*

Definicija 3.1.5. *Za dva disjunktna grafa $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ se definira njihova unija $G_1 \cup G_2$ kao graf $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$.*

Definicija 3.1.6. *Graf je **povezan** ako se ne može prikazati kao unija neka dva grafa. U suprotnom, graf je **nepovezan**. Svaki se nepovezani graf može prikazati kao unija povezanih grafova. Svaki član te unije se zove **komponenta povezanosti**.*

Formalno, neka je $G = (V, E)$ usmjeren graf. Za zadan vrh $V_i \in V$, neka je $In(V_i)$ skup vrhova takav da:

$$In(V_i) = \{V_j \mid \exists j \ (V_j, V_i) \in E\} \tag{3.1}$$

Elemente skupa $In(V_i)$ nazivamo prethodnicima vrha V_i. Analogno, neka je $Out(V_i)$ skup vrhova takav da:

$$Out(V_i) = \{V_j \mid \exists j \ (V_i, V_j) \in E\} \tag{3.2}$$

Elemente skupa $Out(V_i)$ nazivamo sljedbenicima vrha V_i.
Ako se sa $S : V \to \mathbb{R}$ označi vrijednost vrha V_i, prema [4] vrijednost vrha V_i definirana je s:

$$S(V_i) = (1 - d) + d \cdot \sum_{V_j \in In(V_i)} \frac{1}{|\, Out(V_j)\,|} S(V_j) \tag{3.3}$$

d je takozvani faktor prigušenja[1] i poprima vrijednosti iz intervala $\langle 0, 1\rangle$. Uloga faktora d je integrirati u model vjerojatnost skakanja s danog vrha u grafu na drugi vrh, slučajno izabran. U kontekstu pretraživanja web stranica u [4], faktor d realizira model slučajnog izabiranja[2], gdje korisnik odabire poveznice na slučajan način s vjerojatnošću d. U kontekstu pretraživanja po dokumentu, faktor d se može protumačiti kao korisnikovo "čitanje" dokumenta - pročita se samo jedna riječ na određenoj lokaciji, a ta se lokacija odabire na slučajan način s vjerojatnošću d. Faktor d se uobičajeno postavlja na 0.85.

Iz formule (3.3) se vidi da je namijenjena usmjerenim bestežinskim grafovima - usmjerenim jer postojanje poveznice s jedne stranice na drugu ne znači da postoji poveznica u obratnom smjeru, a bestežinskim jer bi bilo neobično za stranicu da višestruko daje poveznicu na drugu stranicu. No, može se prilagoditi za oba slučaja.

Ukoliko se primjenjuje na neusmjerene grafove, skupovi $In(V_i)$ i $Out(V_i)$ definirani s (3.1) i (3.2) su jednaki i uz oznaku $Nghbs(V_i) = In(V_i) = Out(V_i)$ formula izgleda:

$$S(V_i) = (1-d) + d \cdot \sum_{V_j \in Nghbs(V_i)} \frac{1}{|\, Nghbs(V_j)\,|}\, S(V_j). \tag{3.4}$$

Ukoliko se primjenjuje na težinske grafove, za brid od vrha V_i do vrha V_j se definira w_{ij} te se definira težinska vrijednost vrha V_i $WS : V \to \mathbb{R}$ na način:

$$WS(V_i) = (1-d) + d \cdot \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{v_k \in Out(V_j)} w_{jk}}\, WS(V_j). \tag{3.5}$$

Za primjenu prethodno opisanih formula, potrebno je konstruirati graf od dokumenta. Zadani je tekst potrebno obraditi na način opisan u poglavlju 1 - segmentirati riječi, lematizirati ih i označiti ih.

Ovisno o zadanoj problematici, vrhovi grafa mogu biti riječi, fraze pa čak i cijele rečenice. Također, povezanost vrhova također određuje problematika - vrhovi se mogu povezati na osnovu leksičke ili semantičke veze, na osnovu značenja itd.

3.1.2 Izračunavanje TextRank algoritma

Obzirom da je formula (3.3) rekurzivna, postavlja se pitanje kako izračunati vrijednosti za vrhove u grafu. Neka je funkcija $TextRank\colon V \times \mathbb{N} \to \mathbb{R}$ zadana na sljedeći način:

$$\begin{gathered} t = 0, \text{ odnosno inicijalni korak, postavlja se } TextRank(V_i, 0) = 1, \\ t > 0\colon TextRank(V_i, t) = (1-d) + d \cdot \sum_{V_j \in In(V_i)} \frac{1}{|\, Out(V_j)\,|}\, TextRank(V_j, t-1). \end{gathered} \tag{3.6}$$

[1] engl. *damping factor*

[2] engl. *random surfer model*

Inicijalno se svim vrhovima postavlja vrijednost 1, a u svakom narednom koraku se računaju vrijednosti na osnovu prethodne iteracije. Ako se vrhovi grafa numeriraju $V = (V_1, \ldots, V_N)$, gdje je N broj svih vrhova, prethodna formulacija se može izraziti matrično.

Neka je $\mathbf{S} \in \mathbb{R}^{N \times N}$ matrica susjedstva za graf $G = (V, E)$, tj.

$$\mathbf{S}_{ij} = \begin{cases} 1 & \text{ako } (V_i, V_j) \in E \\ 0 & \text{inače} \end{cases}.$$

Nadalje, neka je matrica $\mathbf{A} \in \mathbb{R}^{N \times N}$ definirana na sljedeći način:

$$\mathbf{A}_{ij} = \begin{cases} (1-d) + d \cdot \frac{\mathbf{S}_{ij}}{|\,Out(V_j)\,|} & \text{ako } |\,Out(V_j)\,| \neq 0 \\ 1 & \text{inače} \end{cases}.$$

Tada rješenje jednadžbe

$$\mathbf{x} = \mathbf{A}\mathbf{x}, \text{ za } \mathbf{x} \in \mathbb{R}^N \setminus \{\mathbf{0}\} \tag{3.7}$$

predstavlja izračunate vrijednosti TextRank algoritma za pojedini vrh ($x_i = S(V_i)$, za $i \in \{1, \ldots, N\}$). Na ovako formuliran problem se može primijeniti metoda potencija[3], analogno kao u [4]. Prema Perron-Frobeniusovom teoremu, metoda potencija će sigurno konvergirati (postojat će rješenje $\mathbf{x} \in \mathbb{R}^N \setminus \{\mathbf{0}\}$ koje ima samo nenegativne vrijednosti) ako je matrica $\mathbf{A}$ ireducibilna i ima nenegativne vrijednosti[4].

Za inicijalno rješenje se uzima $\mathbf{x}^{(0)} = (1, \ldots, 1)$. Pri svakoj iteraciji se za svaki vrh računa njegova greška. Greška vrijednosti vrha je definirana kao razlika između stvarne vrijednosti vrha i vrijednosti izračunate u trenutnoj iteraciji. Obzirom da su podaci neoznačeni, stvarna vrijednost vrha nije unaprijed poznata te se vrši aproksimacija greške, na način da se za vrh računa razlika vrijednosti u dvije uzastopne iteracije.

Oznaka $\mathbf{x}_i^{(k)}$ predstavlja vrijednost $S(V_i)$ u k-toj iteraciji, za $i \in \{1, \ldots, N\}$ i $k \in \mathbb{N}$. Iteracije algoritma se zaustavljaju kad se dosegne konvergencija, koja se postiže kad greška bilo kojeg vrha padne ispod zadane tolerancije ϵ, odnosno kad se zadovolji sljedeći kriterij:

$$|\,\mathbf{x}_i^{(k+1)} - \mathbf{x}_i^{(k)}\,| < \epsilon, \text{ za } \epsilon > 0, i \in \{1, \ldots, N\}, k \in \mathbb{N}. \tag{3.8}$$

3.1.3 Izvlačenje ključnih riječi

TextRank algoritam za izvlačenje ključnih riječi je nenadziran. Kandidati za vrhove su riječi (točnije njihove leme), a relacija za konstrukciju bridova je relacija zajedničkog

[3] više se može pronaći na `https://www.fsb.unizg.hr/mat-4/PA2/eig_val.pdf`

[4] preuzeto s `www.inm.ras.ru/vtm/texts/difiore-lecture-toe_1o.pdf`

pojavljivanja[5] - dva su vrha spojena ako se (neki njihovi oblici) u tekstu pojavljuju s međusobnom udaljenosti od maksimalno N riječi, gdje je N broj između 2 i 10.

Prije nego što riječ bude dodana u graf kao vrh, može se postaviti dodatni sintaktički filter, koji odobrava riječi s posebnim oznakama. Primjerice, mogu se dopuštati samo imenice, samo glagoli, sve klase riječi ili kombinacija klasa. U ovom pristupu dopuštaju se samo opće imenice i pridjevi.

Dobiveni konstruirani graf je neusmjereni i bestežinski, na kojeg se primjenjuje formula (3.6) do konvergencije.

3.1.3.1 Primjer primjene TextRank algoritma za izvlačenje ključnih riječi

Neka je zadan (kratki) dokument sljedećeg sadržaja:

> Dozvola iz točke II. ove Odluke izdaje se na rok od najdulje 30 godina i obuhvaća istražno razdoblje i razdoblje eksploatacije koje započinje izravnom dodjelom koncesije u slučaju ispunjenja uvjeta za dodjelu iste sukladno točki IV. ove Odluke. Vlada Republike Hrvatske može produljiti rok trajanja dozvole iz stavka 1. ove točke na način i uz uvjete propisane odredbama članka 19. Zakona o istraživanju i eksploataciji ugljikovodika. Računanje početka roka trajanja dozvole iz stavka 1. ove točke počinje teći od dana stupanja ugovora o istraživanju i podjeli eksploatacije ugljikovodika na snagu.

Slijedi obrada dokumenta po koracima (navedeni na početku poglavlja):

1. identificirane tekstualne jedinice su opće imenice i pridjevi, dakle sljedeće riječi (navedene u lematiziranom obliku):

   > dozvola, točka, odluka, rok, godina, istražni, razdoblje, eksploatacija, izravan, dodjela, koncesija, slučaj, ispunjenje, uvjet, isti, vlada, republika, trajanje, stavak, način, propisati, odredba, članak, zakon, istraživanje, ugljikovodik, računanje, početak, dan, stupanje, ugovor, podjela, snaga

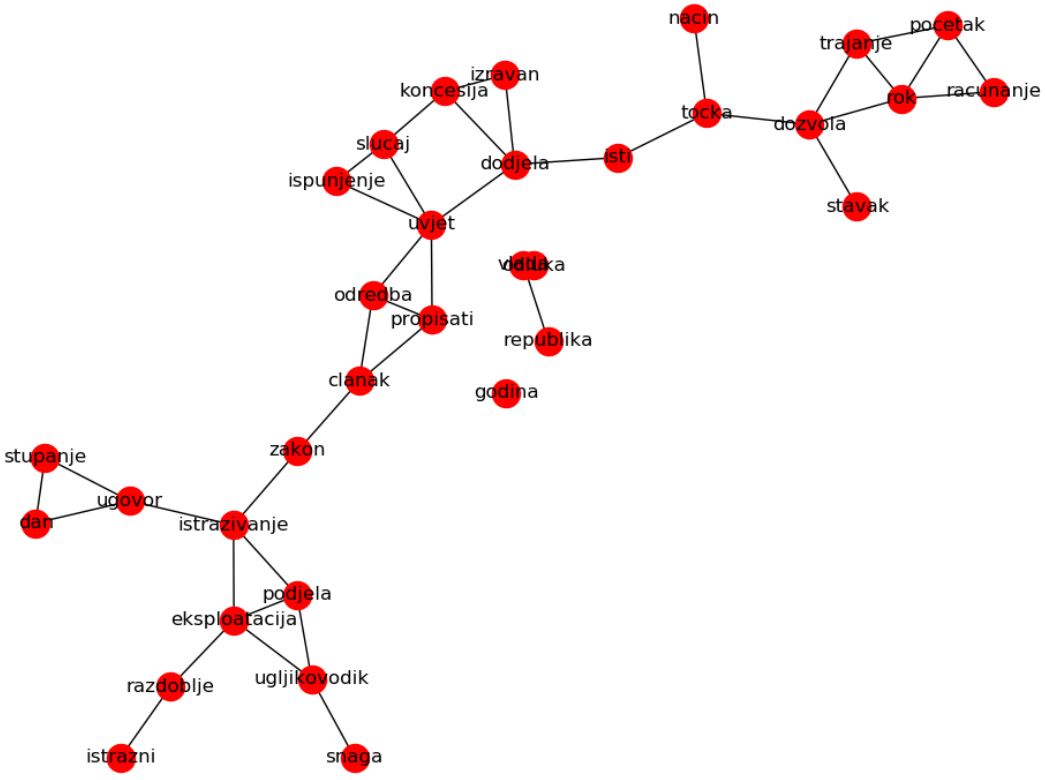
   Ukoliko dvije riječi imaju istu lemu, riječ nije ponovno dodana u graf. Navedene riječi se dodaju u graf kao vrhovi.

2. sljedeće je identificiranje veza između vrhova u grafu, za što je odabrana relacija zajedničkog pojavljivanja, definirana na početku poglavlja 5.3.3 i parametar $N = 2$. Prema toj relaciji identificiraju se sljedeći bridovi:

[5] engl. *co-occurrence*

(dodjela, izravan), (stavak, dozvola), (ugljikovodik, podjela), (trajanje, početak), (ugovor, dan), (eksploatacija, podjela), (ugljikovodik, eksploatacija), (eksploatacija, razdoblje), (isti, dodjela), (članak, propisati), (točka, dozvola), (odredba, uvjet), (uvjet, ispunjenje), (dozvola, trajanje), (početak, računanje), (način, točka), (zakon, članak), (razdoblje, istražni), (republika, vlada), (istraživanje, zakon), (podjela, istraživanje), (snaga, ugljikovodik), (eksploatacija, istraživanje), (uvjet, slučaj), (koncesija, dodjela), (trajanje, rok), (koncesija, izravan), (rok, početak), (dozvola, rok), (istraživanje, ugovor), (ispunjenje, slučaj), (odredba, propisati), (članak, odredba), (rok, računanje), (slučaj, koncesija), (točka, isti), (propisati, uvjet), (stupanje, dan), (ugovor, stupanje), (dodjela, uvjet)

Konstruirani graf je prikazan na slici 3.1



Slika 3.1: Konstruirani graf

3. idući je korak iteriranje algoritma do konvergencije, čime vrhovi dobivaju sljedeće vrijednosti (zbog preglednosti je uključeno samo prvih 10):

Lematizirana riječ	Vrijednost
dozvola	1.5595
točka	1.2653
rok	1.4416
istražni	0.5477
razdoblje	0.9404
eksploatacija	1.5222
izravan	0.7383
dodjela	1.3812
koncesija	1.0405
slučaj	1.0142

4. zadnji je korak silazno sortiranje po njihovoj vrijednosti i selekcija prikladnih vrhova. Uz zahtjev da se iz dokumenta želi izvući 4 ključne riječi, ključne riječi za ovaj dokument bi bile: uvjet, dozvola, eksploatacija, rok.

3.2 tf-idf algoritam

Na početku ove sekcije se uvode pojmovi potrebni za objašnjavanje algoritma.

Definicija 3.2.1. *Konačni multiskup M na skupu S je uređeni par $M = (S, m)$, gdje je $m\colon S \to \mathbb{N}_0$ funkcija za koju je $\sum_{x\in S} m(x)$ konačan broj. Za $x \in S$ se broj $m(x)$ zove kratnost od x. Broj elemenata multiskupa M, u oznaci $|M|$ je $\sum_{x\in S} m(x)$.*

U teoriji dohvata informacija, definira se model vektorskog prostora[6] za reprezentiranje tekstualnih dokumenata, gdje su dokumenti predstavljeni kao vektori. Osnovne jedinice su pojedinačne riječi. Pretpostavka je da su sve riječi međusobno nezavisne. Dokument se gleda kao vreća riječi[7] - multiskup riječi, u kojem nije bitan poredak, samo broj pojava svake riječi. Skup svih dokumenata se naziva korpus.

Neka $D = \{d_1, \ldots, d_N\}$ predstavlja skup svih dokumenata (korpus), a $N = |D|$ broj svih dokumenata, i neka je $W = \{w_1, w_2, w_3, \ldots, w_T\}$ skup svih riječi, u lematiziranom obliku, koje se javljaju u dokumentima iz D te $T = |W|$ kardinalitet tog skupa. Tada se za

[6] engl. *vector space model*
[7] engl. *bag of words*

$j \in \{1, \ldots, N\}$ j-ti dokument može prikazati kao $d_j = (w_{1,j}, w_{2,j}, \ldots, w_{T,j})$.
Neka su $j \in \{1, \ldots, N\}$ i $i \in \{1, \ldots, T\}$ te $d_j \in D$ i $w_i \in d_j$. Uzimajući u obzir da su dokumenti zapravo multiskupovi, $m_{d_j}(w_i)$ označava kratnost riječi w_i u dokumentu d_j.
Tada za $j \in \{1, \ldots, N\}$ i $i \in \{1, \ldots, T\}$ te $d_j \in D$ i $w_i \in W$ vrijedi:

$$w_{i,j} = \begin{cases} m_{d_j}(w_i) & \text{ako } w_i \in d_j \\ 0 & \text{inače} \end{cases} \tag{3.9}$$

tf-idf (engl. *term frequency - inverse document frequency*) je numerička vrijednost koja odražava koliko je bitna riječ u dokumentu unutar korpusa. Često se koristi kod odlučivanja koliko je koja informacija bitna tijekom pretraživanja (u osnovnoj/modificiranoj formi ili kao dio kompleksnijeg sustava). Intuitivno, tf-idf vrijednost riječi je proporcionalna s brojem pojava riječi u određenom dokumentu, a obrnuto je proporcionalna s brojem pojava riječi u cijelom korpusu - time se realizira činjenica da se neke riječi pojavljuju često u svakom dokumentu i nisu specifične za temu dokumenta.

Računanje tf-idf vrijednosti sastoji se od dva dijela:

- računanje frekvencije izraza (engl. *term frequency* - **tf** dio)
- računanje inverzne frekvencije u dokumentima (engl. *inverse document frequency* **idf** dio)

3.2.1 Računanje frekvencije izraza

Računanje frekvencije izraza intuitivno predstavlja koliko je izraz bitan u nekom dokumentu. Najjednostavniji način kako izračunati vrijednost bi bio pobrojati sve pojave riječi u dokumentu.
Neka je $tf\colon W \times D \to \mathbb{R}$. Može se računati na sljedeće načine:

- Booleova "frekvencija":

$$tf(w, d) = \begin{cases} 1 & \text{ako } w_{i,j} > 0 \\ 0 & \text{inače} \end{cases}, \tag{3.10}$$
$$\text{gdje je } w = w_i \text{ za } i \in \{1, \ldots, T\} \text{ i } d = d_j \text{ za } j \in \{1, \ldots, N\}$$

- frekvencija riječi u dokumentu:

$$tf(w, d) = w_{i,j} \text{ , uz uvjete kao u (3.10)} \tag{3.11}$$

- frekvencija riječi u dokumentu prilagođena s brojem riječi u dokumentu:

$$tf(w, d) = w_{i,j} + |d| \text{ , uz uvjete kao u (3.10)} \tag{3.12}$$

- frekvencija riječi skalirana logaritamski:

$$tf(w,d) = \log(1 + w_{i,j}) \text{ , uz uvjete kao u (3.10)} \tag{3.13}$$

- modificirana frekvencija - frekvencija riječi dijeli se s frekvencijom riječi koja se najčešće pojavljuje u dokumentu, u svrhu sprječavanja pristranosti prema dužim dokumentima:

$$tf(w,d) = 0.5 + 0.5 \cdot \frac{w_{i,j}}{\max_{k \in \{1,\dots,T\}}\{w_{k,j}\}} \text{ , uz uvjete kao u (3.10)} \tag{3.14}$$

  Umjesto vrijednosti 0.5 može se staviti bilo koji $K \in \langle 0, 1\rangle$.

3.2.2 Računanje inverzne frekvencije u dokumentima

Računanje inverzne frekvencije je intuitivno mjera koja pokazuje koliko riječ zapravo daje informacije o temi dokumenta - smatra se da riječ dobro opisuje temu dokumenta ukoliko se riječ često pojavljuje u zadanom dokumentu, a rijetko u ostalima. Analogno, ukoliko se riječ pojavljuje često u većini dokumenata, smatra se da ona ne donosi nikakvu informaciju specifičnu za temu dokumenta, tj. da ta riječ pripada tzv. *stop words*[8].

Neka je $idf\colon W \to \mathbb{R}$. idf se računa po sljedećoj formuli:

$$idf(w) = \log \frac{N}{n_w}, \tag{3.15}$$

gdje je N broj svih dokumenata u korpusu, a n_w najčešće zadan formulom:

$$n_w = |\{d \in D \,|\, w \in d\}|. \tag{3.16}$$

n_w predstavlja broj svih dokumenata u kojima se pojavljuje riječ w. Ukoliko je skup svih riječi W zadan odvojeno od skupa dokumenata D (odnosno, skup riječi W nije nastao analizom korpusa), postoji mogućnost da se neka riječ $w \in W$ uopće neće pojavljivati u korpusu, što bi u (3.16) dovelo do dijeljenja s nulom. U tom slučaju se formula (3.16) modificira s:

$$n_w = 1 + |\{d \in D \,|\, w \in d\}|. \tag{3.17}$$

Iz formule (3.15) slijedi da što je veći n_w, vrijednost idf je manja, odnosno, što je riječ češća u dokumentima, to joj je manja idf vrijednost.

Pored zadavanja idf kao u (3.15), postoje i drugi načini, primjerice vjerojatnosna interpretacija, detaljnije opisana u [5].

[8] riječi koje se jako često javljaju zbog specifičnosti jezika, primjerice riječ *the* za engleski jezik

3.2.2.1 Teorijsko objašnjenje idf

Indikator idf je uveden kao mjera *specifičnosti izraza* 1972. i nakon uspostavljanja činjenice da je jako dobra heuristika, došlo je do potrebe da se uspostavi teorijska podloga. Naredna tri desetljeća su posvećena pronalaženju objašnjenja zašto idf tako dobro radi.

Utvrđeno je da su pristupi iz informacije teorija problematični, ali da postoji dobra podloga i za idf i za $tfidf$ u tradicionalnom vjerojatnosnom modelu, o čemu se više može naći u [5].

3.2.3 tf-idf formula

Kombinirajući neku od tf formula (3.10) - (3.14) i idf formulu (3.15), definira se

$$tfidf\colon W \times D \to \mathbb{R}$$

na način:

$$tfidf(w,d) = tf(w,d) \cdot idf(w). \tag{3.18}$$

Vrijednost $tfidf$ za zadanu riječ w i dokument d raste što se češće riječ w javlja u dokumentu d (**tf** dio), a pada ukoliko se riječ w javlja često u cijelom korpusu D (**idf** dio).

3.2.3.1 Primjer primjene tf-idf

Za kraj poglavlja se navodi primjer primjene *tf-idf* algoritma. Za izračunavanje *tf* se koristi formula (3.11), a za *idf* se koristi formula (3.15).

Radi jednostavnosti, riječi se u skupu W neće pamtiti u lematiziranom obliku nego onako kako se pojavljuju u tekstu, a korpus se sastoji od 2 dokumenta ($N = 2$), koji se sastoje od po jedne rečenice:
1. dokument: ”Neka je ovo rečenica i neka je ovo primjer i neka nije.”
2. dokument: ”Neka je ovo drugi.”

Korpus je tada $D = \{d_1, d_2\}$, a skup svih riječi

$W = \{w_1, \ldots, w_7\} =$ {”neka”, ”je”, ”ovo”, ”rečenica”, ”i”, ”primjer”, ”nije”, ”drugi”}.

Reprezentacije dokumenata su tada dane s:

$$\text{Prvi dokument: } d_1 = (3,2,2,1,2,1,1,0)$$
$$\text{Drugi dokument: } d_2 = (1,1,1,0,0,0,0,1)$$

Računa se $tfidf$ za riječ ”neka” i dokument d_1:

$$tf(\text{”neka”}, d_1) = w_{1,1} = 3$$
$$idf(\text{”neka”}) = \log \frac{N}{|\{d \in D \,|\, w \in d\}|} = \log \frac{2}{2} = 0$$
$$tfidf(\text{”neka”}, d_1) = tf(\text{”neka”}, d_1) \cdot idf(\text{”neka”}) = 3 \cdot 0 = 0$$

Sada se računa $tfidf$ za riječ ”primjer” i dokument d_1:

$$tf(\text{”primjer”}, d_1) = w_{6,1} = 1$$
$$idf(\text{”primjer”}) = \log \frac{N}{|\{d \in D \,|\, w \in d\}|} = \log \frac{2}{1} \approx 0.30103$$
$$tfidf(\text{”primjer”}, d_1) = tf(\text{”primjer”}, d_1) \cdot idf(\text{”primjer”}) = 1 \cdot 0.30103 = 0.30103$$

U oba slučaja se vidi djelovanje faktora idf:

- u prvom primjeru, iako riječ ”neka” ima najveću frekvenciju u dokumentu d_1, u konačnici je ocijenjena kao nebitna za prvi dokument, jer se pojavljuje u svakom dokumentu unutar korpusa, što dovodi do toga da joj je idf vrijednost 0.
- u drugom primjeru, iako riječ ”primjer” ima samo jednu pojavu u dokumentu, ocijenjena je kao bitnija od riječi ”neka” zbog idf faktora - riječ ”primjer” se pojavljuje samo unutar dokumenta d_1, pa je ocijenjena specifičnom za taj dokument.

Poglavlje 4

Korišteni alati

Projekt je izrađen kao mrežna aplikacija[1] i koristi sljedeće alate i tehnologije:

- Java
- Eclipse IDE
- Maven
- Spring
- Vaadin
- ReLDI mrežni servis

U nastavku poglavlja detaljnije se opisuje svaka od navedenih stavki.

4.1 Java

Java je u najužem značenju programski jezik, iako podrazumijeva još nekoliko različitih značenja: Java kao platforma, Java kao JVM[2], Java kao JRE[3], Java kao JDK[4], Java kao Javin ekosustav.[5]. Javin virtualni stroj (JVM) je program specifičan za određenu platformu koji izvršava aplikacije pisane u Java programskom jeziku. JRE je dio Javine platforme koji omogućava pokretanje bilo koje Javine aplikacije koja je već prevedena. JDK u sebi

[1] engl. *web application* - programsko rješenje kojem se pristupa putem internet preglednika koristeći internet ili intranet

[2] engl. *Java virtual machine*

[3] engl. *Java runtime environment*

[4] engl. *Java development kit*

[5] Preuzeto iz [7]

uključuje JRE i implementaciju samog prevodioca te pomoćne druge alate.

Kao programski jezik, Java je objektno-orijentiran jezik, široke upotrebe. Jedan od razloga zašto je postao opće prihvaćen jezik je temeljna ideja napiši-jednom-izvrši-bilogdje[6] - ugrađena je potpora za višeplatformnost. To se postiže tako da se program (napisani kod) prvo prevede u strojni jezik virtualnog stroja (engl. *bytecode*) koji zatim interpretira Javin virtualni stroj. Programer tako ne mora brinuti izvodi li se njegov program na 32-bitnom ili 64-bitnom računalu, na operativnom sustavu Windows ili Linux.

Java ima ugrađenu podršku za paralelno programiranje, rad s internetskim protokolima UDP i TCP, distribuirano programiranje te podršku za izradu mrežnih aplikacija.

4.1.1 Javine platforme za programiranje

Sve Javine platforme sastoje se od Javinog virtualnog stroja i sučelja za programiranje aplikacija - API [7]. API je kolekcija softverskih komponenti koje se mogu koristiti kako bi se razvile aplikacije. Postoje 4 platforme za programiranje:

- Java SE (*Standard edition*) - pruža osnovnu funkcionalnost za programiranje u Javi - definirani su osnovni tipovi i objekti Java programskog jezika te klase koje omogućavaju Javinu ugrađenu podršku.
- Java EE (*Enterprise edition*) - nadskup Jave SE, pruža dodatnu podršku za izradu skalabilnih, pouzdanih i modularnih mrežnih aplikacija.
- Java ME (*Micro edition*) - podskup od Jave SE, s posebnom podrškom za izvođenje na manjim uređajima, npr. mobilni uređaji.
- Java FX - platforma za mrežne aplikacije koja koristi podskup standardnog API-ja. Ove aplikacije koriste grafičke kartice kako bi poboljšale performanse.

U ovom je projektu korištena Java EE, verzija 8u191.

4.1.2 Eclipse IDE

Pored Javinog temeljnog razvojnog alata (JDK), dostupne su i grafički orijentirane razvojne okoline, od kojih su danas najpoznatije: NetBeans, Eclipse, IntelliJ IDEA, a NetBeans i Eclipse su besplatne za korištenje. Za razvoj ove aplikacije korišten je Eclipse Photon.

[6] engl. *write once, run anywhere*

[7] engl. *Application Programming Interface*

4.2 Maven

Prevođenje Javinog napisanog koda u strojni jezik predstavlja mukotrpan posao ukoliko se radi o imalo većem broju datoteka. Također, svaki korišteni razvojni okvir[8] u aplikaciji bi se morao ručno referencirati da se kod ispravno prevede.

Postoje dostupni alati za automatizaciju različitih faza u razvojnom ciklusu aplikacije, pa tako i kod prevođenja i korištenja drugih razvojnih okvira. Jedan od najčešće korištenih alata za aplikacije pisane u Java jeziku je Maven. Temelji se na konceptu objektnog modela projekta - POM[9], gdje je projekt zamišljen kao jedan objekt u kojem su opisane sve njegove bitne karakteristike (naziv projekta, verzija, korišteni alati i razvojni okviri). Njegovi glavni objektivi su:

- olakšanje procesa prevođenja
- pružanje uniformnog sustava za prevođenje - svi projekti, neovisno o svojoj veličini ili korištenim tehnologijama se jednako prevode i imaju jednaku strukturu
- pružanje ključnih informacija o projektu na jednom mjestu
- pružanje uputa za razvoj projekta
- lagano ažuriranje na novije verzije

U ovom je projektu korištena Maven 3 verzija alata.

4.3 Spring

Spring je razvojni okvir otvorenog koda[10], čija je temeljna ideja smanjiti kompleksnost pisanja Java EE aplikacija, koje sadržavaju puno *boilerplate* koda[11], prvi puta pušten 2003. godine. Od tada Spring razvija brojne module koji pružaju različite usluge, od kojih su neki:

- modul za aspektno-orijentirano programiranje - pruža podršku za brojne sigurnosne standarde, protokole i alate čime olakšava uvođenje autentifikacije i autorizacije u aplikaciji
- modul za pristup podacima - pruža podršku za pristup bazama podataka

[8] engl. *framework*

[9] engl. *Project Object Model*

[10] engl. *open source*

[11] standardizirani kod potreban za većinu funkcionalnosti koje pruža standardna mrežna aplikacija, nespecifičan za logiku same aplikacije, čime nepotrebno otežava i produžuje pisanje koda

- modul za MVC[12] - olakšava implementaciju poznatog modela za mrežne aplikacije
- modul za transakcije - pruža podršku za olakšano i pouzdano obavljanje transakcija[13]

4.3.1 Spring Boot

Pored modula, Spring razvija i projekte koji služe kao posrednik između programera i modula, gdje se programeru dodatno olakšava upotreba modula, prateći paradigmu konvencija-prije-konfiguracije[14]. Navedenu paradigmu koriste razvojne okoline u svrhu smanjivanja odluka koje programer mora napraviti kako bi konfigurirao projekt i razvojni okvir koji želi koristiti, bez gubljenja fleksibilnosti koje modul standardno pruža. U osnovi, programeru se želi omogućiti da pri konfiguraciji mora specificirati samo one dijelove koji su nekonvencionalni za njegovu aplikaciju, a da se za ostale, standardne dijelove pobrine sam razvojni okvir.

Jedan od takvih projekata je Spring Boot, koji omogućava brže i jednostavnije konfiguriranje i pokretanje aplikacije. Glavna ideja Spring Boot projekta je da se komponente koje su potrebne svakoj aplikaciji (npr. povezivanje s bazom podataka, vođenje dnevnika promjena[15]) automatski uključe u aplikaciju, čime se smanjuje broj konfiguracijskih datoteka i parametara, a programer mora upisati samo nužne parametre (npr. korisničko ime i lozinku za bazu podataka).

U ovom je projektu korišten Spring Boot 2, koji koristi razvojni okvir Spring 5.

4.4 Vaadin

Vaadin je razvojna okolina za korisnička sučelja mrežne aplikacije zasnovana na Java programskom jeziku. Glavna svrha Vaadina je povećati produktivnost programera, postići skalabilnost aplikacije i bolje korisničko iskustvo. Osnovne značajke razvojne okoline Vaadin su:

- za razvoj aplikacije u Vaadinu potrebno je znati samo programski jezik Java
- izvršavanje koda na strani poslužitelja - za razliku od glavnog jezika za razvoj mrežnih aplikacija, JavaScript, koji se izvodi na strani klijenta, sva poslovna logika aplikacije ostaje na samo jednoj strani - strani poslužitelja
- dizajniranje korisničkog sučelja - napisani kod nije potrebno ručno prevoditi u JavaScript kod koji se može izvoditi na strani klijenta

[12] engl. *Model View Controller*
[13] niz funkcija koje se sekvencijalno obavljaju i smatraju kao nerazdvojive cjeline
[14] engl. *convention over configuration*
[15] engl. *log*

- model programiranja zasnovan na komponentama - Vaadin dolazi s gotovim komponentama, koje se mogu ponovno iskoristiti i lagano nadograditi te smanjuju problem kompatibilnosti sa različitim preglednicima
- testiranje i sigurnost - Vaadin nudi alate s kojima se mogu automatizirati testovi za provjeru kvalitete i sigurnosti aplikacije

4.4.1 Vaadin - Spring integracija

Obzirom da je okolina Spring fokusirana na arhitekturu i logiku aplikacije, a okolina Vaadin na izgled aplikacije i ono što korisnik zapravo vidi, napravljen je dodatak koji omogućuje korištenje obje okoline istovremeno, koji također dolazi unutar Spring Boot projekta.

Korištena verzija Vaadin okoline u projektu je 8.

4.5 ReLDI mrežni servis

Kao što je prethodno spomenuto u 1.1, alat za dodavanje oznaka i lematiziranje dostupan je preko grafičkog sučelja na stranici ili kao mrežni servis. Obzirom da je potrebno obraditi velik broj datoteka, korišten je mrežni servis, a za komunikaciju sa servisom i izmjenu podataka korišten je JSON (engl. *JavaScript Object Notation*) format.

Pri korištenju grafičkog sučelja nije zamijećeno ograničenje na duljinu datoteke, dok se pri pozivu servisa datoteka neće obraditi ukoliko je veća od 5kB (~ 4000 znakova).

Poglavlje 5

Implementacija

U ovom poglavlju se predstavljaju faze pri izradi aplikacije te detalji različitih implementacija algoritama raspravljanih u poglavlju 3. Potpuni postupak se nalazi u metodi `processing`, unutar koje se specificiraju parametri algoritma i pozivaju potrebne metode:

Algoritam 2: Funkcija processing

```
//pretprocesiranje zakona
 PretProcesiranjeZakona.saveAllElementsInFiles ();

//poziv web servisa, obavljen van ovog projekta

//ucitavanje u bazu
UcitajTekstZakonaITokenUBazu.ucitajTekstZakona ();
UcitajTekstZakonaITokenUBazu.ucitajTokene ();

lista ← dohvatiSveDokumenteIzBaze ();
//potrebno za tf-idf algoritam
SpremiKorpusUBazu.spremi (lista);

// ...  podesavanje parametara ...
NadjiKljucneRijeci.setLista (lista);
NadjiKljucneRijeci.nadjiTextRank (N, T);
NadjiKljucneRijeci.nadjiTextRankMultipleWindowSize (T, minN, maxN);
NadjiKljucneRijeci.nadjiTfIdf (T);
NadjiKljucneRijeci.nadjiTextrankIdf (N, T);
NadjiKljucneRijeci.nadjiTextrankMulWinIdf (minN, maxN, T);
```

5.1 Struktura podataka

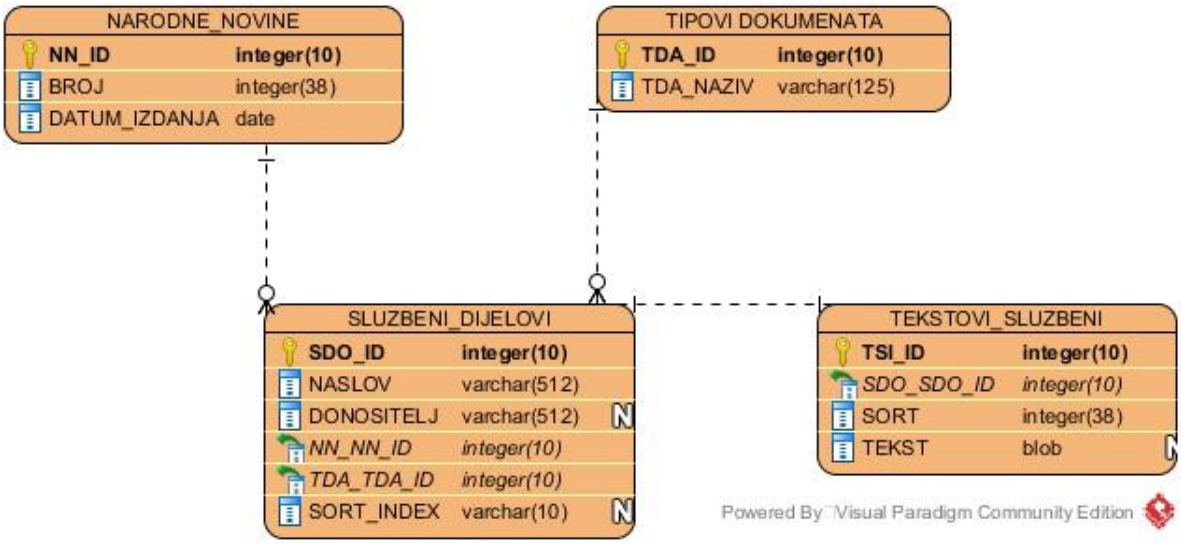
Svi su podaci spremljeni u bazu podataka. Dostupni su podaci za sve Narodne novine od početka siječnja 2015. godine do kraja studenoga 2016. godine i ima ih ukupno 250. Za svaki primjerak u bazi je spremljen jedinstveni ID, broj novina i datum izdanja.

Svaki primjerak novina sadrži jedan ili više službenih dijelova, koji mogu biti različitog tipa. Tipovi dokumenata spremljeni su u zasebnu tablicu, gdje je za svaki primjerak prisutan jedinstveni ID te naziv tipa dokumenta. Ukupno ih ima 22.

Za svaki službeni dio pohranjeni su sljedeći podaci: jedinstveni ID, naslov, donositelj, ID novina kojem pripada, ID tipa dokumenta koji opisuje taj dio te indeks po kojem se službeni dijelovi mogu sortirati unutar istih novina. Ukupno ih ima 5025.

Sami tekst svakog službenog dijela pohranjen je zasebno kao službeni tekst. Za svaki je tekst pohranjen njegov jedinstveni ID, ID službenog dijela u kojem se nalazi, indeks za sortiranje te sami tekst službenog dijela. Obzirom da je službeni tekst samo fizički odvojen od službenog dijela no pripada istoj cjelini, također ih ima 5025.

Odnos prethodno opisanih tablica vidimo na slici 5.1. Za prikaz odnosa koristimo ERD[1], koji modelira entitete i veze među njima.



Slika 5.1: ER-shema

[1] engl. *Entity Relationship Diagram*

5.2 Pretprocesiranje podataka

U ovoj sekciji se navode svi koraci koji su prethodili samoj implementaciji algoritama za nalaženje ključnih riječi.

5.2.1 Spremanje u datoteke

Svi službeni tekstovi su spremljeni kao HTML[2] dokumenti. Svaki dokument, pored ostalih HTML elemenata i CSS selektora, sadržava HTML element naslov i paragraf klase "*story*" u kojem se nalazi sadržaj cijelog dokumenta, osim jednog. Taj je dokument izostavljen iz daljnjeg procesiranja te je nadalje broj službenih tekstova 5024. Neki tekstovi sadržavaju link na datoteke, čiji sadržaj nije uzet u obzir tijekom daljnjeg procesiranja.

Svaki je dokument pročišćen na način da se uzima sadržaj u elementu naslov te pročišćeni tekst iz paragrafa klase "*story*".

Tako pročišćeni dokument se spremao u datoteke od po 4000 znakova (zbog ograničenja na duljinu datoteke, spomenuto u 4.5), ili manje ukoliko se dogodi da bi prekid nakon točno 4000 znakova raspolovio riječ i spremio jedan dio u jednu datoteku, a drugi u drugu.

5.2.2 Poziv mrežnog servisa

Nakon zapisivanja svih tekstova u datoteke prihvatljive dužine, poziva se mrežni servis koji tokenizira i lematizira tekstove.

Servis je prilagođen za rad u programskom jeziku Python pa se ovaj dio razvoja aplikacije nije odvijao pod tehnologijama opisanim u prethodnom poglavlju. Svaka se datoteka šalje na obradu te se kao odgovor dobije tokenizirani i lematizirani tekst koji se sprema u datoteku.

Na slici 5.2 prikazan je tekst (kratkog) službenog dijela. Sadržaj u elementu naslov nije ovdje vidljiv.

VISOKI UPRAVNI SUD REPUBLIKE HRVATSKE

1540

PRESUDA

Slika 5.2: Originalni dokument

[2]engl. *HyperText Markup Language* - prezentacijski jezik za izradu mrežnih stranica.

Na slici 5.3 se vidi odgovor kojeg servis vrati - jedan redak označava jednu riječ, prvi stupac je riječ u originalu, drugi stupac je riječ u lematiziranom obliku i treći stupac je oznaka riječi, koja se još zove i *POS tag*.

```
79  79  Mdc
20.07.2015  20.07.2015  Mdo
Presuda presuda Ncfsn
Visokog visok   Agpmsgy
upravnog    upravan Agpmsgy
suda    sud Ncmsg
Republike   republika   Ncfsg
Hrvatske    Hrvatska    Npfsg
broj    broj    Ncmsn
:   :   Z
UsII-121    UsII-121    Mds
/   /   Z
13  13  Mdc
-   -   Z
6   6   Mdc
od  od  Sg
10. 10. Mdo
lipnja  lipanj  Ncmsg
2015.   2015.   Mdo

VISOKI  visok   Agpmsny
UPRAVNI upravan Agpmsny
SUD sud Ncmsn
REPUBLIKE   republika   Ncfsg
HRVATSKE    hrvatski    Agpfsgy

1540    1540    Mdc

PRESUDA presuda Ncfsn
```

Slika 5.3: Parsirani dokument

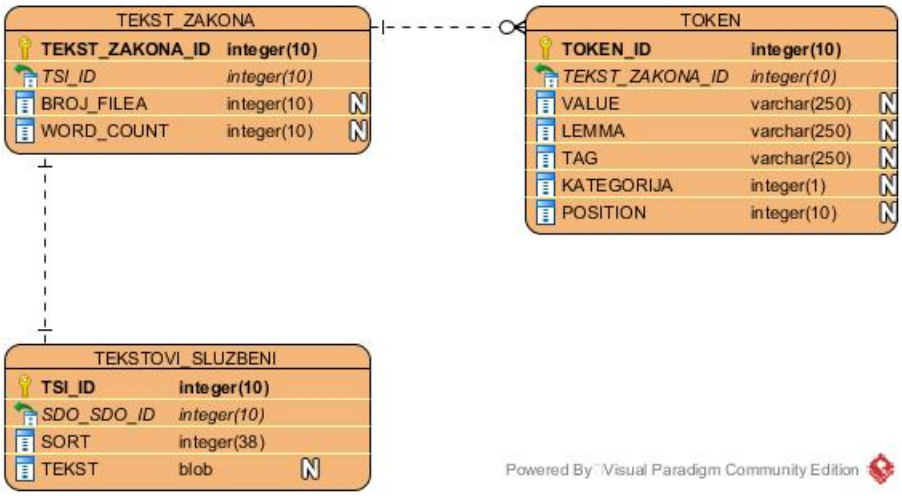
5.2.3 Spremanje dobivenih podataka u bazu podataka

Obzirom da se prikupljeni podaci neće mijenjati, ishod poziva mrežnog servisa može se trajno spremiti u bazu podataka.

U tu svrhu napravljene su dvije nove tablice: $TEKST\_ZAKONA$ i $TOKEN$. Tablica $TEKST\_ZAKONA$ predstavlja mapiranje između jedinstvenih ID-eva za svaki službeni tekst i svih odgovarajućih datoteka u koje su pohranjeni podaci za taj tekst. Tablica sadrži jedinstveni ID, ID službenog teksta, broj datoteke pod kojom je spremljen taj dokument te ukupan broj riječi u tom tekstu.

Tablica $TOKEN$ reprezentira jednu riječ teksta te sadrži sljedeće stupce: jedinstveni ID, ID teksta kojem pripada, originalnu vrijednost, lematiziranu vrijednost, *POS tag*, kategoriju kojoj pripada i poziciju. Kategorija označava kojoj vrsti riječi pripada riječ i ima ih sveukupno 13, a detalji se mogu naći na `http://nl.ijs.si/ME/V5/msd/html/msd-hr.html#msd.categories-hr`. Pozicija označava redni broj kada se zadana riječ pojavila u tekstu.

Na slici 5.4 je prikazana ER-shema novih tablica.



Slika 5.4: ERD za novododane tablice u bazu podataka

Parsiranjem datoteka dobivenih od mrežnog servisa i ubacivanjem u bazu završava se faza pretprocesiranja.

5.3 Implementacija algoritama

U ovoj se sekciji, istim redoslijedom, opisuju implementacije algoritama opisanih u poglavlju 3.

5.3.1 Osnovni TextRank algoritam

U ovoj implementaciji, konstruirao se neusmjereni bestežinski graf, a na vrhove se primjenjivala formula (3.3). Prvi korak pri implementaciji je implementacija klasa koje olakšavaju konstrukciju grafa i primjenu algoritma.

Osnovna je klasa *Vrh*, koja predstavlja jedan vrh u grafu, a sadrži *token* (podatak preuzet iz tablice) te vrijednost vrha u čvoru, inicijalno postavljenu na 1. Dva se vrha smatraju jednakim ukoliko su im *tokeni* jednakih kategorija i jednakih lema.

Iduća je klasa *Brid* koja predstavlja brid u grafu te se sastoji od dva vrha - oni vrhovi koje taj brid spaja. Dva se brida smatraju jednakim ukoliko su im vrhovi jednaki (smjer nije bitan).

Zadnja je klasa *Graf*, koja sadrži skup vrhova i skup bridova te metode za dodavanje, brisanje i nalaženje bridova i vrhova u grafu itd. Za svaki dokument konstruira se po jedan graf.

Za svaki dokument se iz baze podataka dohvaćaju svi *tokeni* koji pripadaju tom dokumentu, u redoslijedu kako su i spremljeni (omogućeno zbog atributa $POSITION$). Nakon dohvata *tokena*, konstruira se graf na način da se za svaki element iz liste provjerava prolazi li sintaktički filter. Ukoliko *token* prolazi sintaktički filter, dodaje se novi vrh u graf te se konstruiraju bridovi za dodani vrh: za zadan N, gleda se N *tokena* koji prethode trenutnom, provjerava se jesu li u istoj rečenici te prolaze li sintaktički filter. Ukoliko prolaze, dodaje se novi brid između novonapravljenog vrha i pronađenog susjeda. Ukoliko se u graf pokuša dodati već postojeći vrh, ne konstruira se novi nego se ažuriraju bridovi na već postojećem vrhu.

Prilikom konstrukcije grafa, obzirom na filter, moguće je da graf ne bude povezan. Primjerice, za rečenicu:

> Plaće rukovodećih državnih službenika iz članka 1. ove Uredbe obračunavat će se prema koeficijentima složenosti poslova utvrđenim ovom Uredbom od 1. veljače 2015. godine.

i za $N = 2$, zadnje dvije riječi koje prolaze filter su *veljače* i *godine* i jedini brid je između te dvije riječi. Kod provjere konvergencije ijednog od ova dva vrha pri prvoj iteraciji algoritma, izmjerena bi greška bila 0 i algoritam bi prestao s radom.

Zbog toga, nakon konstrukcije grafa se provodi čišćenje - nalazi se najveća komponenta povezanosti (formalno definirana u 3.1.6) te se miču vrhovi koji joj ne pripadaju. Za pronalaženje najveće komponente povezanosti koristi se pretraživanje u dubinu[3].

Nakon čišćenja grafa, na vrhove grafa se primjenjuje formula (3.6). Broj iteracija je stavljen na 30. Iteracije algoritma prestaju nakon što je dosegnuta konvergencija, objašnjena formulom (3.8). ϵ je stavljen na vrijednost 0.0001.

Nakon konvergencije algoritma slijedi evaluiranje rješenja. Silazno se sortiraju vrhovi grafa po njihovoj vrijednosti i uzima se prvih T vrhova. Pri pozivu metode TextRank za izvlačenje ključnih riječi, potrebno je specificirati parametre:

N - broj riječi koje se smatraju u okolini za zadanu riječ

T - broj ključnih riječi koje algoritam nađe

Ukoliko se ne žele specificirati, standardne vrijednosti su $N = 2$ i $T = 8$.

[3] engl. *depth-first search*

5.3.2 Osnovni tf-idf algoritam

Za računanje tf vrijednosti koristi se formula (3.14), a za računanje idf vrijednosti se koristi formula (3.15).

Obzirom da je za računanje *idf* vrijednosti za svaku riječ potrebna informacija u koliko je dokumenata iz korpusa ona prisutna, svi bi se dokumenti trebali držati u memoriji tokom izvođenja, što je nemoguće za svih 5024 dokumenata.

Zbog tog ograničenja se obrada dokumenata $tf - idf$ algoritmom sprema u bazu te se spremljeni podaci po potrebi dohvaćaju iz baze. U tu su svrhu kreirane dvije pomoćne tablice, $DOKUMENT$ i $RIJEC$.

Tablica $RIJEC$ predstavlja jednu riječ jednog dokumenta. Sadrži ID *tokena*, ID teksta zakona (ID dokumenta kojem pripada), broj pojava u dokumentu, broj pojava u korpusu te tf, idf i $tfidf$ vrijednosti.

Tablica $DOKUMENT$ predstavlja jedan dokument, sadrži ID teksta zakona (ID dokumenta kojeg predstavlja) i ID najčešće riječi koja se pojavljuje u dokumentu, obzirom da se za računanje tf koristi formula (3.14) za koju je potrebna ta informacija.

Na slici 5.5 se vidi odnos pomoćnih tablica prema entitetima koji predstavljaju dokument i riječi u njemu - $TEKST\_ZAKONA$ i $TOKEN$.

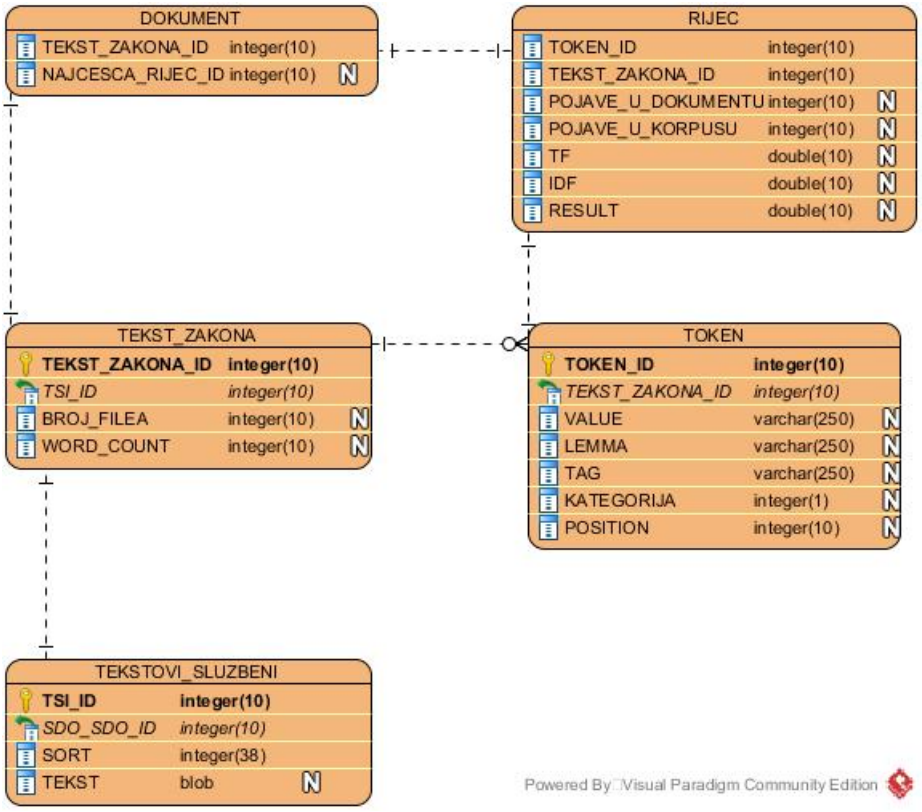
Analogno kao kod implementacije TextRank algoritma, implementiraju se pomoćne klase koje olakšavaju postupak.

Osnovna je klasa $TfRijec$, koja sadrži svoj *token*, broj pojava u dokumentu i tf vrijednost. Analogno kao kod klase Vrh, dvije se riječi smatraju jednakim ukoliko su im *tokeni* jednakih kategorija i jednakih lema.

Iduća je klasa *Dokument*, koja predstavlja jedan dokument iz korpusa, a sadrži skup riječi (multiskup je ostvaren tako da svaka riječ za sebe pamti broj pojava u dokumentu) te informaciju koja se riječ najčešće pojavljuje u dokumentu. *Dokument* još sadrži pomoćne metode za dodavanje riječi / povećavanje broja pojava riječi, pronalaženje riječi, izračunavanje tf vrijednosti za riječi i pronalaženje najčešće riječi u dokumentu.

Zadnja je klasa *Korpus*, koja inicijalizira dokumente, sprema ih u bazu i računa idf vrijednosti na sljedeći način:

- za svaki dokument iz liste:
  - dohvati listu *tokena* koji pripadaju dokumentu, provjeri koji tokeni prolaze sintaktički filter, dodaj ih u dokument kao $TfRijec$
  - za svaku riječ iz dokumenta izračunaj njezinu tf vrijednost
  - dobivene vrijednosti pohrani u bazu
- za svaki dokument iz liste:



Slika 5.5: ERD za pomoćne tablice za izgradnju korpusa

- dohvati listu riječi koji pripadaju dokumentu
- za svaku riječ izračunaj njezin broj pojava u korpusu i njezin idf
- za svaku riječ izračunaj njezin $tfidf = tf \cdot idf$
- dobivene vrijednosti pohrani u bazu

Nakon prethodnih koraka, za izvlačenje ključne riječi ovom metodom potrebno je dohvatiti sve riječi za određeni dokument te ih sortirati silazno prema vrijednosti.

Pri pozivu metode tf-idf, potrebno je specificirati parametar T, koji predstavlja broj ključnih riječi koje algoritam nađe. Ukoliko se ne želi specificirati, standardna je vrijednost $T = 8$.

5.3.3 Višestruki poziv TextRank algoritma s različitim parametrom *N* - *TextRankMulWin*

U sekciji je definiran parametar *N* kao broj riječi koji se smatraju u okolini za zadanu riječ. U literaturi se okolina riječi naziva još i okvir, a *N* je veličina okvira. U originalnom članku [3] je testiranjem utvrđeno da je u većini slučajeva najbolja vrijednost za parametar *N* 2.

Kao što je spomenuto u 5.3.1, pri obradi dokumenata TextRank algoritmom se koristi sintaktički filter. Uzimajući u obzir način na koji su dokumenti pisani (referiranje na druge dokumente po njihovom broju, navođenje datuma) i držeći parametar $N = 2$, konstruirani graf gotovo sigurno postaje nepovezan. Štoviše, riječi bitne za temu dokumenta mogu biti izbačene u tom procesu (jer se nepovezani vrhovi izbacuju iz grafa).

U tu svrhu, TextRank algoritam se modificira na način da se za jedan dokument osnovni TextRank algoritam pozove nekoliko puta s različitim vrijednostima za *N*, u rasponu $minWindowSize - maxWindowSize$. Dobiveni rezultati se kombiniraju na sljedeći način:

Algoritam 3: TextRank s varijabilnim parametrom *N*

```
ulaz : minWindowSize: najmanja veličina okvira,
       maxWindowSize: najveća veličina okvira
izlaz: lista ključnih riječi
finalneKljucneRijeci ← prazna mapa;
za i ∈ {minWindowSize, . . . , maxWindowSize} radi
    vrhoviKandidati ← osnovniTextRank (i);
    za vrh ∈ vrhoviKandidati radi
        value ← vrh.dohvatiVrijednost ();
        ako finalneKljucneRijeci.sadrziKljuc (vrh) :
            finalneKljucneRijeci [vrh ] ← finalneKljucneRijeci [vrh ] + value;
        inače:
            finalneKljucneRijeci [vrh ] ← value
```

Ukoliko ključna riječ već postoji u finalnim ključnim riječima, zbrajaju se vrijednosti već postojeće i pronađene, inače se ključna riječ sprema u mapu sa svojom vrijednosti. Na ovaj se način istovremeno proširuje pretraga ključnih riječi, no i održavaju ključne riječi koje su već ocijenjene kao dobre. Dobivena lista ključnih riječi se sortira silazno po vrijednosti.

U ovoj je implementaciji moguće podešavati parametre:

minN - najmanja veličina okvira

maxN - najveća veličina okvira

T - broj ključnih riječi koje algoritam nađe

Standardne vrijednosti su $minN = 2$, $maxN = 5$ i $T = 8$.

5.3.4 Dodavanje idf vrijednosti TextRank algoritmu - *TextRank-idf*

Obzirom da su dokumenti pisani prirodnim jezikom od strane čovjeka, mogu se pojaviti riječi koje su neispravno napisane. Također, može se dogoditi greška pri obradi prirodnog jezika koja će neki izraz neispravno zabilježiti kao imenicu ili pridjev, a u stvarnosti je nešto drugo. Primjerice, u jednom je dokumentu datum ”16.01.2015” greškom označen kao pridjev sa pripadnom lemom ”16.01.201i”. Na ovakve greške je $tf - idf$ algoritam osjetljiv, jer će ove riječi (iako možda imaju malu tf vrijednost) imati veliku idf vrijednost i biti predložene za ključne riječi.

Također, TextRank algoritam informacije o dokumentu prikuplja isključivo iz njega - nema informaciju o tome koliko je riječ specifična za pojedini dokument.

Zbog prethodno navedenih nedostataka, uvodi se modifikacija na osnovni TextRank algoritam - TextRank algoritmom se dobiju kandidati za ključne riječi. Za svaki se kandidat računa njegova idf vrijednost (točnije, dohvaća se iz baze podataka izračunata vrijednost) te se množi s vrijednosti dobivenom iz TextRank algoritma.

Preciznije, definira se konačna vrijednost vrha $value\colon V \to \mathbb{R}$. Konačna vrijednost vrha $V_i \in V$ dobiva se kombiniranjem formula (3.3) i (3.15):

$$value(V_i) = S(V_i) \cdot idf(V_i). \tag{5.1}$$

$idf(V_i)$ je moguće izračunati zbog načina kako su implementirane klase Vrh i $Rijec$ - objekti od obe klase sadrže isključivo jedan *token* koji je jedinstven za riječ u odnosu na cijeli korpus. Na taj se način može definirati ”jednakost” objekta iz klase Vrh i objekta iz klase $Rijec$ - jednaki su ukoliko sadrže isti *token*. $idf(V_i)$ se dobije tako da se za vrh V_i pronađe odgovarajuća riječ te se uzme njezina idf vrijednost.

Konačna lista ključnih riječi dobiva se kad se lista kandidata silazno sortira po konačnoj vrijednosti.

Parametri koje je moguće podešavati u ovoj implementaciji su:

N - veličina okvira

T - broj ključnih riječi koje algoritam nađe

Standardne vrijednosti su $N = 2$ i $T = 8$.

5.3.5 Dodavanje idf vrijednosti TextRank algoritmu s varijabilnim parametrom N - *TextRankMulWin-idf*

Uzimajući u obzir primjedbe na algoritme u 5.3.3 i 5.3.4, uvodi se konačna modifikacija:

- za zadani dokument, obavi se višestruki poziv TextRank algoritma s različitim parametrom *N*, kao u 5.3.3
- konačna vrijednost vrha dobiva se skaliranjem izračunate vrijednosti s idf vrijednosti vrha, kao u 5.3.4

Kao i u prethodnoj implementaciji, konačna lista ključnih riječi se dobiva silaznim sortiranjem po konačnoj vrijednosti.

Parametri koje je moguće podešavati i njihove standardne vrijednosti su isti kao kod višestrukog poziva TextRank algoritma 5.3.3.

5.4 Postprocesiranje podataka

Neovisno o primijenjenom algoritmu, faza postprocesiranja je ista. Ishod svakog algoritma je lista ključnih riječi, za koju se još provjerava postoje li elementi te liste koji su si susjedni u dokumentu - ako postoje, spajaju se u jednu ključnu riječ.

Za sve rezultate napravljena je dodatna tablica $KLJUCNE\_RIJECI$ koja ima jedinstveni ID, ID dokumenta čije su ključne riječi spremljene te kolone za spremanje rezultata algoritama:

- $KW\_TEXTRANK$ - ključne riječi osnovnog TextRank algoritma, opisanog u 5.3.1
- KW_TFIDF - ključne riječi osnovnog $tf - idf$ algoritma opisanog u 5.3.2
- $KW\_TEXTRANK\_MUL\_WIN\_SIZE$ - ključne riječi *TextRankMulWin* algoritma, opisanog u 5.3.3
- $KW\_TEXTRANK\_IDF$ - ključne riječi *TextRank-idf* algoritma, opisanog u 5.3.4
- $KW\_TEXTRANK\_MUL\_WIN\_IDF$ - ključne riječi *TextRankMulWin-idf* algoritma opisanog u 5.3.5

5.5 Rezultati

Obzirom da su podaci na kojima se radilo neoznačeni, ne mogu se izračunati nikakve mjere točnosti za prethodno opisane algoritme. Umjesto toga, prikazuje se primjer na

jednom sažetku koji se nalazi van ovog korpusa i ima označene ključne riječi te usporedba korištenih algoritama i jedan primjer iz ovog korpusa.

5.5.1 Rezultati na sažetku s označenim ključnim riječima

Sažetak i njegove ključne riječi su preuzete s CROSBI portala[4], a glasi:

> Fraktali su geometrijski objekti čija je fraktalna dimenzija strogo veća od topološke dimenzije. Drugačije rečeno, to su objekti koji daju jednaku razinu detalja neovisno o razlučivosti koju koristimo. Fraktali imaju tri važna svojstva koja ih karakteriziraju: samosličnost, fraktalna dimenzija i oblikovanje iteracijom. Nakon uvoda, u drugom poglavlju (Fraktali i kaos) uz fraktale je obrađena i tema povezanosti fraktala i teorije kaosa, te deterministčkog kaosa – određenog i definiranog kaosa. U trećem poglavlju (Fraktali) je opisan način kreiranja fraktala uz pomoć matematičkih funkcija. Dani su jednostavni primjeri i objašnjeni glavni pojmovi vezani uz same fraktale. Na kraju poglavlja detaljno je opisano sedam osnovnih vrsta fraktala. U četvrtom poglavlju (Primjena fraktala) dan je pregled primjene fraktala u obradi slika, te dato je nekoliko primjera i pojašnjenja metoda koje se koriste i njihovih primjena u računalnoj grafici. U petom poglavlju (Program) detaljno su obrađeni kodovi potrebni za iscrtavanje pojedinih fraktala, te na kraju prikazani su primjeri iscrtavanja nekih od fraktala i njegove mogućnosti.

Pripadne ključne riječi glase: ”Fraktali; interaktivni program”.

Obzirom da $tf - idf$ algoritam vrijednost riječi određuje na temelju cijelog korpusa, a ovdje je dostupan samo jedan dokument (s različitom tematikom od one koja se nalazi u korpusu), prikazat će se samo rezultati TextRank i *TextRankMulWin* algoritma, za $T = 4$ ključne riječi.

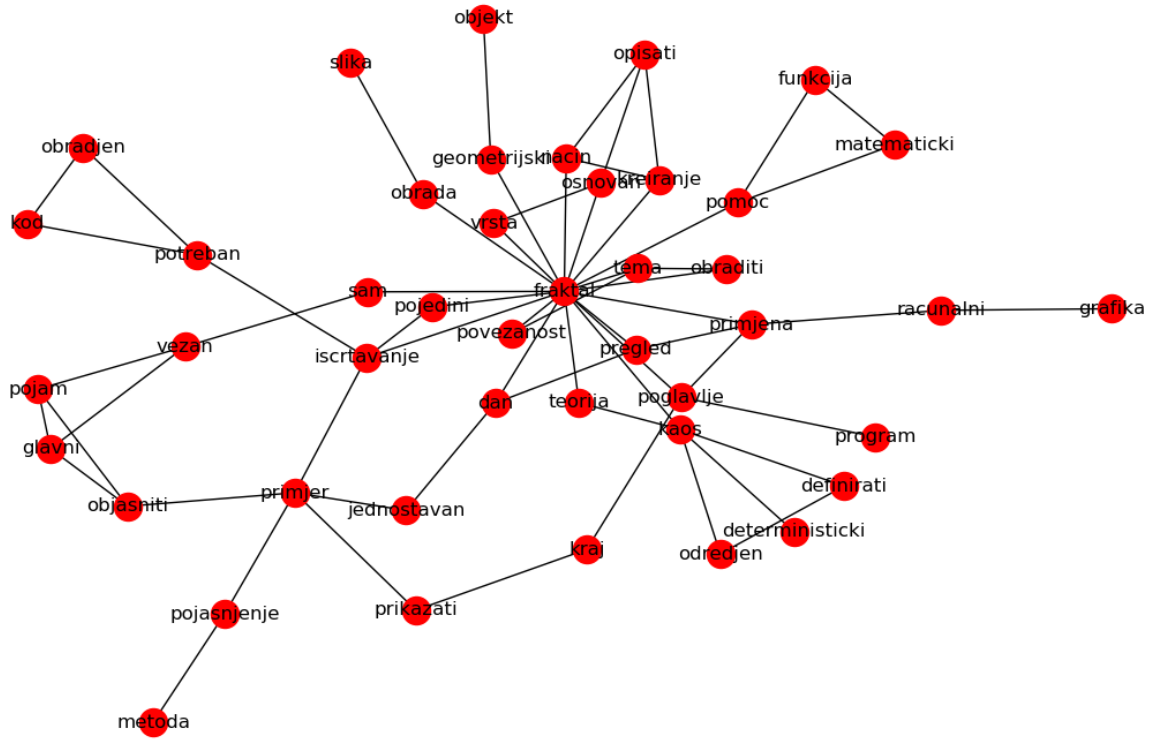
Graf sažetka za TextRank algoritam s parametrom $N = 2$, nakon pronalaženja najveće komponente povezanosti, je prikazan na slici 5.6

Pronađene ključne riječi su sljedeće:

- TextRank algoritam: fraktal, kaos, primjer, poglavlje
- *TextRankMulWin* algoritam: fraktal, fraktalni dimenzija, primjer, primjena

Vrhovi koji su izbačeni iz TextRank algoritma su: fraktalni, samosličnost, dimenzija, topološki, velik, oblikovanje, iteracija, reći, jednak, detalj, razina, razlučivost, važan, svojstvo, uvod, dati, mogućnost.

[4] dostupno na stranici `https://www.bib.irb.hr/`



Slika 5.6: Graf sažetka za TextRank s parametrom $N = 2$

Na ovom se primjeru vidi kako djeluje poboljšanje algoritma - one riječi koje su izbačene (fraktalni, dimenzija) od strane TextRank algoritma su, po *TextRankMulWin* algoritmu, ocijenjene kao ključne riječi, a bolje opisuju sažetak nego što ga opisuje riječ kaos.

Također, vidi se i nedostatak metoda za izvlačenje ključne riječi - pripadna druga ključna riječ sažetka glasi ”interaktivni program”, no nijedan od opisanih algoritama ju sigurno neće naći jer se navedena fraza ne nalazi u danom tekstu.

5.5.2 Usporedba korištenih algoritama

Sve su ključne riječi u konačnici spremljene sortirano silazno po njihovim konačnim vrijednostima - očekuje se da je prva ključna riječ (ili kombinacija istih, ukoliko je došlo do spajanja) najbitnija. Svih 5 algoritama imaju identičnu prvu ključnu riječ za 2053 dokumenata, odnosno u 40.86% slučajeva.

U ostalim se usporedbama mjerila općenita sličnost između nađenih ključnih riječi.

Ključne riječi se segmentiraju ukoliko su spojene i njihov se poredak zanemaruje (primjerice, ukoliko je jedan algoritam na prvo mjesto smjestio "ključna riječ", a drugi na šesto mjesto "riječ" te sedmo "ključna", u ovoj se usporedbi to smatra kao podudaranje). Za svaki se dokument dijeli ukupan broj pojava zajedničkih ključnih riječi sa svim pojavama svih ključnih riječi. Primjerice, neka su pokrenuta $N = 2$ algoritma, algoritam A i algoritam B i nađene sljedeće ključne riječi:

algoritam A: krv uređaj, članak, pravilnik, popis, broj, dan, nov, tekst
algoritam B: krv uređaj, pomagalo hrvatski zavod, članak, pravilnik, popis, zdravstven osiguranje, stavak, količina

Zajedničke ključne riječi za oba algoritma su: krv, uređaj, članak, pravilnik i popis - ima ih 5, dakle ukupan broj njihovih pojava je $5 \cdot N = 10$, a ukupan broj svih ključnih riječi je 21. Dakle, mjera sličnosti algoritama A i B je $\frac{10}{21} = 0.4762$.

Za konačnu mjeru sličnosti algoritama A i B se uzima aritmetička sredina po svim dokumentima iz korpusa.

Napravljene su sljedeće usporedbe po prethodno opisanoj mjeri:

Algoritam 1	Algoritam 2	Rezultat
TextRank	*TextRankMulWin*	82.5%
TextRank	$tf - idf$	67.79%
TextRankMulWin	$tf - idf$	75.14%
$tf - idf$	*TextRank-idf*	67.79%
TextRank	*TextRank-idf*	100%
TextRank	*TextRankMulWin-idf*	82.53%
TextRankMulWin	*TextRankMulWin-idf*	93.97%
$tf - idf$	*TextRankMulWin-idf*	75.04%

Naposljetku je uspoređeno koliko svih 5 algoritama imaju zajedničkih ključnih riječi: 63.11%.

Iz prethodnog se vidi da je izračunata mjera za TextRank i $TextRank - idf$ algoritme 1.0 - algoritmi nalaze identične ključne riječi, a dodavanje idf faktora originalnom TextRank algoritmu nema nikakav utjecaj, osim da eventualno permutira nađene ključne riječi.

Kod *TextRankMulWin* i *TextRankMulWin-idf* algoritma dodavanje idf faktora ipak igra neku ulogu, obzirom da je njihova sličnost 93.97%. Kao što je objašnjeno, pozivanjem višestrukog originalnog algoritma se učvršćuju već nađene ključne riječi, no i daje se šansa drugim, novopronađenima, na koje djelovanje idf faktora može imati pozitivnog učinka.

5.5.3 Rezultati na dokumentu iz korpusa

U konačnici, slijedi cjelovit primjer rada algoritama na jednom dokumentu.

Na slici 5.7 je naveden jedan dokument. Nađene ključne riječi po algoritmima su sljedeće:

Algoritam	Nađene ključne riječi
TextRank	donošenje prostoran plan park priroda, izvoran tekst utvrditi, ispravak odluka, izvršen usporedba, pogreška
TextRankMulWin	donošenje prostoran plan park priroda, samoborski gorje hrvatski sabor, izvoran tekst utvrditi, izvršen usporedba, ispravak odluka, narodni novina broj, objaviti, provođenje
$tf - idf$	donošenje prostoran plan park priroda, samoborski gorje hrvatski sabor, ispravak odluka, ležaj, stavak, žumberak, oštrc, dol
TextRank-idf	donošenje prostoran plan park priroda, izvoran tekst utvrditi, ispravak odluka, izvršen usporedba, pogreška
TextRankMulWin-idf	donošenje prostoran plan park priroda, samoborski gorje hrvatski sabor, izvoran tekst utvrditi, izvršen usporedba, odluka, objaviti

Kao što se vidi, svi su algoritmi našli identičnu prvu ključnu riječ. Nadalje, vidi se da *TextRankMulWin* donosi bolje rješenje od TextRank algoritma, obzirom da je našao više ključnih riječi koji odgovaraju naslovu dokumenta.

Kao što je spomenuto u 5.3.4, $tf - idf$ algoritam je osjetljiv na pogreške - ovdje je ”Oštrc” neispravno označen kao opća imenica i rijetko se pojavljuje u dokumentima (točnije, ovo je jedina pojava te riječi) što podiže njezinu idf vrijednost te ona završava označena kao ključna riječ.

Nadalje se vidi da su ključne riječi $TextRank-idf$ i TextRank algoritma ovdje identične, tj. da idf faktor nije utjecao dovoljno čak ni na permutaciju pronađenih ključnih riječi. Naposljetku, ključne riječi *TextRankMulWin-idf* algoritma prema očekivanju najbolje opisuju temu dokumenta - što je već dobro napravio *TextRankMulWin* algoritam, uz dodatno poboljšanje gdje se miču često korištene riječi u cijelom korpusu - broj, provođenje, narodni, novina, ispravak.

HRVATSKI SABOR

99

Nakon izvršene usporedbe s izvornim tekstom utvrđena je pogreška u Odluci o donošenju prostornog plana Parka prirode Žumberak – Samoborsko gorje koja je objavljena u »Narodnim novinama« broj 125 od 27. listopada 2014. te se daje

ISPRAVAK

ODLUKE O DONOŠENJU PROSTORNOG PLANA PARKA PRIRODE ŽUMBERAK – SAMOBORSKO GORJE

U Odredbama za provođenje članak 84. stavak 1. treba glasiti:

(1) Postojeći kapacitet izletišta (T4) je:

Ekoselo Žumberak u Koretićima	30 ležaja
Seoski turizam »Lijepo brdo Žumberak« u Pećnom	6 ležaja
Oštrc	
Veliki dol	
Japetić	46 ležaja
Poljanice	
Ukupno:	82 ležaja

Klasa: 011-03/14-01/05

Zagreb, 13. siječnja 2015.

Tajnik Hrvatskoga sabora

Slaven Hojski, dipl. iur., v. r.

Slika 5.7: Primjer dokumenta

Bibliografija

[1] Jon M. Kleinberg, *Authoritative Sources in a Hyperlinked Environment*, J. ACM **46** (1999), br. 5, 604–632, ISSN 0004-5411, http://doi.acm.org/10.1145/324133.324140.

[2] Nikola Ljubešić, Filip Klubička, Željko Agić i Ivo Pavao Jazbec, *New Inflectional Lexicons and Training Corpora for Improved Morphosyntactic Annotation of Croatian and Serbian*, Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016) (Paris, France) (Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk i Stelios Piperidis, ur.), European Language Resources Association (ELRA), may 2016, ISBN 978-2-9517408-9-1 (engleski).

[3] R. Mihalcea i P. Tarau, *TextRank: Bringing Order into Texts*, Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing, July 2004.

[4] Lawrence Page, Sergey Brin, Rajeev Motwani i Terry Winograd, *The PageRank Citation Ranking: Bringing Order to the Web*, 1999.

[5] Stephen Robertson, *Understanding inverse document frequency: On theoretical arguments for IDF*, Journal of Documentation **60** (2004), 2004.

[6] Ian H. Witten, Eibe Frank i Mark A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011, ISBN 0123748569, 9780123748560.

[7] Čupić Marko, *Programiranje u Javi*, dostupno na http://java.zemris.fer.hr/nastava/opjj/book-2015-09-30.pdf (studeni 2018.).

Sažetak

U ovom je radu predstavljen jedan način kako poboljšati pretraživanje po dokumentima pisanim prirodnim jezikom - otkrivanjem ključnih riječi dokumenata. Ukratko se priča o obradi prirodnog jezika, važnoj disciplini kod analize dokumenta. Zatim se priča o procesu otkrivanja ključnih riječi i podjeli metoda. Detaljnije se obrađuju metode korištene pri izradi aplikacije: TextRank i $tf - idf$ algoritam.

Prije opisa implementacije, navode se i ukratko opisuju korišteni alati i tehnologije za izradu aplikacije. Zatim se predstavlja postupak izrade aplikacije, koji se sastoji od pretprocesiranja, primjene algoritama (osnovna dva i njihove tri modifikacije) te postprocesiranja. Naposljetku se navodi usporedba rezultata te primjer dokumenta i nađenih ključnih riječi.

Summary

This thesis presents one way of improving the process of searching documents written in natural language - by discovering keywords. It starts with a brief description of natural language processing, a sub-field of computer science, information engineering, and artificial intelligence that is very important for text analysis. Next chapter presents keyword extraction and its classification of methods. Two methods used in application are discussed in detail: TextRank and $tf - idf$ algorithm.

Before describing the implementation process, a list and a short description of used tools and technologies in application is given. Then follows the description of implementation process, which consists of pre-processing, application of algorithms (two basic and their three modifications) and post-processing. Finally, comparison between used methods and an example is given.

Životopis

Lena Kamenjaš je rođena 21.9.1995. u Wuppertalu, Savezna Republika Njemačka. U Orašju, gdje je i živjela, je pohađala osnovnu školu. Svoje srednjoškolsko obrazovanje nastavlja u Županji, u općoj gimnaziji. Zatim, 2013. godine, upisuje preddiplomski studij Matematika na matematičkom odsjeku Prirodoslovno-matematičkog fakulteta, Sveučilište u Zagrebu. Po završetku preddiplomskog studija, na istom fakultetu upisuje diplomski studij, smjer Računarstvo i matematika, 2016. godine.