Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gregor Pevec

Nadgradnja spletnega orodja za analizo besedilnih zbirk TextExplore

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Matija Marolt

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V okviru diplomske naloge se seznanite s tematiko analize tekstovnih zbirk z metodami obdelave naravnega jezika. Spletno orodje za anlizo besedilnih zbirk TextExplore nadgradite s pogledi, ki omogočajo vizualizacijo dokumentov glede na čas ali lokacijo nastanka, ter vizualizacijo vsebine dokumenta v obliki drevesnega prikaza.

Zahvalil bi se mentorju izr. prof. dr. Matija Maroltu za vse nasvete in pomoč.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application Programming Interface	Aplikacijski programski vmesnik
CSS	Cascading Style Sheets	Kaskadne stilske podloge
HTML	HyperText Markup Language	Jezik za označevanje nadbesedila
JS	JavaScript	JavaScript
JSON	JavaScript Object Notation	Objektna notacija JavaScript
REST	Representational State Transfer	Prenos predstavitvenega stanja
SQL	Scripted Query Language	Strukturiran povpraševalni jezik

Povzetek

Naslov: Nadgradnja spletnega orodja za analizo besedilnih zbirk TextExplore

Avtor: Gregor Pevec

Cilj diplomskega dela je bila nadgradja obstoječe spletne aplikacije z več interaktivnimi vizualizacijami podatkov, ki uporabniku omogočajo analizo uvožene zbirke podatkov. Spletna aplikacija je zgrajena z ogrodjem AngulaJS, za hrambo in iskanje pa uporablja podatkovno bazo ElasticSearch. Pred implementacijo smo raziskali področja vizualizacije besedilnih zbirk in procesiranja naravnega jezika. Ugotovili smo, da obstaja več vrst in tipov vizualizacij besedilnih zbirk. Za predstavitev podatkov smo se odločili implementirati tri različne vizualizacije. Vsaka predstavlja podatke uvožene zbirke iz drugega zornega kota.

Ključne besede: vizualizacija podatkov, procesiranje naravnega jezika, spletna aplikacija.

Abstract

Title: Improvements of the TextExplore web-based tool for text corpora analysis

Author: Gregor Pevec

The aim of this diploma thesis was to upgrade an existing web application with several interactive data visualizations that enable the user to analyze the imported data corpus. The web application has been build with an AngularJS framework: for data storage and retrieval it uses the ElasticSearch database. Prior to implementation, the areas of text visualization and processing of the natural language were examined. We have found that there are several types of text visualization. We implemented three different visualisations. Each visualisation shows data of imported corpus from a different angle.

Keywords: data visualization, natural language processing, web application.

Poglavje 1

Uvod

Vizualizacija podatkov z razvojem novih tehnologij in vedno večjega števila podatkov dobiva nov pomen. Podatki so v sodobnem svetu zelo pomembni, zato jih moramo prikazati tako, da iz njih dobimo čim več uporabnih informacij. Eden izmed načinov, kako iz velike količine podatkov prikazati koristne informacije, je uporaba vizualizacije. Z vizualizacijo omogočimo razumevanje informacij, ki nam na prvi pogled niso vidne. Podatki so pogosto v naravnem oziroma govorjenem jeziku, ki ga računalnik ne razume, zato jih s procesom analize naravnega jezika pretvorimo v računalniku razumljivo obliko. Takšne podatke lahko nato učinkovito prikažemo z interaktivnimi vizalizacijami. V diplomskem delu smo razvili več interaktivnih vizualizacij, s katerimi prikažemo pomembne informacije iz uvožene zbirke podatkov.

1.1 Vizualizacija

Vizualizacija je predstavitev podatkov v obliki, ki je za uporabnika lažje razumljiva. Podatke in informacije uporabniku posredujemo na interaktiven način, z uporabo vizualnih elementov, kot so grafikoni in zemljevidi. S takšnim prikazom omogočimo lažje razumevanje, hkrati pa ne izgubimo vrednosti podatkov.

Glavni namen vizualizacije je posredovanje informacij na jasen in učinkovit

način, kar ne pomeni, da mora biti vizualizacija dolgočasna, da bi bila funkcionalna [17]. Za uporabnika podatki postanejo uporabni šele takrat, ko so prikazani na razumljiv način. Z vizualizacijo skušamo iz velike količine nestrukturiranih podatkov izluščiti bistvo ter tako zagotoviti prikaz najpomembnejših podatkov. To najlažje storimo s predstavitvijo podatkov v obliki grafov, grafikonov in drugih vizualnih metod, ki uporabniku pomagajo odkrivati skrite vzorce v podatkih [13].

V nasprotju s splošnim razmišljanjem ima vizualizacija podatkov dolgo zgodovino. Njene korenine segajo v najzgodnejše obdobje oblikovanja zemljevidov in vizualnih upodobitev ter kasneje v obdobje tematske kartografije in statističnih prikazov na področjih medicine in znanosti. Med najzgodnejše grafične prikaze štejemo prikaz premikanja nebesnih teles skozi čas, ki je bil ustvarjen v 10. stoletju. Za prvo vizualno predstavitev statističnih podatkov štejemo graf prikaza razdalje v zemljepisni dolžni od Toleda do Rima, ki ga je ustvaril Michael Florent van Langren leta 1644. Z razvojem statistike se je razvila ideja o grafični predstavitvi podatkov. William Playfair se pogosto šteje za izumitelja večine danes najpogosteje uporabljenih grafičnih prikazov. Najprej je leta 1786 razvil črtni in stolpični grafikon kasneje, leta 1801, pa še tortni diagram. Naslednji večji mejnik je bilo devetnajsto stoletje, ki ga štejemo za zlato dobo statistične grafike. V tem obdobju so se v znanstvenih revijah začeli pojavljati grafični prikazi analize podatkov. Razvila se je tudi večina sodobnih grafikonov. Zamisel o vizualizaciji podatkov se je razvijala in s prihodom računalnikov in hitrim razvojem tehnologij naredila nov velik korak naprej. Danes lahko velike količine podatkov obdelamo z visiko zmogljivo programsko opremo, kar zopet odpira nove možnosti na področju vizualizacije podatkov [18].

Profesor Edward Tufte je v svoji knjigi leta 1983 opisal načela in pristope za učinkovit prikaz vizualizacije. Dobre vizualizacije naj:

- prikažejo podatke,
- gledalca spodbudijo k razmišljanju o vsebini in ne o metodologiji, grafičnem oblikovanju ali nečem drugem,

- ne popačijo podatkov,
- predstavijo veliko številk na majhnem prostoru,
- uskladijo prikaz velikega nabora podatkov,
- spodbudijo uporabnika, da primerja podatke,
- prikažejo podatke na več nivojih, od najširšega do najožjega,
- prikažejo razumljiv in jasen namen,
- bo povezana s statističnim in besednim opisom podatkov [22].

1.2 Vizualizacija besedilnih zbirk

Z večjo razpoložljivostjo elektronskih dokumentov, kot so spletne strani, spletna mesta z novicami in članki, se pojavlja tudi vse večja količina informacij. Pri tem se srečujemo z izzivom, kako učinkovito odkriti uporabne informacije iz tako velikih besedilnih zbirk, ne da bi pri tem podrobno prebrali vsak dokument. Zato do informacij poskušamo priti z uporabo vizualizacije. Poznamo več tehnik vizualizacije besedilnih zbirk, ki jih lahko v osnovi razvrstimo v tri kategorije:

- vizualizacija vsebinskih podobnosti dokumentov,
- vizualizacija vsebine dokumenta,
- vizualizacija čustev dokumenta [16].

1.2.1 Vizualizacija vsebinskih podobnosti dokumentov

Vizualizacija vsebinskih podobnosti dokumentov je tradicionalna metoda za predstavitev povzetka besedilne zbirke. Tašne vizualizacije dokumentov so predstavljene kot točke na zaslonu. Velikost in barva točke sta odvisni od pomembnosti in pomena dokumenta [16]. Najuspešnejši primer takšne

vizualizacije je orodje *Google Ngram Viewer*, ki temelji na pogostosti uporabe določene besedne zvezi v dokumentih, objavljenih v različnih letih. Ko uporabnik vnese besedno zvezo, orodje v bazi podatkov poišče ujemanja in ustvari graf. Ta prikaže razmerje med frekvenco in objavljenim letom, kar nam omogoča, da za neko besedno zvezo pridobimo njen razvoj skozi leta. Orodje za iskanje besedne zveze uporablja besedilno zbirko, ki vsebuje več kot 5 milijonov knjig [7].

1.2.2 Vizualizacije vsebine dokumenta

Vizualizacije takšnega tipa predstavijo vsebino dokumenta. Prikažejo samo vsebino dokumenta ali značilnost, kot je na primer dolžina stavka. Primera takšne vizualizacije sta *tag cloud* in *word tree*. *Tag cloud* se uporablja za prikaz ključnih besed, ponavadi na spletnih mestih. Pogosteje uporabljene besede dokumenta prikažemo z večjo pisavo in tako omogočimo hitro zaznavanje besed, ki v dokumentu najbolj izstopajo. Pri *tag cloud* vizualizaciji so besede razvrščene naključno; vizualizacija ne razkrije besednega razmerja [9]. Za prikaz razmerja med besedami uporabimo *word tree*, ki za izbrano besedo prikaže kontekste, v katerih se nahaja znotraj dokumenta. Takšen način vizualizacije smo uporabili v aplikaciji, za prikaz vsebine dokumenta [23].

1.2.3 Vizualizacija čustev dokumenta

Vizualizacija predstavi analizo čustev in razpoloženj dokumenta. Na splošno je cilj takšne vizualizacije določiti odnos govorca ali pisatelja do določene teme oziroma celotnega dokumenta. Uporablja se za analiziranje, na primer podatkov družbenih omrežij za potrebe trženja [16].

1.3 Analiza in procesiranje naravnega jezika

Procesiranje naravnega jezika (ang. *Natural language processing ali skrajšano NLP*) je podporočje umetne inteligence. Področje se osredotoča na omogočanje računalnikom, da razumejo in obdelujejo človeške jezike. Računalniki lahko veliko hitreje in učinkoviteje obdelajo strukturirane podatke kot ljudje, vendar ljudje za komuniciranje uporabljamo podatke v nestrukturiranih oblikah, s katerimi računalniki ne znajo delati, ker za obdelavo takšnih podatkov ni standardiziranih tehnik [21].

Strukturirani podatki so sestavljeni iz jasno definiranih podatkovnih tipov. Primer takšnih podatkov je tabela v relacijski podatkovni bazi. Nestrukturiranim podatkom ni mogoče jasno določiti tipa, ker ni jasne povezave med njimi. To so običajno besedila, tudi video datoteke in besedila iz družbenih omrežij [8]. Takšne podatke običajno shranjujemo v nerelacijske podatkovne baze, kot je ElasticSearch.

Področji procesiranja naravnega jezika in računalniške obdelave besedil se hitro razvijata z ravojem novih računalniških tehnologij. S povečanjem števila podatkov se povečuje tudi potreba po njihovem boljšem razumevanju in analizi. V prihodnosti lahko pričakujemo še večji razcvet področja.

1.3.1 Tematska analiza

Pri procesiranju naravenga jezika pogosto poskušamo iz besed in stavkov dokumenta razbrati temo, kar storimo s procesom modeliranja tem (ang. *Topic modeling*). Modeliranje tem je vrsta statističnega modeliranja za odkrivanje abstraktnih tem, ki se pojavljajo v zbirki dokumentov [20]. V dokumentu, ki se nanaša na določeno temo, pričakujemo, da se bodo nekatere besede pojavile večkrat. Tako v dokumentu, ki govori o računalništvu, pričakujemo, da se bo vsaj nekajkrat pojavila beseda računalnik. Za posamezni dokument lahko običajno določimo več različnih tem z različnim odstotkom ujemanja.

Latent Dirichlet Allocation ali skrajšano LDA je eden izmed najpreprostejših modelov za določanje teme iz mešanice besed. Na modelu LDA teme-

ljijo tudi številni drugi modeli. LDA je verjetnostni model zbirke dokumentov oziroma korpusa. Osnovna ideja modela je, da so dokumenti predstavljeni kot naključne mešanice tem. Vsak dokument se obravnava kot mešanica tem in vsaka tema kot mešanica besed [19] [15].

Delovanje modela LDA lahko opišemo z dvema praviloma. Prvo pravilo zagovarja, da je vsak dokument mešanica naključnih tem. To pomeni, da ima lahko v modelu z dvema temama prvi dokument 90 % ujemanje s temo A in 10 % ujemanje s temo B, mentem ko ima drugi dokument 30 % ujemanje s temo A in 70 % ujemanje s temo B. Drugo pravilo govori o tem, da je vsaka tema sestavljena iz mešanice besed. Če za primer vzamemo model dveh dokumentov, enega na temo politike in drugega na temo zabave, pričakujemo da bodo najpogostejše besede v prvem, na temo politike, *predsednik*, *parlament* in *vlada*, v drugem dokumentu, na temo zabave pa besede *film*, *televizija* in *filmski igralec*. Pomembno si je zapomniti tudi, da se lahko nekatere besede pojavijo v več temah hkrati. Primer takšne besede bi bil lahko *denar*. LDA uporablja obe pravili hkrati. V zbirki dokumentov išče mešanice besed, ki so povezane z določeno temo, hrati pa določa mešanice tem, ki opisujejo vsak dokument [10] [15].

1.4 Cilji

Cilj diplomskega dela je bila nadgradja obstoječe aplikacije. Obstoječa aplikacija omogoča uvoz zbirke dokumentov v bazo ElasticSeach. Pri uvozu se za zbirko podatkov izvede proces tematske analize z orodjem Mallet. Rezultat nastalih tem pri procesu tematske analize in seznam dokumentov uvožene zbirke podatkov je prikazan z vizualizacijama. V diplomskem delu smo aplikacijo izboljšali tako, da smo jo nadgradili z novimi funkcionalnostmi in vizualizacijami:

- vizualizacija dokumentov po času,
- izvoz korpusa podatkov,

- vizualizacija dokumentov po lokaciji nastanka,
- vizualizacija besedila dokumenta z metodo drevesnega prikaza.

1.5 Struktura dela

Delo je sestavljeno iz 4 poglavij. V 1. poglavju so predstavljeni področje in sorodna dela. Bralec spozna osnove vizualizacije in procesiranja naravnega jezika. 2. poglavje opisuje orodja in ogrodja, uporabljana pri razvoju aplikacije. V 3. poglavju sledi predstavitev razvoja in implemetacije vizualizacij. Delo se zakjuči s 4. poglavjem, ki vsebuje zaključek in opis možnosti nadaljnjega razvoja.

Poglavje 2

Orodja in ogrodja

Za izdelavo spletne aplikacije smo uporabili ogrodje Angular JS, označevalna jezika HTML in CSS ter programski jezik JavaScript. Aplikacija za shranjevanje podatkov uporablja podatkovno bazo ElasticSearch, ki omogoča hitro in preprosto delo z besedilnimi dokumenti. Glavno orodje, uporabljeno za tematsko analizo uvoženih podatkov, je bilo Mallet, ki omogoča preprosto uporabo s pomočjo ukazne vrstice. Za prikaz grafikonov vizualizacij smo uporabili JavaScript knjižico D3.js. V nadaljevanju so podrobneje predstavljena orodja in ogrodja, uporabljena pri razvoju spletne aplikacije.

2.1 AngularJS

AngularJS je aplikacijsko ogrodje za izdelavo dinamičnih spletnih aplikacij, zasnovano na programskem jeziku JavaScript. Ogrodje spremeni tradicionalni HTML (ang. *Hyper Text Markup Language*) v dinamičnega z avtomatsko sinhronizacijo modela in pogleda. Deluje tako, da prebere stran HTML, ki ima dodatne vgrajene atribute (ang. *tag attributes*). Prebrane atribute razlaga kot direktive za povezovanje vhodnih ali izhodnih delov strani z modelom, ki ga predstavljajo standardne spremenljivke JavaScript. Vrednosti teh spremenljivk se lahko ročno nastaviijo v kodi ali pridobijo iz statičnih ali dinamičnih virov JSON [3] [11].

2.2 HTML, CSS in JavaScript

HTML (ang. Hyper Text Markup Language) je označevalni jezik, namenjen izdelavi spletnih strani. Strukturo strani opišemo s pomočjo elementov HTML, ki so sestavljeni iz značk (ang. tags). Posamezna značka je običajno sestavljena iz začetka in konca, med katera dodamo poljubno besedilo. Brskalniki uporabijo značke za upodobitev vsebine spletne strani. Za določitev stila posameznemu elementu HTML smo uporabili jezik CSS (ang. Cascading Style Sheets). CSS nam omogoča, da posameznemu elementu HTML določimo oblikovne lastnosti, kot so na primer: barva, velikost in pozicija. Oblikovne lastnosti lahko zapisujemo v ločeno datoteko in jih tako ločimo od vsebine strani, kar poveča preglednost napisane kode.

S pomočjo jezika JavaScript uporabniku omogočimo, da se aktivno vključi v delovanje spletne strani z upravljanjem dinamičnih nalog spletne strani. JavaScript je objektno orientiran skriptni programski jezik, ki je na voljo vsakemu uporabniku brez uporabe licenc. Z njegovo pomočjo spletno stran spremenimo v interaktivno ter tako uporabniku omogočimo, da lahko ob pritisku na gumb odpre novo pogovorno okno. Uporaba jezika JavaScript je podprta v vseh novejših brskalniki [4].

Uporabili smo tudi nekaj dodatnih knjižnic JavaScript, kot sta jQuery [5], ki se uporablja za lažjo manipulacijo z jezikom JavaScript, in D3.js [1], ki skrbi za prikaz grafikonov.

2.3 ElasticSearch

ElasticSearch je odprtokodni sistem za iskanje in analiziranje besedila, napisan v programskem jeziku Java. Uporablja se predvsem v aplikacijah, ki za svoje delovanje uporabljajo kompleksne funkcije iskanja podatkov [2].

ElasticSearch se na več načinov razlikuje od klasične relacijske baze podatkov. Glavna razlika je, da v bazi namesto ključev hrani dokumente, ki so predstavljeni kot objekt JSON. Za poizvedbe ne uporablja klasičnega jezika SQL (Scripted Query Language), temveč lasten jezik, ki temelji na objektih

JSON [14].

Deluje kot samostojen strežnik, ki s pomočjo RESTful API omogoča branje, pisanje in posodabljanje podatkov v realnem času [2]. Za takojšen vpogled in testiranje pravilnosti podatkov v bazi ElasticSearch smo uporabili orodje Kibana, ki ponuja preprost grafični vmesnik za iskanje in vizualizacijo podatkov baze v realnem času [6].

2.4 Mallet

Mallet je knjižnica, ki omogoča obdelavo naravnega jezika, napisana v programskem jeziku Java. Vključuje številna orodja za obdelavo, med katerimi je za nas najpomembnejše orodje za modeliranje tem, ki se v naši aplikaciji uporablja pri obdelavi podatkov uvoženega korpusa. Razvito je bilo na Univerzi v Massachusettsu in je prosto dostopno brez uporabe licenc.

Pred uporabo je orodje treba namestiti na računalnik. Delo z orodjem poteko preka s pomočjo ukazov, ki jih vpisujemo v ukazno vrstico. Najpomembnejši in prvi ukaz, ki ga izvedemo, je *import*. Ta ukaz v Mallet uvozi korpus podatkov in ustvari posebno vrsto datoteke s končnico *mallet*, nad katero lahko nato izvajamo tematsko analizo. Tematsko analizo izvedemo z ukazom *train-topics*, ki mu dodamo ime datoteke, ki je nastala pri uvozu. Ukazu lahko dodamo tudi več parametrov, s katerimi prilagodimo modeliranje. Nastavimo lahko na primer število tem, ki jih želimo za uvožen korpus podatkov.

2.5 Bitbucket

Za verzioniranje in shranjevanje programske kode na splet smo uporabili spletno storitev Bitbucket, ki ponuja brezplačne zasebne repositorije, v katerih lahko hranimo programsko kodo in z njo povezano dokumentacijo.

Poglavje 3

Spletna aplikacija

Poglavje opisuje strukturo in razvoj spletne aplikacije. Spletna aplikacija je sestavljena iz več različnih vizualizacij, ki uporabniku posredujejo informacije izbrane zbirke podatkov. Zbirko podatkov je treba pred uporabo aplikacije uvoziti v bazo ElasticSearch. Uvoz je urejen na strani *Settings*, do katere uporabnik dostopa s pomočjo glavne navigacije aplikacije. Prehod med vizualizacijami je omogočen s pomočjo povezav leve navigacije.

3.1 Uvoz podatkov v bazo

Kot je bilo že omenjeno, je treba pred uporabo aplikacije v bazo ElasticSearch uvozi podatke zbirke dokumentov. Zbirko je treba pred uvozom pretvoriti v format JSON. Primer podatkov, pripravljenih za uvoz, je prikazan na sliki 3.1.

Uporabnik podatke uvozi na strani *Settings*, in sicer tako, da pritisne na gumb *New*. Ob pritisku na gumb *New* se odpre pogovorno okno za uvoz novega korpusa. Uporabnik v pogovornem izbere datoteko JSON za uvoz ter določi naziv in jezik uvoženega korpusa. Preden uporabnik potrdi uvoz s pritiskom na gumb *ok*, lahko nastavi tudi nekatere parametre, ki se upoštevajo pri procesu tematske analize uvoženega korpusa z orodjem Mallet. V primeru, da uporabnik teh parametrov ne nastavi, se pri analizi uporabi privzeta

```
[{
    "Title": "1. PEGAM IN LAMBERGAR (DVOBOJ JUNAKA Z VELIKANOM) / 1.",
    "Publication": "SLP 1: PRIPOVEDNE PESMI",
    "Author": "Ljudska pesem",
    "Location": "Gorenjsko",
    "LocationCoords": [
      125926.0000,
      447217.0000,
      46.2753697598736,
      14.3103248139116
    ],
    "Region": "Gorenjsko",
    "Text": "Pegam dirja po Dunaju,\nda ogenj izpod kopit škropi.",
    "Lemma": "pegam dirjati po dunaj \n da ogenj izpod kopito škropiti",
    "Year": 1775,
    "OtherInfo": "Kraj: Gorenjsko (?)"
}]
```


Slika 3.1: Primer podatkov, pripravljenih za uvoz.

vrednost parametrov. Pomen in privzete vrednosti parametrov so podrobno opisani v tabeli 3.1.

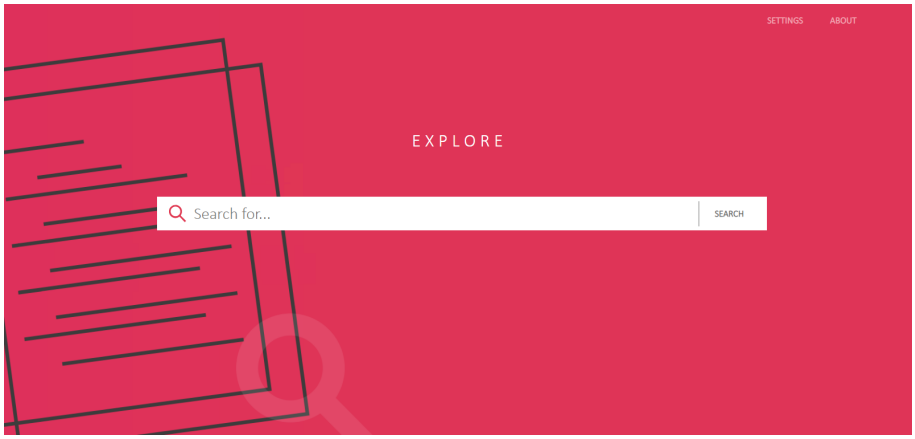
Po potrditvi uvoza aplikacija preveri, ali so podatki v pravilnem formatu JSON in ali korpus z izbranim imenom že obstaja v bazi. V primeru pravilnega formata in prostega imena se v bazo ElasticSearch uvozijo podatki korpusa. Vsak dokument v korpusu je svoj zapis v bazi in je s korpusom povezan z identifikatorjem. Nato se za korpus izvede tematska analiza z orodjem Mallet, na podlagi izbranih parametrov. Po končanem uvozu in analizi je korpus viden na seznam uvoženih korpusov, na strani *Settings*. Uporabnik lahko z izbiro korpusa in pritiskom na gumb *Update* izvede novo tematsko analizo že uvoženega korpusa. Izvede se enak postopek analize kot pri uvozu novega korpusa, na volju pa so tudi enaki parametri. Uporabniku sta na volju tudi gumba *Delete* in *Export*; s prvim izbriše korpus iz baze ElasticSearch, z drugim pa izvozi podatke korpusa. Pri izvozu korpusa iz baze ElasticSearch se podatki izbranega korpusa pretvorijo v format JSON in shranijo na izbrano lokacijo uporabnikovega sistema.

Tabela 3.1: Parametri tematske analize.

Parameter	**Opis**	**Privzeta vrednost**
število tem	Število tem, ki jih vrne analiza	1
število besed	Število besed za vsako temo	20
število dokumentov	Število dokumentov, ki opišejo temo	100
odstotek ujemanja dokumenta	Odstotek, nad katerim lahko govorimo o ujemanju dokumenta s temo	0
število ponovitev	Kolikokrat se ponovi proces tematske analize, preden dobimo teme	1000

3.2 Uvodna stran

Uporabnik spletne aplikacije je ob prvem obisku preusmerjen na uvodno stran. Na njej se nahajata besedilno polje za vpis iskalnega niza in gumb *search*. Ob pritisku na gumb sprožimo iskanje dokumentov za vpisan niz v bazi ElasticSearch. Aplikacija uporabnika preusmeri na stran *Documents*, kjer so prikazani rezultati iskanja. V glavi strani je na voljo povezava na stran *Settings*, na kateri uvozimo podatke v bazo ElasticSearch. Omenjene lastnosti uvodne strani prikazuje slika 3.2.

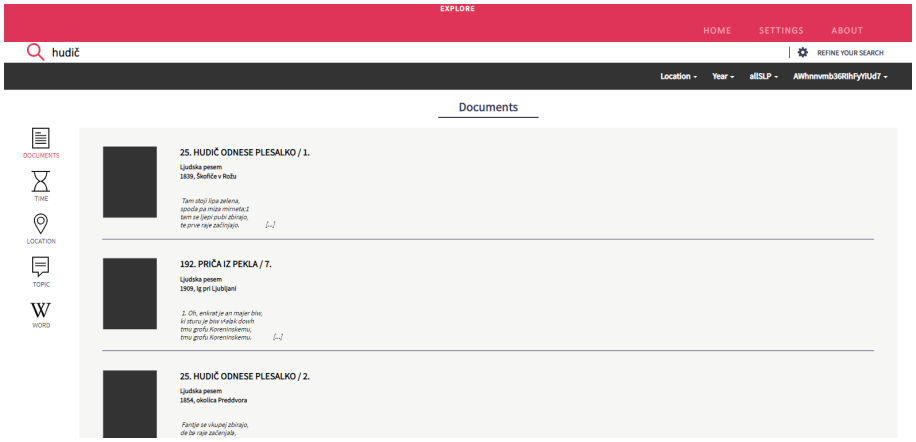


Slika 3.2: Uvodna stran spletne aplikacije.

3.3 Seznam dokumentov

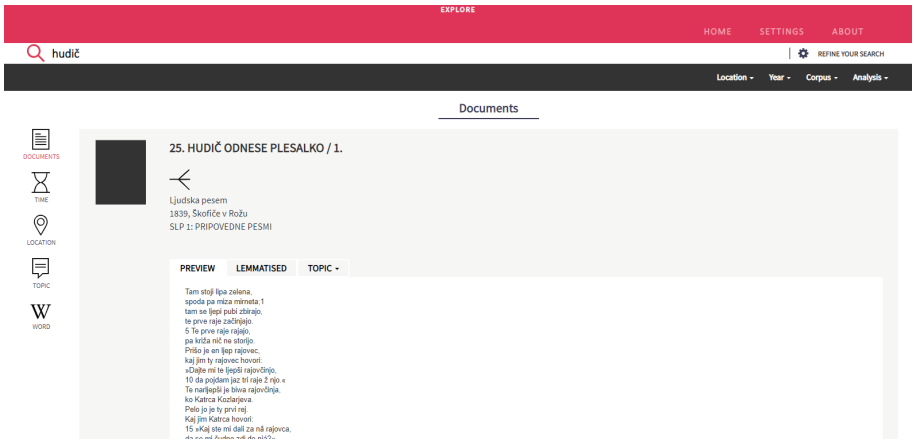
Stran prikazuje seznam dokumentov za vpisan iskalni niz, ki se lahko nahaja v naslovu dokumenta, imenu avtorja, besedilu dokumeta ali lematizaciji dokumenta. Dokumenti so razvrščeni tako, da so na vrhu seznama dokumenti, ki iskalni niz vsebujejo v naslovu. Privzeto se iskanje dokumentov izvede za vse uvožene korpuse podatkov. Iskanje lahko omejimo s filtrom, ki se nahaja pod glavno navigacijo. Filter je sestavljen iz spustnih seznamov, s katerimi lahko nastavimo: korpus, tematsko analizo, lokacijo in leto. Ob izbiri filtra se izvede ponovno iskanje dokumentov, pri čemer se upošteva izbrani filter. Na vrhu strani je besedilno polje, v katerem vidimo trenutni iskalni niz. V polje lahko vpišemo nov niz in z gumbom *REFINE YOUR SEARCH* sprožimo novo iskanje.

Ob pritisku na ime ali sliko dokumenta se uporabniku odpre stran s podrobnostmi dokumenta. Stran uporabniku omogoča, da izbrani dokument podrobneje razišče. Na vrhu strani so navedeni osnovi podatki za izbrani dokument: naslov, avtor, lokacija in publikacija. Pod osnovnimi podatki se nahajo trije zavihku: *PREVIEW, LEMMATISED* in *TOPIC*. V zavihku *PREVIEW* je prikazano celotno besedilo dokumenta. Lematizacija dokumenta je prikazana v zavihku *LEMMATISED*. Zavihek *TOPIC* prikazuje



Slika 3.3: Seznam dokumentov.

seznam tematskih analiz korpusa dokumenta. Ob izbiri analize je uporabnik preusmerjen na stran *TOPIC*, ki je podrobno opisana v poglavju 3.10.



Slika 3.4: Podrobnosti dokumenta.

Uporabnik lahko s pritiskom na gumb pod naslovom dokumenta podrobneje pregleda strukturo dokumenta. Ob kliku na gumb se za izbrani dokument prikaže vizualizacija besedila dokumenta v obliki drevesa (ang. Word tree). S takšnim načinom vizualizacije prikažemo kontekste, v katerih se nahaja posemezna beseda besedilnega dokumenta. Je grafična različica tra-

Slika 3.5: Vizualizacija besedila s pomočjo metode word tree.

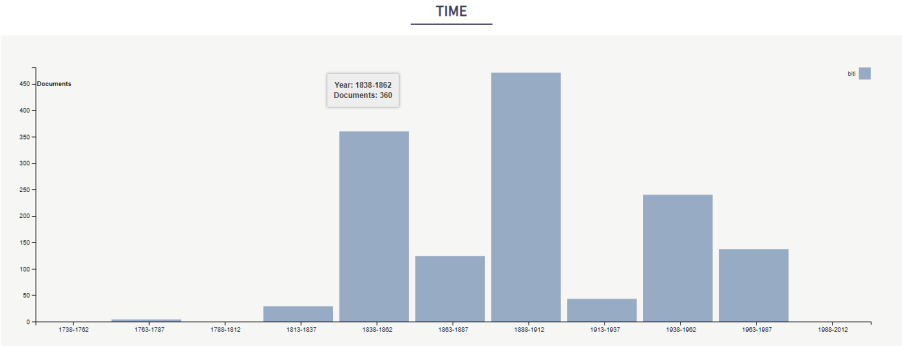
dicionalne metode *ključnih besed med besedilom* (ang. Key Word in Context). Omogoča hitro poizvedovanje in raziskovanje besedila ter preprosto prepoznavanje ponavljanja besednih zvez [23]. Prikaz vizualizacije je razdeljen na dva dela; na levi strani so prikazani izbrana beseda in konteksti, v katerih se pojavlja, na desni strani pa je prikazano celotno besedilo dokumenta. Novo besedo izbremo tako, da jo vpišemo v iskalno polje na vrhu strani ali pa jo izberemo iz prikaza celotnega besedila na desni strani. Uporabnik ima na voljo tudi gumb *reverse tree*, s katerim spremeni iskanje kontekstov besede. Namesto da za besedo prikažemo vse kontekste, kjer se beseda nahaja na začetku, prikažemo tiste, kjer se beseda nahaja na koncu. Za prikaz drevesa smo uporabili knjižnico Word tree [12].

3.4 Vizualizacija po letu nastanka

Ob izbiri povezave *Time* v levi navigaciji se prikaže vizualizacija dokumentov po obdobju nastanka. Namen vizualizacije je prikazati, kako so dokumenti za izbrani iskalni niz in korpus razdeljeni po obdobjih. Posamezno obdobje je prikazano kot stolpec na grafikonu, kjer je višina stolpca odvisna od količine dokumentov, narejenih v tem odbobju. Za izris grafikona smo uporabili

knjižnico JavaScript D3.js [1]. Prvotno smo razdelitev dokumentov želeli prikazati po letih, in ne po obdobjih. Ker imajo lahko dokumenti korpusa, najdeni za iskalni niz, več sto različnih let nastanka, bi nastalo preveč stolpcev in grafikon bi poslal nepregleden, zato smo se odločili, da bomo leta združili v obdobja. Odločili smo se, da je lahko zaradi preglednosti diagrama največ 20 obdobij. Leta združimo v obdobja tako, da izmed vseh najdenih dokumentov za vpisan iskalni niz poiščemo najmanjše in največje leto. Razliko med njima, razdelimo na največ 20 enakih obdobij.

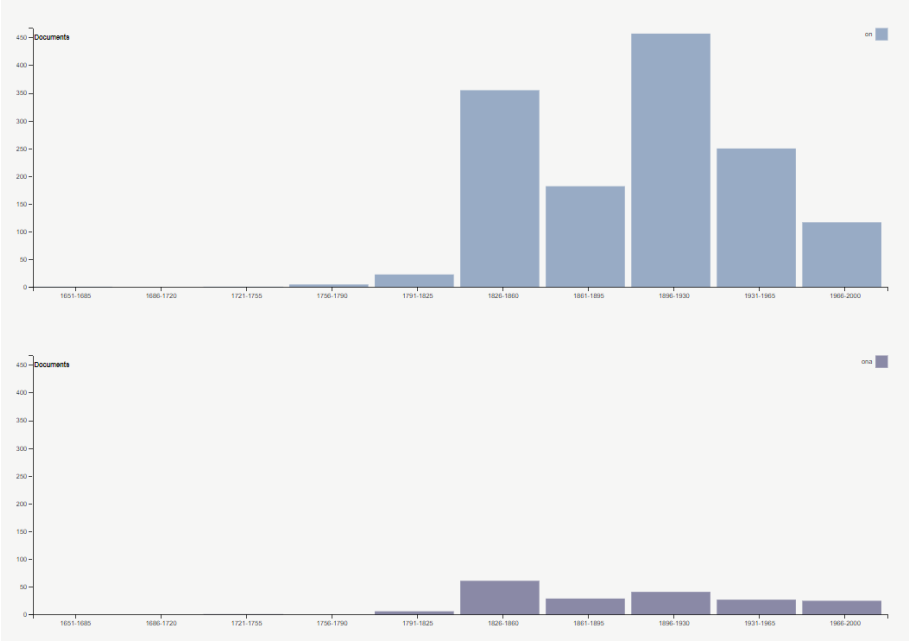
S *filrom*, ki se nahaja pod glavno navigacijo, lahko spremenimo izbrani korpus. Ob spremembi korpusa ponovno poiščemo dokumente, ki vsebujejo iskalni niz. Pri tem se ponovno izriše tudi vizualizacija.



Slika 3.6: Porazdelitev dokumentov po obdobjih.

Ob naslonitvi miškinega kazalca na določen stolpec se poleg stolpca prikaže prostor, v katerem so zapisane podrobnosti obdboja: leta obdobja in število dokumentov, nastalih v tem obdobju. S pritiskom na stolpec se pod grafikonom prikaže seznam dokumentov, ki so nastali v izbranem obdobju.

Vizualizacija omogoča tudi primerjavo razdelitve dokumentov po obdobjih za več iskalnih nizov. Uporabnik v iskalno polje vnese več nizov, ki jih loči z vejico. Pri razdelitvi let v obdobja se uporabita najmanjše in največje leto med vsemi najdenimi dokumenti. Za vsak niz se izriše svoj grafikon, kar omogoča preprosto primerjavo delitve dokumentov po obdobjih med nizi.



Slika 3.7: Primerjava porazdelitve dokumentov po obdobjih za dva iskalna niza.

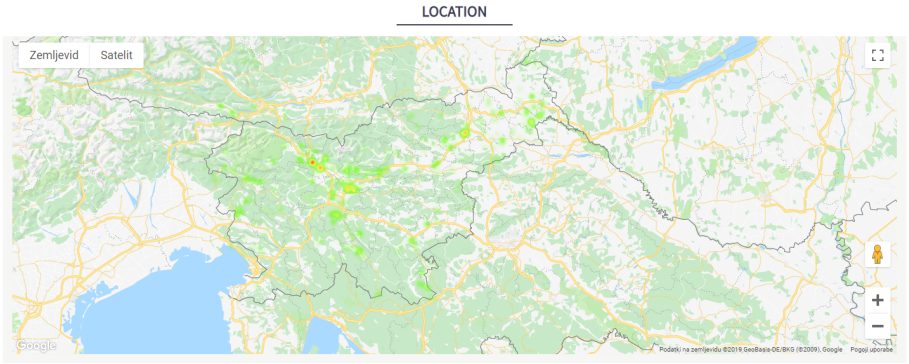
3.5 Vizualizacija dokumentov po lokaciji nastanka

Ob izbiri povezave *Location* v levi navigaciji se prikaže vizualizacija dokumentov po kraju nastanka in izbranem korpusu. Vizualizacija omogoča uporabniku, da z zemljevida razbere lokacije, kjer je nastalo največ oziroma najmanj dokumentov.

Za prikaz lokacij na zemljevidu smo se odločili, ker so lahko dokumenti enega korpusa iz več ali samo ene države oziroma kraja. Prikaz zemljevida je urejen s programsko knjižico Google Maps API, ki omogoča uporabo Google zemljevida (ang. Google Maps). Vsak dokument na zemljevidu prikažemo kot grafično podobo, za kar smo uporabili funkcijo toplotnega zemljevida (ang. heatmap). Ta nam omogoča, da lokacijo vsakega dokumenta prikažemo kot barvno piko, pri čemer je lokacija pike odvisna od zemljepisne dolžine in

širine kraja nastanka dokumenta. Barva pike je odvisna od števila dokumentov, nastalih na lokaciji. V primeru, da je na neki lokaciji nastal samo en dokument, je pika svetlo zelene barve, medtem ko so območja z velikim številom dokumentov odarvana rdeče. Tako so že na prvi pogled razvidne lokacije, kjer je nastalo največ dokumentov. Dokumenti, ki nimajo podatka o zemljepisni dolžini in širini kraja nastanka, so izvzeti iz prikaza. Primer prikaza lokacij nastanka dokumentov je viden na sliki 3.8.

Ob pritisku na določeno območje na zemljevidu se pod zemljevidom prikaže seznam dokumentov, ki so nastali v okolici izbranega območja. Seznam dokumentov se ustvari tako, da se za izbrano točko na zemljevidu določita zemljepisna dolžina in zemljepisna širina. Na podlagi zemljepisne dolžine in širine se nato izračuna razdalja do lokacije vsakega dokumenta. Seznam dokumentov prikazuje samo prvih 10 dokumentov, ki imajo kraj nastanka najbližje lokaciji pritiska na zemljevid. Dokumenti so razvrščeni tako, da so na vrhu seznama dokumenti z najkrajšo razdaljo do lokacije.



Slika 3.8: Prikaz lokacij nastanka dokumentov.

3.6 Vizualizacija tem korpusa

Do vizualizacije dostopamo s povezavo *Topic* v levi navigaciji. Za izbrani korpus in tematsko analizo v *filtru* se prikaže seznam tem. Vsaka vrstica v seznamu predstavlja eno temo, ustvarjeno pri procesu tematske analize. Za vsako temo je prikazenih največ 20 besed, ki podrobneje opisujejo temo ter odsotek ujemanja teme s korpusom. Število tem in besed je odvisno od izbranega števila pri procesu tematske analize. Proces tematske analize za ime teme določi naključen niz črk in številk, zato smo dodali možnost urejanja imena teme. Ime teme uredimo s pritiskom na gumb, ki se nahaja desno od imena teme. Pri tem se izvede poizvedba, ki posodobi ime teme v bazi ElasticSearch.

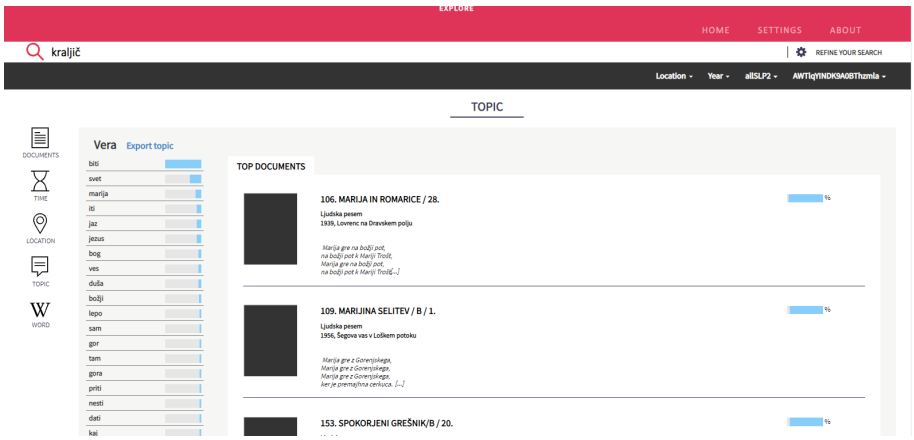
Slika 3.9: Seznam tem za izbrano analizo korpusa.

Ob pritisku na vrstico teme se odpre nova stran prikazana na sliki 3.10, ki prikaže podobnejšo analizo izbrane teme. Na levem delu strani se uporabniku za izbrano temo prikaže seznam besed, ki temo najbolje opišejo. Poleg vsake besede je prikazan odstotek ujemanja besede s temo. Desni del strani prikazuje seznam dokumentov za izbrano temo.

Uporabnik lahko celotno analizo teme, skupaj s seznamom dokumentov, izvozi v formatu JSON. To stori tako, da pritisne na gumb *Export topic*, ki se nahaja v levem zgornjem kotu strani.

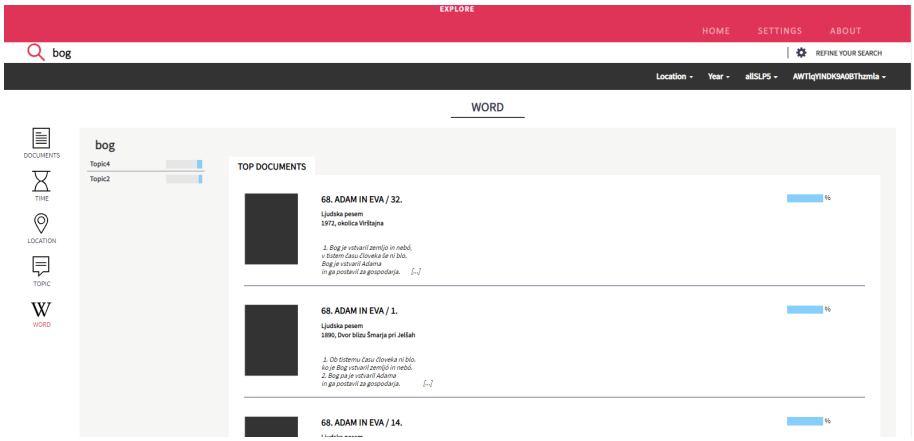
3.7 Vizualizacija besed korpusa

Zadnja povezava v levi navigaciji uporabniku omogoča analizo vpisanega niza. Uporabniku se prikaže seznam tem, v katerih se nehaja vpisani niz. Za prikaz seznama tem mora biti vpisan niz beseda, ki se nahaja v vsaj eni temi,



Slika 3.10: Podrobnosti teme.

ki jo dobimo pri tematski analizi. Poleg vsake teme je prikazan odsotek, ki nam pove, koliko se beseda ujemana z neko temo. Desni del strani prikazuje seznam dokumentov, v katerih se nahaja vpisana beseda. Ob pritisku na ime dokumenta je uporabnik preusmerjen na stran, ki prikazuje podrobnosti dokumenta.



Slika 3.11: Podrobnosti besede.

Poglavje 4

Zaključek

V okviru diplomskem dela smo predstavili področja vizualizacije besedilnih zbirk in procesiranja naravnega jezika. Predstavili smo razvoj spletne aplikacije ter orodja in ogrodja, ki smo jih pri tem uporabili. Uspešno smo implementirali zastavljene vizualizacije in tako izpolnili osnovni cilj. Pri tem smo za vsako vizualizacijo najprej poiskali primeren prikaz glede na tip podatkov ter ga nato implementirali. Držali smo se pravila, da so vizualizacije med seboj neodvisne, kar omogoča lažjo nadgradnjo oziroma morebitne spremembe v prihodnosti. Celoten sistem aplikacije je bil vzpostavljen v lokalnem okolju in bi ga bilo treba pred prehodom v produkcijo prestaviti na storitev, ki omogoča gostovanje spletnih aplikacij.

Aplikacija pušča še veliko prostora za nadgradnjo v prihodnosti. Smiselno bi bilo dodati možnost deljenja podatkov med več uporabniki aplikacije. Prav tako bi lahko aplikacijo povezali z orodjem, ki omogoča lematizacijo besedil; zdaj to uporabnik stori pred uvozom podatkov v aplikacijo.

Literatura

[1] Data-driven documents. Dosegljivo: https://d3js.org/. [Dostopano: 12. 1. 2019].

[2] Elasticsearch. Dosegljivo: https://www.elastic.co/products/elasticsearch. [Dostopano: 20. 1. 2019].

[3] Introducing json. Dosegljivo: https://www.json.org/. [Dostopano: 12. 1. 2019].

[4] An introduction to javascript. Dosegljivo: https://javascript.info/intro. [Dostopano: 12. 1. 2019].

[5] jquery. Dosegljivo: https://jquery.com/. [Dostopano: 12. 1. 2019].

[6] Kibana. Dosegljivo: https://www.elastic.co/products/kibana. [Dostopano: 20. 1. 2019].

[7] Ngram viewer. Dosegljivo: https://books.google.com/ngrams. [Dostopano: 31. 1. 2019].

[8] Structured vs. unstructured data. Dosegljivo: https://www.datamation.com/big-data/structured-vs-unstructured-data.html. [Dostopano: 20. 1. 2019].

[9] Tag cloud. Dosegljivo: http://edutechwiki.unige.ch/en/Tag_cloud. [Dostopano: 31. 1. 2019].

[10] Topic modeling. Dosegljivo: https://www.tidytextmining.com/topicmodeling.html. [Dostopano: 17. 1. 2019].

[11] What is angularjs? Dosegljivo: https://docs.angularjs.org/guide/introduction. [Dostopano: 12. 1. 2019].

[12] Word tree. Dosegljivo: https://www.jasondavies.com/wordtree/. [Dostopano: 20. 1. 2019].

[13] Big data visualization principles. Dosegljivo: https://itsvit.com/blog/big-data-visualization-principles/, 2017. [Dostopano: 16. 1. 2019].

[14] Berkay Akdal, Zehra Gül Çabuk Keskin, Erdem Eser Ekinci, and Geylani Kardas. Model-driven Query Generation for Elasticsearch. In *2018 Federated Conference on Computer Science and Information Systems*, page 853–862, 2018.

[15] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[16] Nan Cao and Weiwei Cui. *Overview of Text Visualization Techniques*, pages 11–40. 01 2016.

[17] Data visualization and infographics. Dosegljivo: https://www.smashingmagazine.com/2008/01/monday-inspiration-data-visualization-and-infographics/. [Dostopano: 15. 1. 2019].

[18] M. Friendly. A brief history of data visualization. In C. Chen, W. Härdle, and A Unwin, editors, *Handbook of Computational Statistics: Data Visualization*, volume III, pages 1–25. Springer-Verlag, Heidelberg, 2006. (In press).

[19] Matthew Hoffman, Francis R. Bach, and David M. Blei. Online learning for latent dirichlet allocation. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances*

in Neural Information Processing Systems 23, pages 856–864. Curran Associates, Inc., 2010.

[20] Susan Li. Topic modeling and latent dirichlet allocation (lda) in python. Dosegljivo: `https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24`, 2018. [Dostopano: 16. 1. 2019].

[21] George Seif. An easy introduction to natural language processing. Dosegljivo: `https://towardsdatascience.com/an-easy-introduction-to-natural-language-processing-b1e2801291c1/`, 2019. [Dostopano: 16. 1. 2019].

[22] Edward Tufte. The visual display of quantitative information. *Graphics Press*, 1983.

[23] Martin Wattenberg and Fernanda B. Viégas. The word tree, an interactive visual concordance. *IEEE Transactions on Visualization and Computer Graphics*, 14, 2008.