Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Miha Debenjak

Sledenje razvoju raziskovalnih tematik

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Tomaž Curk

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Raziskovalci svoje dosežke redno objavljajo v obliki znanstvenih člankov. Razumevanje tematik, ki jih raziskujejo in o katerih poročajo, je pomembno za razumevanje razvoja znanstvenega področja. V ta namen preučite metode za rudarjenje besedil in razumevanje naravnega jezika ter razpoložljive bibliografske vire. Razvite sistem, ki bo olajšal sledenje razvoju raziskovalnih tematik. Razviti sistem uporabite na praktičnem primeru skupine raziskovalcev.

Zahvaljujem se mentorju doc. dr. Tomažu Curku za vso pomoč in motivacijo pri izvedbi diplomske naloge.

Zahvala gre tudi družini in prijateljem, ki so mi pomagali in me spodbujali v času študija. Hvala tudi Martinu za pomoč pri prevajanju.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
LDA	latent Dirichlet alocation	latentna Dirichletova alokacija
SICRIS	Slovenian Current Research Information System	Informacijski sistem o raziskovalni dejavnosti v Sloveniji
HTML	Hypertext markup language	Jezik za označevanje nadbesedila
XML	Extensible markup language	Razširljivi označevalni jezik
JSON	JavaScript Object Notation	Objektni zapis JavaScript
URL	Uniform, resource locator	Enolični krajevnik vira
API	Application programming interface	Aplikacijski programski vmesnik
SSH	Secure Shell	protokol Secure Shell

Povzetek

Naslov: Sledenje razvoju raziskovalnih tematik

Avtor: Miha Debenjak

Spremljanje raziskovalnih tematik znanstvenih objav skupine raziskovalcev je zanimivo in hkrati pomembno za razumevanje razvoja nekega znanstvenega področja in raziskovalcev, ki delujejo na področju. Raziskovalci se ukvarjajo z različnimi področji, zato se tudi teme, o katerih pišejo v znanstvenih objavah, razlikujejo. Na podlagi besed, ki so uporabljene v znanstvenih člankih, lahko določimo teme, o katerih raziskovalci razpravljajo. V diplomski nalogi je opisano pridobivanje podatkov o člankih, njihova analiza in modeliranje tem člankov. Izvedena je bila tudi analiza zastopanosti različnih tem skozi čas, kar nam pove o aktualnosti tem v določenem času. Zgrajen sistem smo uporabili za analizo publikacij Fakultete za računalništvo in informatiko Univerze v Ljubljani.

Ključne besede: modeliranje tem, model LDA, rudarjenje besedil, vizualizacija, razvoj tematik, obdelava naravnega jezika.

Abstract

Title: Tracking research topics

Author: Miha Debenjak

Following the research topics of the scientific publications of a group of researchers is interesting and at the same time important for understanding the development of a scientific field and researchers working in it. The researchers do not work in the same field, therefore the topics of their work differ. Topics of articles can be identified on the basis of words used. The thesis describes the acquisition of data on articles, their analysis and modelling of topics that they discuss. In addition, an analysis was conducted on the representation of different topics over time, which shows most frequently discussed topics in certain time periods. This system was used for the analysis of publications of the Faculty of Computer and Information Science at the University of Ljubljana.

Keywords: topic modeling, LDA model, text mining, visualization, topic development, natural language processing.

Poglavje 1

Uvod

Raziskovalci objavljajo znanstvene in strokovne dosežke v številnih domačih in tujih znanstvenih revijah. Povezave do velike večine člankov, ki so delo slovenskih raziskovalcev, najdemo na spletnem portalu SICRIS. Portal ponuja podatke o raziskovalcih, njihovi raziskovalni dejavnosti in uspešnosti. Večina člankov je povezanih z drugima dvema spletnima portaloma *Web of Science* ali *Scopus*, ki ponujata podrobnejše podatke o članku, med drugim tudi povzetek v angleščini. Na podlagi povzetkov lahko hitro ugotovimo, s katerimi temami se ukvarjajo posamezni raziskovalci.

Na področju spremljanja aktualnih raziskovalnih tematik že obstajajo storitve, ki omogočajo vpogled v raziskovalno dejavnost organizacij in njihovih zaposlenih. To so, na primer, *ResearchGate*, *Google Scholar* in slovenski SICRIS. Na omenjenih spletnih straneh najdemo podatke o raziskovalnih ustanovah, avtorjih, člankih in indeksih citiranosti. Omogočajo nam hitro iskanje člankov, ki jih potrebujemo. Pri nekaterih člankih ponujajo tudi povezave do celotnega besedila članka. Omenjene strani pa ne omogočajo enostavnega sledenja trendom raziskovanja in posledično objavljanja člankov.

Zanimivo bi bilo videti, kako se raziskovalne tematike spreminjajo v času. Z različnimi grafičnimi prikazi bomo predstavili zastopanost tematik v različnih časovnih obdobjih. Prikazali bomo, katere teme so bile v določenem času aktualne. V diplomskem delu smo se osredotočili na analizo razisko-

valnih tematik objav raziskovalcev Fakultete za računalništvo in informatiko Univerze v Ljubljani (UL FRI), objavljenih v obdobju 1990-2019.

Poglavje 2

Pridobivanje in obdelava podatkov

2.1 Uporabljena orodja in tehnologije

BibMine

V sklopu projekta Turizem 4.0 je nastal program *BibMine* [14], ki sta ga razvila Ajda Pretnar in Tomaž Curk. Gre za sistem, ki iz portala *Scopus* pridobiva podatke o člankih na temo turizma in jih analizira. Vključuje procesiranje besedil, modeliranje tem in vizualizacijo dobljenih rezultatov. Pri našem delu smo izhajali iz *BibMine*, saj je veliko korakov v procesu precej podobnih.

Jupyter Notebook

Okolje *Jupyter Notebook* [7] je interaktivno programersko orodje, ki omogoča izvajanje kode v živo, uporabo interaktivnih vizualizacij, enačb, slik, navadnega besedila in drugih priročnih orodij. Med kodo lahko dodajamo besedilo tipa *Markdown*, kar zelo pripomore k preglednosti skript in rezultatov. Koda se poganja v jedrih *Jupyter*. Le-ta omogočajo različne programske jezike, med drugim tudi programski jezik Python.

Predlagani sistem za spremljanje raziskovalnih tematik smo sestavili iz večih *Jupyter Notebook*, ki so povezani med sabo. Ker mora izvajanje kode potekati na fakultetnem omrežju, se *Jupyter Notebook* poganjajo iz oddaljenega strežnika. S pomočjo protokola *SSH* se povežemo na oddaljeni strežnik, kjer urejamo in poganjamo kodo. Da bi imeli celotno izkušnjo, ki jo ponuja *Jupyter Notebook* in kodo urejali in poganjali iz svojega računalnika, moramo izvesti naslednje korake:

- povežemo se na oddaljeni strežnik s protokolom *SSH*
- na strežniku poženemo ukaz:

  ```
  jupyter notebook --no-browser --port=portNumber
  ```

- na svojem računalniku poženemo ukaz:

  ```
  ssh -N -L portNumber:localhost:portNumber
  <remote_user>@<remote_host>
  ```

- v brskalnik vpišemo http://localhost:portNumber/

Na ta način lahko upravljamo *Jupyter Notebook* na svojem računalniku, izvajajo pa se na oddaljenem strežniku, kjer se tudi shranjujejo vse spremembe in rezultati.

2.2 Viri bibliografskih podatkov

SICRIS

SICRIS [4] je informacijski sistem o raziskovalni dejavnosti v Sloveniji. Trenutno vsebuje podatke o več kot 2500 raziskovalnih ustanovah in skupinah, med njimi tudi podatke o UL FRI. Iz sistema smo dobili podatke o objavljenih člankih zaposlenih na fakulteti. Na podlagi besedil smo skušali določiti teme, s katerimi se ukvarjajo zaposleni na fakulteti, ter ugotoviti, kako se trendi spreminjajo v različnih časovnih obdobjih.

Scopus

Scopus [3] je zbirka člankov, njihovih podatkov, osnutkov in indeksov citiranosti. Zbirka je del nizozemskega podjetja *Elsevier*, ki je eno večjih podjetji na področju znanstvenega založništva. Do podatkov dostopamo s klici API. Za uporabo vmesnika API se moramo registrirati na spletni strani *Elsevier* in tam pridobiti ključ za API. Klice API moramo izvajati iz fakultetnega omrežja, kjer imamo več pravic, da nam vmesnik API vrne tudi povzetek članka, ki je osnova za nadaljnje delo.

Web of Science

Tako kot na *Scopus* tudi na portalu *Web of Science* [5] najdemo bibliografske podatke o člankih in indekse citiranosti. Tudi za uporabo tega portala je potrebna prijava na fakultetnem omrežju. Najprej smo skušali podatke pridobivati s pomočjo klicev API, a naša fakulteta s trenutno pogodbo s Clarivate Analytics, lastniki portala, nima dostopa do vmesnika API. Zato smo se odločili, da bomo podatke pridobivali iz spleta. Njihov plačljivi vmesnik API omogoča hitrejše in zanesljivejše pridobivanje podatkov iz portala.

2.3 Pridobivanje podatkov

Postopek se začne na strani SICRIS, kjer lahko dostopamo do podatkov o objavljenih člankih določenega avtorja ali ustanove. S pomočjo šifre ustanove izvedemo poizvedbo, ki nam vrne vse zapise člankov te ustanove. V našem primeru, za Fakulteto za računalništvo in informatiko Univerze v Ljubljani (UL FRI), uporabimo šifro 1539. Poleg tega izberemo časovno obdobje, za katerega želimo pridobiti članke. Stran, ki nam jo vrne poizvedba, shranimo v obliki HTML in XML, da nam tega postopka ni treba ponavljati vsakič znova. Datoteke se shranjujejo v projektu, v mapi *downloads/cobiss-org*.

Sledi analiza shranjene datoteke XML. Iz nje izluščimo povezave do spletnih portalov *Scopus* in *Web of Science*. Iz teh portalov bomo kasneje namreč

pridobivali ostale podatke, ki so pomembni za analizo. Izbiramo lahko med dvema portaloma - *Scopus* in *Web of Science*. Ker se postopka za pridobivanje podatkov iz teh poratlov razlikujeta, bosta opisana vsak posebej.

Scopus. Najprej opišimo pridobivanje podatkov iz portala *Scopus*, saj je to bistveno lažje. Preko povezav na članke na *Scopus*, ki smo jih dobili iz shranjene XML datoteke pridobljene iz SICRIS, izvedemo API klic. S tem klicem prejmemo XML kodo s podatki o članku. Za uporabo vmesnika *Scopus* API se moramo najprej registrirati na *Elsevierjevi* spletni strani in tam zaprositi za ključ za API. *Elsevier* nam nudi veliko različnih vmesnikov API. Nas zanima samo API *Abstract Retrieval* in sicer različica *META-ABS*, ki vrača tudi povzetek članka. Klici API morajo biti izvajani iz fakultetnega omrežja, saj je edina različica, ki je dostopna zunaj fakultetnega omrežja *BASIC*, ki pa ne vrača povzetka in je zato neuporabna. Kodo XML, ki jo klic vrne, shranimo v datoteko XML.

Za ***Web of Science*** je bil uporabljen drugačen pristop. Ker nimamo dostopa do vmesnikov API *Clarivate Analytics*, smo morali podatke pridobiti iz spleta. Ker funkcija *get* iz Pythonove knjižnice *Requests* ne zna pravilno obdelati preusmeritev, ki se dogajajo v ozadju klica te funkcije nad URL-jem, ki ga imamo shranjenega v podatkih iz SICRIS, smo morali uporabiti knjižnico *MechanicalSoup* [2]. Tako smo simulirali obisk spletne strani preko brskalnika in shranili kodo HTML obiskane spletne strani.

Iz pridobljenih datotek HTML in XML moramo nato izluščiti podatke, ki nas zanimajo. Tako lahko na portalu *Scopus* dobimo podatke o naslovu, avtorjih, ključnih besedah, datumu objave in citiranosti ter osnutek. Na strani *Web of Science* pa lahko poleg vseh teh dobimo še več podatkov, recimo o tem, kdo je financiral raziskavo, na podlagi katere je bil kasneje napisan članek. S pomočjo knjižnice *Langdetect* [1, 16] osnutku določimo jezik, v katerem je napisan. Pridobljene podatke shranimo v formatu JSON v direktorij *data*. S pomočjo imena ločimo datoteke, ki hranijo zapise o člankih iz *Scopus* od tistih, ki hranijo podatke člankov iz *Web of Science*.

Obdelovati želimo aktualne podatke. Zato je sistem narejen tako, da pre-

veri, če v projektu že obstaja direktorij za članek. V primeru, da ne obstaja, ustvari nov direktorij in vanj shrani HTML oziroma kodo XML strani. Ime dobi po indeksu članka na strani. Če direktorij že obstaja, pa sistem preveri, kdaj je bila zadnjič posodobljena datoteka s podatki. Datoteke, ki so stare več kot 14 dni, se ponovno shranijo. Tako zagotovimo, da imamo vedno aktualne podatke, hkrati pa da ne vedno znova shranjujemo kode HTML in XML ter tako potencialno preskočimo en korak v procesu pridobivanja podatkov.

2.4 Obdelava podatkov

Podatke iz datotek JSON uvozimo v podatkovni okvir *Pandas* [12]. Za nadaljnje delo je najpomembnejši atribut podatkovnega okvirja povzetek članka. Besedila povzetkov moramo pred začetkom gradnje modelov obdelati in pripraviti za nadaljnjo obdelavo.


```
{
   "Title":"Improving mobile operator information system efficiency through EAI",
   "Authors":[
      "Rozman I.",
      "Juric M.B.",
      "Hericko M.",
      "Vezocnik I.",
      "Krisper M."
   ],
   "Keywords":[
      "CORBA",
      "EAI",
      "Integration",
      "J2EE",
      "Methods",
      "Web services"
   ],
   "Date":"2004",
   "Abstract":"Ability to integrate legacy assets and reusing their functionality in modern
   "Language":"en",
   "Citations":"1"
}
```


Slika 2.1: Primer članka shranjenega v formatu JSON.

Za procesiranje naravnega jezika je bila uporabljena knjižnica *Natural Language Toolkit* (NLTK) [10]. Najprej iz podatkovnega okvirja odstranimo morebitne članke, ki so brez povzetkov. Nato iz besedil odstranimo fraze in stop besede. Besedilo tokeniziramo in njegovim besedam določimo iztočnice (ang. *lemmatization*) ter poiščemo pare besed (t.i., bigrame). Odstranimo

tudi besede, ki se prevečkrat oziroma premalokrat pojavijo v zbirki dokumentov. Povzetki lahko vsebujejo tudi za nas nepotrebne fraze, kot so na primer podatki o avtorskih pravicah, ki se pojavljajo na koncu povzetkov. Ker se podatki pridobivajo iz spletnih strani, se lahko zgodi, da povzetki vsebujejo tudi razne oznake HTML. Tudi te nam o vsebini povzetka ne povedo nič, zato jih odstranimo. Tako obdelano besedilo shranimo v novem stolpcu podatkovnega okvirja.

Poglavje 3

Modeliranje tem

Modeliranje tem je eno izmed področij strojnega učenja in obdelave naravnega jezika. V dani zbirki besedil skušamo odkriti teme, o katerih avtorji pišejo v besedilih. Tema je osnovni, osrednji predmet obravnavanja ali umetniškega dela. Če več tem, ki so si med seboj podobne, združimo skupaj, dobimo tematiko. Človek je sposoben, na podlagi prebranega besedila določiti, o kateri temi le-to govori. Težava nastane, ko moramo teme določiti obsežnim zbirkam besedil, kar bi človeku vzelo ogromno časa. Glavna ideja za strojno modeliranje tem je predpostavka, da besedila, ki govorijo o isti temi, vsebujejo podobne besede.

Razvitih je bilo več pristopov za modeliranje tem. Najbolj znane so *Probabilistic latent semantic analysis* (*PLSA*) [6], *Latent Dirichlet allocation* (*LDA*) [11], *Pachinko allocation* [8] in *Hierarchical latent tree analysis* (*HLTA*) [9]. V našem delu smo uporabili model *LDA*.

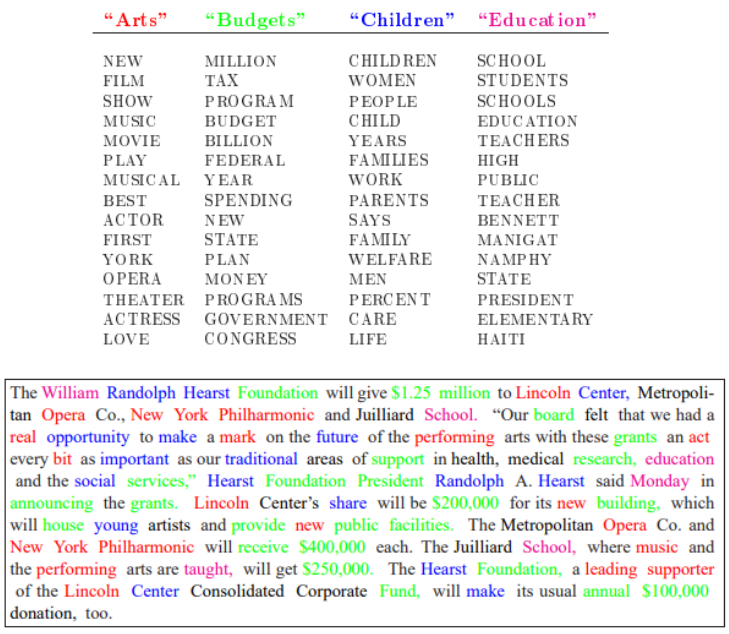
3.1 Model latentne Dirichletove alokacije

Latentna Dirichletova alokacija (*LDA*) je najbolj uporabljen pristop za modeliranje tem. Razvili so ga Blei in sod. leta 2002 [11]. Je posplošena verzija modela *PLSA*. Na podlagi vnaprej določenega števila tem in besed, ki so uporabljene v besedilih, oblikuje teme, ki najbolje opisujejo korpus besedil.

Teme so predstavljene z vektorji, ki so sestavljeni iz besed in njihovih uteži. Tako tema, ki jo dobimo, ni *'Zgradba'*, ampak *'hiša: 0.3, blok: 0.2, stolp: 0.1'*. Katero temo vektor predstavlja, moramo torej odkriti sami. Primer odkrite teme: *learning: 0.009, model: 0.008, present: 0.008, data: 0.007, used: 0.007, algorithm: 0.007, approach: 0.006, system: 0.006, machine: 0.005, rule: 0.005*. Podan je utežen seznam desetih besed, ki najbolje opišejo temo. Iz besed lahko sklepamo, da vektor predstavlja temo, ki opisuje strojno učenje.

Vsaka beseda predhodno procesiranega besedila dokumenta pripada eni izmed tem, ki jih je model odkril. Na podlagi besed, ki so uporabljene v dokumentu, lahko torej določimo, iz katerih tem je dokument sestavljen in kolikšno težo imajo v dokumentu. Na sliki 3.1 je prikazana sestava dokumenta iz odkritih tem. Vsaka barva prikazuje svojo temo. Črne besede so nepomembne za odkrite teme.

Uporabili smo implementacijo metode *LDA* v knjižnici *Gensim* [15].



Slika 3.1: Prikaz sestave dokumenta iz odkritih tem. Vir: [11].

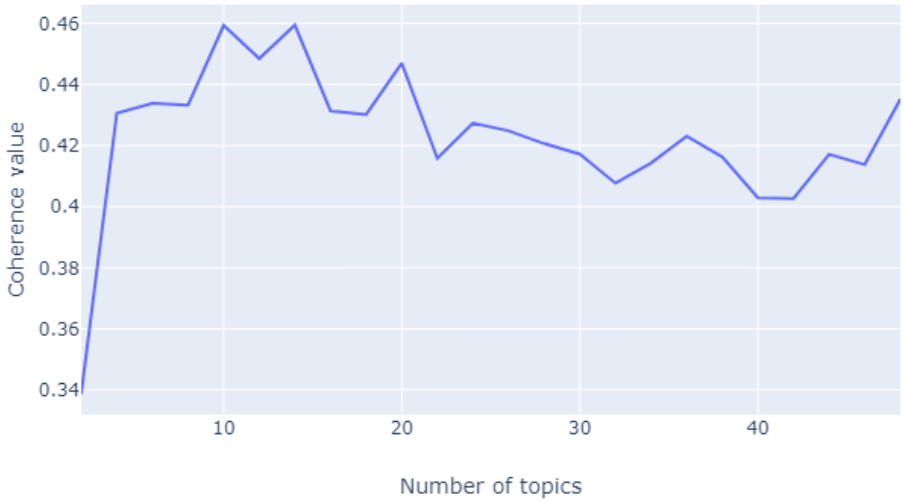
3.2 Določanje optimalnega števila tem

Število tem, ki jih vsebuje korpus besedil, nam je pred postopkom odkrivanja tem neznano. A vendar *LDA* od nas zahteva, da podamo pričakovano število tem, ki naj jih *LDA* oblikuje. Če kot argument podamo premajhno vrednost, nam bo model vrnil obsežne teme, v katerih bo zajetih več manjših tem, ki bi sicer morale biti ločene. Prevelika vrednost pa bo vračala fragmentirane, nerazumljive teme, ki bi najverjetneje morale biti združene. Zanima nas torej, kolikšno je najboljše število tem, ki naj jih model *LDA* poišče v korpusu besedil.

Da določimo optimalno število tem, zgradimo več modelov *LDA*, pri čemer vsak model odkriva različno število tem. Vsakemu modelu nato izračunamo vrednost koherence (ang. *Coherence values*) [13]. Razvitih je bilo več mer za koherenco. Mi smo uporabili mero C_v, ki jo izračunamo v štirih korakih [18]:

1. Segmentacija besed, ki predstavljajo vektorje tem v pare besed.
2. Izračun verjetnosti besed oziroma parov besed.
3. Izračun mere potrditve, ki pove kako močno množica besed podpira drugo množico besed.
4. Izračun koherence (povprečna vrednost mer potrditve).

Po izračunu vseh vrednosti koherence iz dobljenih vrednosti izberemo optimalno število tem. To določimo tako, da izberemo število, ki predstavlja vrh hitro naraščajočih vrednosti koherence [18]. Vrednosti lahko za tem številom še vedno naraščajo, a se potem poveča možnost, da se besede v temah začnejo ponavljati. Prvi vrh nam torej vrne obširnejše teme oziroma tematike, naslednji pa podrobnejše teme, ki so skrite znotraj teh tematik.



Slika 3.2: Koherenca za različne vrednosti pričakovanega števila tem.

Iz grafa na sliki 3.2 lahko razberemo, da je primerno število tem štiri ali deset, ki predstavljata vrha hitro naraščajočih delov grafa. Vrednost koherence pri manj kot štirih temah je prenizka. Pri več kot desetih temah pa začne vrednost koherence počasi padati, kar pomeni, da modeli niso več tako dobri. V korpusu, na katerem je bil pognan algoritem za iskanje optimalnega števila tem, imamo torej štiri tematike, ki se podrobneje delijo na deset tem.

Poglavje 4

Rezultati

Razvili smo sistem, ki za dano časovno obdobje pridobi članke in na izhodu vrne izrise, ki ponazarjajo podatke o tematikah in njihovem razvoju skozi čas. Vsa programska koda, izdelana v diplomski nalogi je prosto dostopna na GitHub[1].

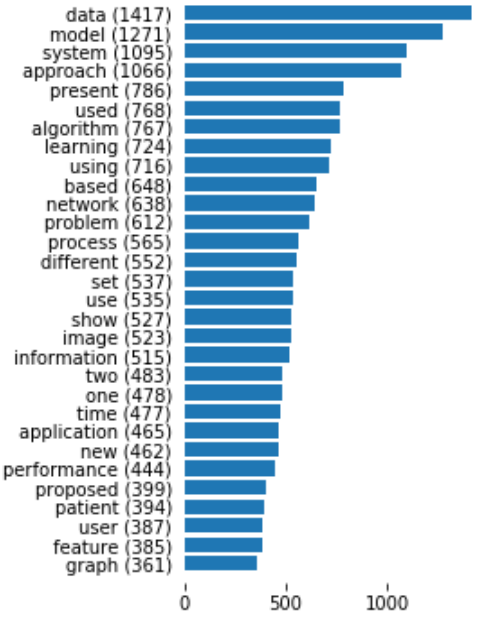
Ker je podatkov o objavah veliko in njihova količina neprestano raste, jih moramo znati primerno predstaviti. Vizualizacija podatkov je grafična predstavitev informacij in podatkov. Z uporabo tabel, grafov, zemljevidov in drugih načinov vizualizacije skušamo ljudem na intuitivno razumljiv način predstaviti podatke, vzorce, povezave med podatki in trende, ki so prisotni v podatkih. Orodja so učinkovita, ker ljudje lažje zaznavamo barve, vzorce ali oblike, kot pa številke ali besede. Tako hitreje razberemo in razumemo podatke.

Nekatere podatke lahko vizualiziramo na preprost način. Na primer, številske podatke prikažemo s krivuljami. Vizualizacija nekaterih drugih podatkov pa je bolj zahtevna. Kako bi na primer grafično prikazali teme in njihove odvisnosti? Tudi za to obstajajo tehnike. V diplomski nalogi smo uporabili vizualizacijo modela *LDA* s pomočjo knjižnice *pyLDAvis*.

[1] `https://github.com/mihad8/razvoj-raziskovalnih-tematik`

4.1 Osnovne vizualizacije

Uporabili smo nekaj osnovnih vizualizacij, ki nam pomagajo razumeti podatke, s katerimi imamo opravka. Najprej si ogledamo najbolj uporabljene besede v celotni zbirki člankov. V spodnjem grafu na sliki 4.1 vidimo, da so besede, ki se uporabljajo v pridobljenih člankih, na prvi pogled res primerne za članke, ki so bili objavljeni s strani raziskovalcev UL FRI.

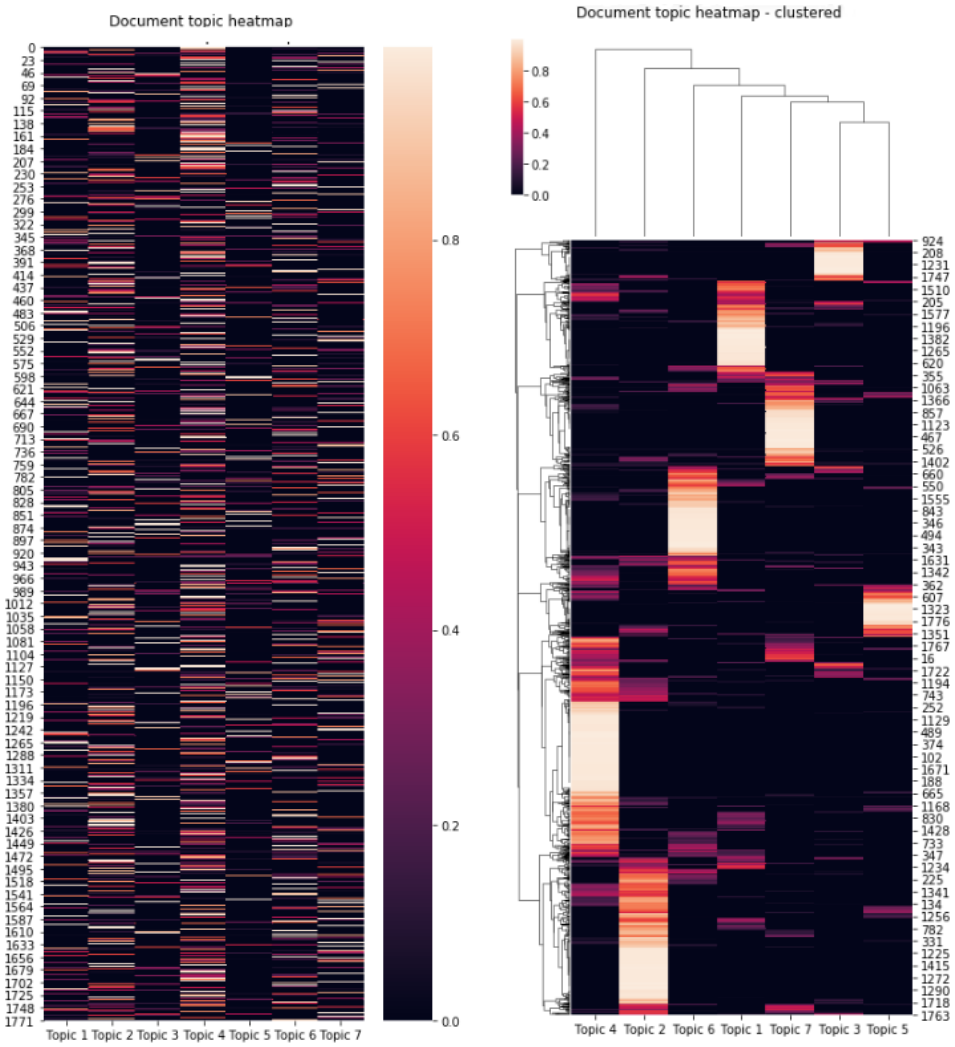


Slika 4.1: Najbolj pogoste besed v korpusu povzetkov člankov UL FRI.

Članki v podatkovnem okvirju seveda niso razvrščeni po temah, temveč po vrstnem redu po katerem so objavljeni na spletni strani SICRIS. Če oblikujemo toplotno karto (ang. *heatmap*), v katerem prikažemo utež tem na članek, pri čemer svetlejša barva pomeni večjo utež in temnejša manjšo, potem dobimo zelo nejasno sliko, saj so članki naključno razporejeni po podatkovnem okviru, ne glede na njihovo temo. Primer takšne toplotne karte je prikazan na sliki 4.2a.

Članke lahko gručimo glede na vsebovanost tem. Tako dobimo urejeno toplotno karto na sliki 4.2b, iz katere razberemo najbolj zastopane teme v

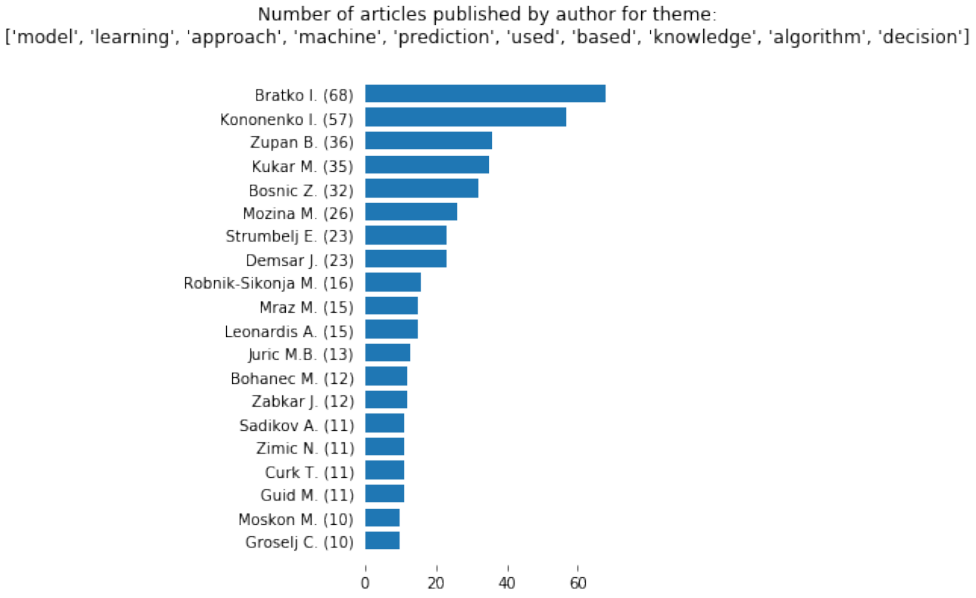
zbirki člankov. Vidimo tudi katera tema ima največ člankov, ki so posvečeni izrecno eni temi; vidni so kot področja svetlo rdeče in bele barve.



(a) (b)

Slika 4.2: Toplotne karte tem (a) v odvisnosti od časa in (b) gručenje člankov glede na prisotnost tem.

Ogledamo si lahko tudi, kateri avtorji pišejo o posameznih temah. V ta namen smo napisali funkcijo *get_topic_authors()*, ki za vsako odkrito temo vrne seznam avtorjev, in za vsakega avtorja število člankov s področja teme. Funkcija vrača le avtorje, ki so objavili vsaj deset člankov z določeno temo. Slika 4.3 prikazuje primer izpisa avtorjev izbrane teme.



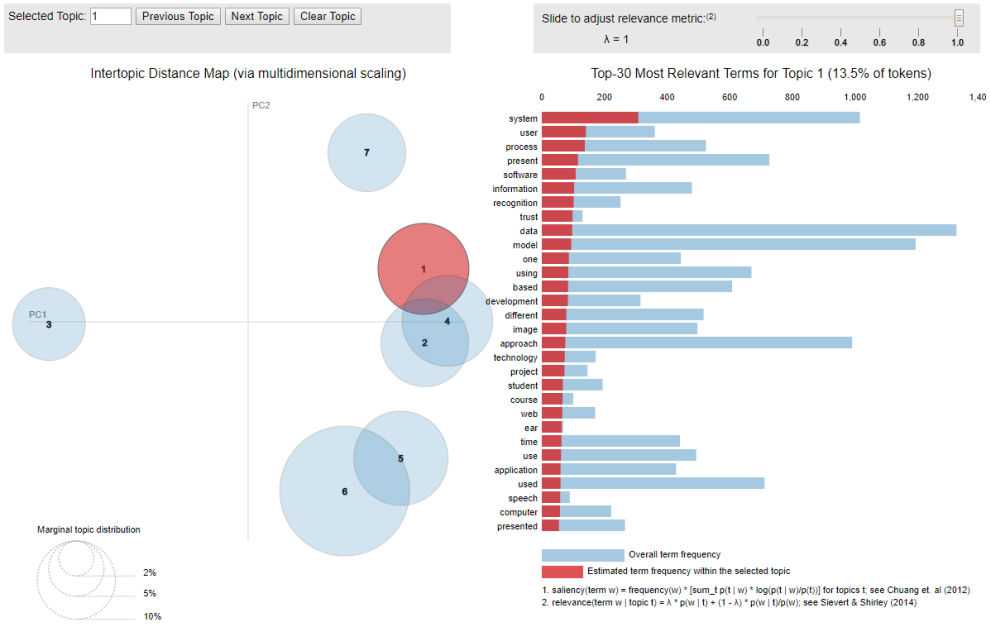
Slika 4.3: Število objavljenih člankov avtorjev za izbrano temo.

4.2 Knjižnica *pyLDAvis*

Knjižnica *pyLDAvis* [17] je najbolj uporabljena knjižnica za predstavitev tem odkritih z uporabo modelov *LDA*. Omogoča interaktivno primerjavo med odkritimi tematikami in odkrivanje sestave posamezne teme. Teme so predstavljene s krogi ravnini. Položaj centra kroga je določena z izračunom razdalje med temami. Z uporabo večdimenzionalnega lestvičenja (*MDS*) dobimo točke v ravnini. Bolj kot sta si temi podobni, bližje sta si središči njunih krogov. Velikost kroga predstavlja zastopanost teme v celotnem korpusu besedil.

Na desni strani se prikaže trideset besed z najvišjimi utežmi za izbrano

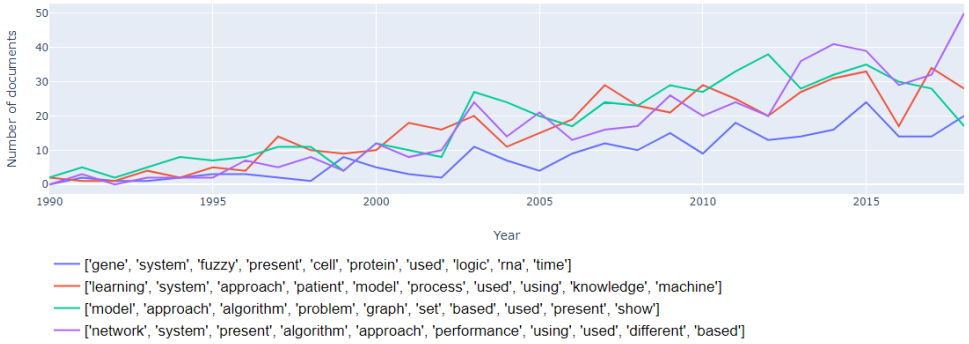
temo. Tu lahko vidimo, ali je beseda pogosta samo v izbrani temi, ali pa se pogosto pojavlja tudi pri ostalih dobljenih temah. Če miško postavimo na enega izmed krogov, se na desni prikažejo besede za izbrano temo. S pomočjo drsnika lahko spreminjamo parameter *lambda*. *Lambda* = 0 prikaže dvižno vrednost (ang. *lift*) besede. To je razmerje verjetnosti besede znotraj teme in verjetnosti besede znotraj celotnega korpusa besedil. *Lambda* = 1 prikaže verjetnost besede znotraj teme. Z izborom besede na desni strani vizualizacije se spreminjajo velikosti krogov, ki so sorazmerne s pomembnostjo izbrane besede za določeno temo.



Slika 4.4: Primer vizualizacije z uporabo knjižnice *pyLDAvis*.

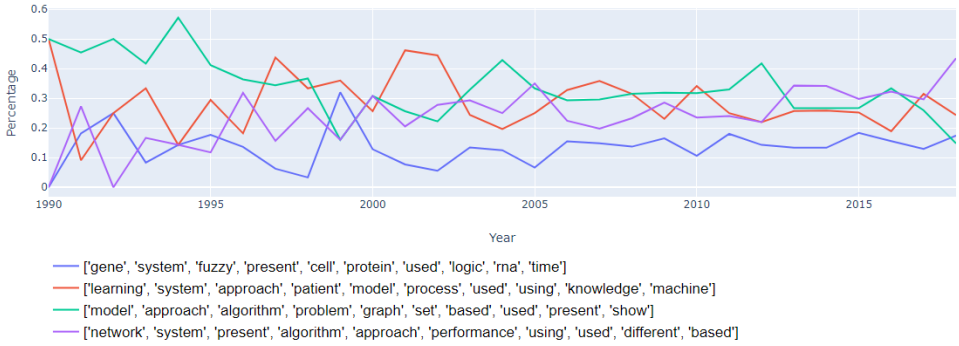
4.3 Prisotnost tem v odvisnosti od časa

Najbolj osnovna časovno odvisna predstavitev je prikaz števila člankov določene teme v enem letu. Vsem člankom določimo dominantno temo. To je tema, ki ji je model *LDA* za določen članek izračunal največjo utež. Po letih preštejemo število člankov z določeno temo in podatke predstavimo s pomočjo krivulje. Primer takšnega grafa je prikazan na sliki 4.5.



Slika 4.5: Število člankov posamezne tematike, v letih 1990-2018.

Ker je vsako leto objavljenih več člankov, se število člankov povečuje pri vseh temah. Da bi izničili vpliv števila vseh objavljenih člankov v danem letu, moramo število člankov v danem letu, ki pripadajo posamezni tematiki, deliti s številom vseh objavljenih člankov v letu. Tako dobimo normaliziran graf, prikazan na sliki 4.6.

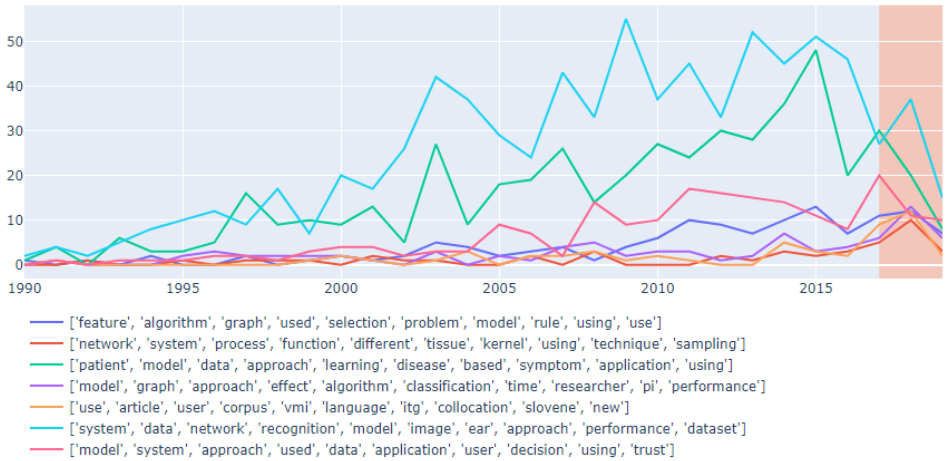


Slika 4.6: Normalizirano število člankov posamezne teme, v letih 1990-2018.

Ker so modeli zgrajeni nad celotno zbirko podatkov, ne povedo veliko o spremembah v trendih raziskovanja in spreminjanju raziskovalnih tematik. Zato je bolj smiselno zgraditi model na manjšem, izbranem časovnem obdobju in s tem dobiti teme, ki so bile tisti čas najbolj aktualne. Z uporabo modela *LDA* nato določimo teme preostalih dokumentov, ki so nastali izven izbranega obdobja.

Najprej iz dokumentov odstranimo besede, ki jih model *LDA* nima v svojem slovarju. Te so namreč za dobljene teme nepomembne, ker se v njih ne pojavljajo. Namesto seznamov besed, ki jih uporablja funkcija iz knjižnice *Gensim LdaModel*, smo uporabili množice. Razlog za to je, da je operacija preseka množic hitrejša kot primerjanje seznamov po elementih. Z uporabo preseka besed iz modela *LDA* in posameznega članka dobimo spremenjene članke, ki vsebujejo samo besede iz modela *LDA*. Dokumente nato preoblikujemo v vreče besed (ang. *bag-of-words*). Besedilo tako predstavimo kot seznam besed in s številom njihovih ponovitev. Pri tem namesto dejanske besede shranimo indeks, ki jo ima beseda v slovarju, ki je shranjen v modelu *LDA*. Vreče besed nato vstavimo v model *LDA*, ki za vsak članek vrne vektor uteži tem. Temu je namenjena funkcija *find_doc_topic()*. Funkcija ima dva parametra: število tem, ki nam jih funkcija vrne v primeru, da je članek sestavljen iz več tem, in meja uteži, da funkcija šteje, da določen dokument vsebuje temo.

Članke, ki jih nismo uporabili za izdelavo modela za teme v manjšem časovnem obdobju, predelamo z opisano funkcijo. Dobljene podatke normaliziramo, podobno kot v prejšnjem primeru. Zastopanost teme skozi čas nato ponazorimo z grafom, kot smo ga uporabili pri prejšnjih vizualizacijah. Na grafu dodatno označimo časovno območje člankov, ki so bili uporabljeni za odkrivanje tem. Primer takšnega grafa je prikazan na sliki 4.7.



Slika 4.7: Zastopanost tem UL FRI odkritih v letih 2016-2019.

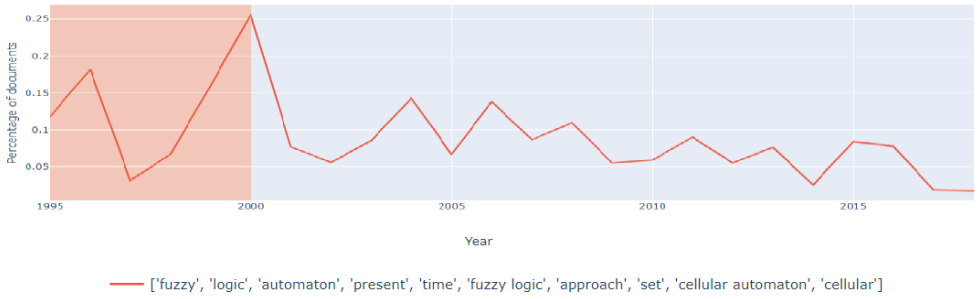
4.4 Razvoj tematik

Raziskav in posledično tudi novih odkritij je veliko. Raziskovalcem se porajajo vedno nova vprašanja in izzivi, kar prispeva k nastajanju novih raziskovalnih področij in tematik. Nekatera stara področja ostajajo aktualna, druga pa počasi izginjajo. Primer uporabe takega sistema je odkrivanje tem, ki so se na novo pojavljale, in izginjanje tem, ki so bile nekoč aktualne.

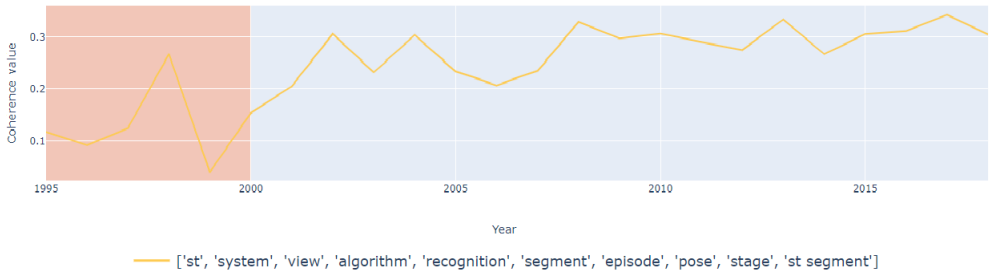
4.4.1 Izginjanje tem

Teme, ki izginjajo, so morale biti nekoč aktualne. Torej, zgradimo model *LDA* na nekem obdobju v preteklosti in opazujemo, kako so odkrite teme zastopane v novejših člankih.

Primer teme, ki skozi leta izgublja na zastopanosti v dokumentih je tema z besedami *fuzzy, logic, automaton, present, time, fuzzy logic, approach, set, cellular automaton* in *cellular*, ki je prikazana na sliki 4.8. Odkrita je bila med leti 1995 in 2000, kadar je dosegla tudi svoj višek. Po letu 2000 pa je njena zastopanost začela padati, kar nakazuje, da tema ni več tako aktualna. Leta 2018 se je pojavila v samo 1,739 odstotkov vseh člankov.



Slika 4.8: Primer izginjanja teme.



Slika 4.9: Primer pojavljanja nove teme.

4.4.2 Pojavljanje novih tem

Za pojavljanje novih tem velja obratno kot za izginjanje. Pričakujemo, da člankov, ki govorijo o teh temah, v preteklosti ni bilo oziroma jih ni bilo veliko. Hkrati pričakujemo, da se bo število teh člankov v kasnejšem obdobju povečalo.

Na sliki 4.9 je primer teme, ki se je pojavila med leti 1995 in 2000. Sestavljena je iz besed *st, system, view, algorithm, recognition, segment, episode, pose, stage* in *st segment*. Na podlagi besed *ST* in *ST segment* lahko sklepamo, da gre za temo povezano z medicino. Iz teh podatkov je razvidno, da so na naši fakulteti vedno bolj aktualne interdisciplinarne raziskovalne teme iz biomedicinskega področja.

Poglavje 5

Zaključek

V sklopu diplomske naloge smo razvili sistem, ki iz spleta pridobi najnovejše podatke o člankih izbrane skupine raziskovalcev, jih pripravi za obdelavo in na njih izvede modeliranje tem. S pomočjo pridobljenih tem lahko ocenimo, katere teme so nastale na novo, katere počasi izgubljajo na moči in katere so stalno prisotne. Hkrati oblikujemo zbirko podatkov o člankih in njihovi citiranosti. Sistem smo uporabili za analizo raziskovalnih področij, s katerimi se ukvarjajo raziskovalci na UL FRI. Z nekaj manjšimi spremembami bi lahko spremljali raziskovalne tematike drugih raziskovalnih skupin.

Za analizo člankov trenutno uporabljamo povzetke člankov. K boljši kakovosti modeliranja tem bi zagotovo pripomoglo izvajanje modeliranja na celotnih besedilih. Prav tako bi dobili boljše rezultate, če bi imeli na voljo več člankov. Trenutno je za UL FRI v portalu SICRIS na voljo 1783 vnosov iz *Scopus* in 1179 vnosov iz *Web of Science*. Število člankov se bo skozi leta seveda povečevalo in tako bo na voljo vedno več podatkov za obdelavo.

Literatura

[1] Langdetect. Dosegljivo: https://pypi.org/project/langdetect/. [Dostopano: 5. 9. 2019].

[2] MechanicalSoup. Dosegljivo: https://mechanicalsoup.readthedocs.io/en/stable/. [Dostopano: 5. 9. 2019].

[3] Scopus. Dosegljivo: https://www.elsevier.com/solutions/scopus. [Dostopano: 6. 9. 2019].

[4] SICRIS. Dosegljivo: http://www.sicris.si/about. [Dostopano: 6. 9. 2019].

[5] Web of Science. Dosegljivo: https://clarivate.com/webofsciencegroup/solutions/web-of-science/. [Dostopano: 6. 9. 2019].

[6] Thomas Hofmann. Probabilistic latent semantic analysis. volume 22, pages 289–296, 1999.

[7] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Perez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter [Unknown. Jupyter notebooks – a publishing format for reproducible computational workflows. 2016.

[8] Wei Li and Andrew Mccallum. Pachinko allocation: Dag-structured mixture models of topic correlations. pages 577–584, 2006.

[9] Tengfei Liu, Nevin Zhang, and Peixian Chen. Hierarchical latent tree analysis for topic detection. pages 256–272, 2014.

[10] Edward Loper and Steven Bird. Nltk: the natural language toolkit. *CoRR*, cs.CL/0205028, 2002.

[11] David M. Blei, Andrew Y. Ng, and Michael Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993, 2013.

[12] Wes Mckinney. pandas: a foundational python library for data analysis and statistics. *Python High Performance Science Computer*, 2011.

[13] David Mimno, Hanna M. Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 262–272, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[14] Ajda Pretnar and Tomaž Curk. BibMine. Dosegljivo: `https://github.com/tourism4-0/BibMine`. [Dostopano: 3. 7. 2019].

[15] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, 2010. ELRA. Dosegljivo: `https://radimrehurek.com/gensim/index.html`. [Dostopano: 5. 9. 2019].

[16] Nakatani Shuyo. Language detection library for Java. Dosegljivo: `http://code.google.com/p/language-detection/`, 2010. [Dostopano: 5. 9. 2019].

[17] Carson Sievert and Kenneth E. Shirley. LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces*, pages 63–70, 2014.

[18] Shaheen Syed and Marco R. Spruit. Full-text or abstract? Examining topic coherence scores using Latent Dirichlet Allocation. *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 165–174, 2017.