Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tadej Škvorc

Orodje za razporejanje člankov na konferencah

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Marko Robnik Šikonja
Somentor: prof. dr. Nada Lavrač

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Ena od težav, ki jo morajo rešiti programski vodje znanstvenih konferenc, je razporeditev sprejetih člankov v primerne predstavitvene termine. Za širšo publiko zanimive članke je potrebno razporediti med več plenarnih sekcij, ožje strokovno usmerjene članke pa v vzporedne sekcije tako, da bodo članki iz istega ožjega podpodročja v isti skupini, istočasno potekajoče vzporedne sekcije pa bodo pokrivale čimbolj različne tematike.

Sestavite orodje za podporo razporejanju člankov na podlagi postopnega gručenja. Omogočite avtomatsko razporeditev člankov in naknadno ročno izboljševanje razporeditve. Informacije o podobnosti člankov črpajte iz besedilnih informacij, kot so naslov, ključne besede in povzetek, poskusite pa uporabiti tudi informacijo o izraženem zanimanju recenzentov pri izbiranju člankov. Predlagajte način za ovrednotenje avtomatske razporeditve in primerjajte dobljeno razporeditev z ročno določeno.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Tadej Škvorc sem avtor diplomskega dela z naslovom:

Orodje za razporejanje člankov na konferencah

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Marka Robnika Šikonje in somentorstvom prof. dr. Nade Lavrač,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 4. septembra 2015 Podpis avtorja:

Zahvaljujem se svoji družini, mentorju in somentorici.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Uporabljene tehnologije **3**

2.1 Python 3

2.2 Django 4

2.3 PostgreSQL 4

2.4 HTML, javascript in CSS 4

2.5 Bootstrap 5

3 Opis aplikacije **7**

3.1 Arhitektura aplikacije 7

3.2 Podatkovna baza 9

3.3 Ročno upravljanje z urnikom 11

3.4 Samodejno razvrščanje člankov 14

3.5 Vizualizacija rezultatov 15

3.6 Izvoz urnika 16

4 Samodejno razvrščanje člankov **19**

4.1 Izgradnja podatkovne množice 20

4.2 Podatki o ocenjevalcih člankov 21

4.3 Metode gručenja 23

4.4 Razvrščanje člankov v urnik 24
4.5 Združevanje gruč 26
4.6 Vizualizacija rezultatov 27

5 Rezultati samodejnega razvrščanja **29**
5.1 Pregled testne množice podatkov 29
5.2 Primerjava uporabljenih metod gručenja 30
5.3 Analiza rezultatov 32
5.4 Učinkovitost uporabe ključnih besed 35

6 Možne izboljšave **37**
6.1 Uporabniški vmesnik 37
6.2 Samodejno razvrščanje člankov 37
6.3 Evaluacija rezultatov 39

7 Zaključek **41**

Seznam uporabljenih kratic

kratica	angleško	slovensko
AIME	Artificial Intelligence in Medicine	konferenca umetna inteligenca v medicini
csv	Comma Seperated Values file format	datoteka v formatu vrednosti ločenih z vejico
DS 2006	International Conference on Discovery Science 2006	mednarodna konferenca Znanost odkrivanja
DBSCAN	Clustering method Density-Based Spatial Clustering of Applications with Noise	metoda za gručenje prostorsko gručenje aplikacij s šumom na podlagi gostote
ICCC	International Conference on Computational Creativity	mednarodna konferenca o računalniški kreativnosti
MVC	Model-View-Controller architecture	razdelitev arhitekture aplikacije na model, pogled in krmilnik
ORDBMS	Object-Relational Database Management System	sistem za upravljanje z objektno relacijskimi podatkovnimi bazami
ORM	Object-Relational Mapping	objektno-relacijsko preslikovanje
pbkdf2	Password-Based Key Derivation Function 2	funkcija za izpeljavo ključa na osnovi gesla 2

PlanSoKD	Planning to Learn and Service-Oriented Knowledge Discovery	konferenca o načrtovanju učenja in ponudbeno usmerjenemu odkrivanju znanja
tf-idf	Term Frequency–Inverse Document Frequency	uteževanje vektorjev s frekvenco izrazov-inverzno frekvenco dokumentov
t-SNE	T-Distributed Stochastic Neighbor Embedding	metoda t-razporejeno vgrajevanje naključno razporejenih sosedov
xls	Microsoft Excel file format	datoteka v formatu programa Microsoft Excel

Povzetek

Organizacija akademskih konferenc je časovno potratno opravilo. Organizatorji konferenc si lahko trenutno pomagajo z majhnim številom programskih orodij, ki omogočajo upravljanje člankov in ročno upravljanje z urnikom konference. V diplomskem delu je predstavljeno inovativno spletno orodje za podporo organizacije konferenc, ki uporabniku poleg ročne izdelave urnika omogoča samodejno razvrščanje člankov v definirane časovne rezine urnika. Razvili smo preprost postopek, ki z uporabo algoritmov gručenja članke najprej združi v gruče glede na njihove medsebojne podobnosti in nato članke na podlagi teh gruč ustrezno razporedi v urnik. Uporabniku je ponujena vizualizacija rezultatov in možnost spreminjanja parametrov tega postopka.

Ključne besede: gručenje, spletna aplikacija, analiza besedil, organizacija konferenc, razporejanje člankov.

Abstract

Organizing academic conferences is a time consuming task. Conference organizers can currently use a small number of software tools that allow managing the paper review process and manual construction of the conference schedule. We present an innovative web application for conference organization support that can automatically assign papers into predefined schedule slots instead of requiring the program chairs to assign them manually. To achieve this functionality we have developed a method that uses clustering algorithms to first group the papers into clusters based on similarities between them and then assigns them into schedule slots. The user gets access to a visualization of the schedule and is able to experiment with various parameters of the method.

Keywords: clustering, web application, text analysis, conference organization, paper scheduling.

Poglavje 1

Uvod

Akademske konference, na katerih znanstveniki med seboj delijo nova odkritja, so ključne za razvoj znanosti. V zadnjih desetletjih so nekatere konference postale izjemno velike. Primer je vsakoletno srečanje društva za nevroznanost, ki v zadnjem desetletju vsako leto privabi okoli 30.000 obiskovalcev [8]. Tako velike konference trajajo več dni, na njih pa strokovnjaki predstavijo tudi več kot 10.000 znanstvenih člankov [10]. To je za znanstvene konference odlična statistika, vendar je njihova organizacija postala težka in zamudna.

Zaradi tega so strokovnjaki v zadnjih letih začeli razvijati računalniška orodja namenjena podpori organizacije konferenc. Glavna funkcionalnost, ki jo taka orodja ponujajo, je podpora oddajanju člankov in recenzentskemu procesu, ki vodi do izbire člankov, ki bodo predstavljeni na konferenci. Nekatera orodja podpirajo tudi razporeditev izbranih člankov v urnik. Tovrstnih orodij je več, primera najbolj pogosto uporabljenih pa sta *EasyChair* [4] in *OpenConf* [7]. Orodji imata skupno slabost, da je treba članke v urnik konference postaviti ročno.

Razporejanje člankov v urnik je časovno zahtevno opravilo. Najprej je potrebno članke razporediti v ločene seje glede na njihovo raziskovalno tematiko, seje pa je potem potrebno postaviti v urnik. Poleg tega morajo seje časovno ustrezati urniku konference. Organizatorjem konferenc bi zelo

koristil program, ki bi znal samodejno razporediti članke v seje glede na podobnost med njimi, poleg tega pa bi zagotavljal, da se tako ustvarjene seje časovno ujemajo s predvidenim urnikom.

V diplomskem delu predstavimo program, ki uporabnikom nudi to funkcionalnost. Gre za spletno aplikacijo, ki je zmožna rešiti dve težavi organizacije konferenc. Uporabnik lahko najprej ročno definira želeno strukturo urnika in uvozi članke, ki se bodo pojavili na konferenci. Ponudimo mu možnost samodejnega razporejanja člankov, ki z uporabo algoritmov gručenja združi podobne članke in jih razporedi v urnik. Dobljen urnik lahko uporabnik ročno spreminja in ga na koncu izpiše oziroma izvozi v obliki, ki je primerna za uporabo izven aplikacije.

V 2. poglavju opišemo tehnologije, ki smo jih uporabili pri izdelavi aplikacije. V 3. poglavju predstavimo arhitekturo aplikacije in možnosti, ki jih ponuja uporabniku. Nato v poglavju 4 predstavimo metodo, ki z uporabo algoritmov gručenja članke ustrezno združi v predhodno definirane skupine. Sledi predstavitev rezultatov, ki smo jih z aplikacijo dosegli (5. poglavje.) in predstavitev možnih izboljšav aplikacije (6. poglavje). 7. poglavje vsebuje zaključek diplomskega dela.

Poglavje 2

Uporabljene tehnologije

V tem poglavju predstavimo nekatere pomembnejše tehnologije, ki smo jih uporabili za razvoj aplikacije.

2.1 Python

Python [1] je splošno namenski skriptni programski jezik, ki je v zadnjih letih postal priljubljen predvsem zaradi razmeroma preproste uporabe. Pri izdelavi aplikacije smo se za python odločili zato, ker omogoča preprost dostop do odprtokodnih knjižnic za besedilno analizo in strojno učenje, poleg tega pa je mogoče z uporabo ogrodja Django v jeziku python pisati spletne aplikacije.

2.1.1 NumPy

NumPy je programski paket za jezik python, ki ponuja boljše matematične funkcije in konstrukte kot sam python. Naša aplikacija iz tega paketa uporablja predvsem metode za računanje z redkimi matrikami. Poleg tega je paket NumPy potreben za uporabo paketa scikit-learn, ki je podrobneje opisan v razdelku 2.1.2.

2.1.2 Scikit-learn

Scikit-learn [28] je programski paket za jezik python, ki ponuja veliko število različnih metod s področja podatkovnega rudarjenja, analize besedil in strojnega učenja. Za našo aplikacijo so iz paketa scikit-learn pomembni algoritmi gručenja (*clustering*) in metode za analizo besedil.

2.2 Django

Django [11] je spletno ogrodje za jezik python, ki omogoča izdelavo spletnih aplikacij. Kljub temu, da jezik python prvotno ni bil namenjen spletnemu programiranju, obstajajo ogrodja, ki omogočajo uporabo jezika za ta namen. Takih ogrodij je več - med nekatere najbolj pogosto uporabljene spadajo Django, Pyramid, Flask, Tornado in še mnogi drugi. Za Django smo se odločili zato, ker je izmed vseh spletnih ogrodij eden najbolj celovitih in priljubljenih.

2.3 PostgreSQL

Glede na to, da gre za spletno aplikacijo, je uporaba podatkovne baze skoraj nujna. Odločili smo se za uporabo podatkovne baze PostgreSQL [9], ki je odprto-koden sistem za upravljanje z objektno-relacijskimi podatkovnimi bazami (ORDMBS, ali object-relational database management system). Za uporabo tega sistema smo se odločili predvsem zato, ker je izjemno robusten, odprtokoden in dobro podprt v ogrodju Django.

2.4 HTML, javascript in CSS

Za programiranje spletnega vmesnika smo uporabili označevalni jezik HTML [5], jezik CSS [3] in programski jezik javascript [15]. HTML je označevalni jezik, ki določa strukturo in vsebino spletnih strani, jezik CSS jim določi izgled, programski jezik javascript pa doda naprednejšo interaktivnost. Pri

uporabi jezika HTML smo uporabili različico HTML5, saj doda podporo za funkcionalnost povleci in spusti (*drag and drop*), kar aplikaciji doda boljšo uporabniško izkušnjo.

2.5 Bootstrap

Bootstrap [2] je ogrodje za izdelavo odzivnih spletnih strani z modernim izgledom. V principu gre za zbirko knjižnic javascripta in datotek CSS, ki poskrbijo za lep izgled spletne strani.

Poglavje 3

Opis aplikacije

V tem poglavju najprej predstavimo arhitekturo aplikacije in principe, na katerih je ta arhitektura zasnovana. Nato podrobneje opišemo delovanje aplikacije in predstavimo njeno funkcionalnost.

3.1 Arhitektura aplikacije

Arhitektura naše aplikacije temelji na splošno uveljavljenih principih odjemalec-strežnik in model-pogled-krmilnik (*Model-View-Controller ali MVC*). Ta principa sta v arhitekturah današnjih spletnih aplikacij zelo pogosta.

3.1.1 Model odjemalec-strežnik

Model odjemalec-strežnik (client-server model) [13] opisuje arhitekturo aplikacij, ki so v grobem razdeljene na dva dela. **Strežniški** del se nahaja na neki uporabniku oddaljeni lokaciji in pogosto vključuje večino poslovne logike in podatkov, ki so potrebni za delovanje aplikacije. Do strežniškega dela hkrati dostopa več **odjemalcev**, ki prikazujeju uporabniški vmesnik in uporabniku omogočajo dostop do aplikacije.

3.1.2 MVC

Arhitektura aplikacije je zasnovana na principu model-pogled-krmilnik (MVC, ali *model-view-controller*) [25]. Ta princip je v zadnjih letih postal izjemno priljubljen pri spletnih aplikacijah, tako da na njem temelji veliko število spletnih ogrodij - med njimi tudi ogrodje Django, v katerem je izdelana naša aplikacija.

Princip MVC razdeli aplikacijo na tri dele:

Model opisuje podatke sistema in poslovno logiko ter spreminja in upravlja s temi podatki. Pri naši aplikaciji gre za objekte, ki opisujejo glavne podatke sistema - kot so na primer podatki o uporabnikih, konferencah in člankih - ter metode, ki te podatke uporabljajo.

Pogled opisuje del sistema, ki uporabniku ponuja interakcijo s sistemom. Pri naši aplikaciji so to posamezne spletne strani, ki uporabniku omogočajo na primer dodajanje ter razvrščanje člankov.

Krmilnik služi kot posrednik med modelom in pogledom. Krmilnik prevede uporabnikove ukaze ter jih posreduje modelu, ki nato nad podatki izvrši ustrezne operacije.

3.1.3 Arhitektura

Arhitektura naše aplikacije je sestavljena iz treh delov, kjer vsak ustreza enemu izmed delov principa MVC:

Podatkovni strežnik - Aplikacija za hranjenje podatkov uporablja strežnik, na katerem teče sistem za upravljanje relacijskih podatkovnih baz PostgreSQL. Na tem strežniku so shranjeni podatki, potrebni za delovanje aplikacije.

Aplikacijski strežnik - Poslovna logika naše aplikacije se nahaja na aplikacijskem strežniku, ki za delovanje uporablja spletno ogrodje Django.

Odjemalec - Uporabnik do aplikacije dostopa preko odjemalca, ki je implementiran kot spletni vmesnik in je uporabniku na voljo preko sodobnih spletnih brskalnikov. Gre za skupino spletnih strani, ki uporabniku omogočajo interakcijo z aplikacijo. Odjemalec za delovanje uporablja tehnologije HTML5, CSS in javascript.

3.2 Podatkovna baza

Pri izdelavi aplikacije smo uporabili klasično relacijsko podatkovno bazo, in sicer sistem PostgreSQL, do katerega aplikacija dostopa z uporabo preslikav ORM (*Object-Relational Mapping*), ki jih ponuja Django. To pomeni, da aplikacija ni vezana na uporabo specifične podatkovne baze, temveč bi lahko PostgreSQL brez večjih težav zamenjali s katerokoli bazo, ki jo podpira Django (to so - brez uporabe dodatnih knjižnic - PostgreSQL, MySQL in SQLite).

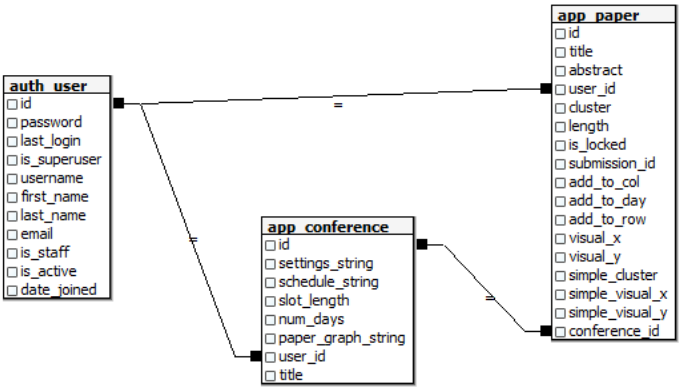
Uporabljeno podatkovno bazo sestavljajo sledeče tabele:

User - Ta tabela opisuje uporabnike aplikacije. Vsak uporabnik ima določeno uporabniško ime in geslo, s katerim se prijavi v aplikacijo. Zaradi varnostnih razlogov so vsa gesla v podatkovni bazi shranjena v obliki rezultata kriptografske razpršilne funkcije (*hash function*) pbkdf2 (*password-based key derivation function 2*). Funkcija geslo preslika v nov niz znakov na način, da je iz novega niza težko dobiti nazaj originalen niz in s tem zaščiti geslo uporabnika v primeru nepooblaščenega vdora v podatkovno bazo.

Conference - Vsak uporabnik lahko v aplikaciji ustvari več konferenc, ki se shranijo v tabelo. Vsaka konferenca ima svoj naslov (*title*) ter identifikacijsko številko uporabnika, ki jo je ustvaril *(user_id)*. Poleg tega vsaka konferenca vključuje še podatke o urniku konference. Ti podatki so shranjeni v poljih *settings_string*, ki hrani strukturo urnika, in *schedule_string*, ki hrani, kje v urniku se nahajajo posamezni članki.

Prisotni sta še manj pomembni polji *slot_length*, ki določa dolžino novododanih polj v urniku, in *num_days*, ki določa, koliko dni traja konferenca.

Paper - V tej tabeli so shranjeni članki, ki so del urnika posamezne konference. Vsak članek je opisan s svojim naslovom (*title*) in s svojim povzetkom (*abstract*). Pod osnovne podatke o članku spadata še podatka o uporabniku, ki je članek ustvaril (*user_id*), in o konferenci, ki ji članek pripada (*conference_id*).



Slika 3.1: Shema glavnih tabel podatkovne baze.

Poleg osnovnih podatkov vsak članek potrebuje še podatke o njegovi lokaciji v urniku. Lokacija je opisana v poljih *add_to_day*, *add_to_row* in *add_to_col*. Te podatki so zadostni za opis urnika konference, ki lahko traja več dni in pri kateri je lahko na urniku prisotnih več člankov hkrati. Prisotno je še polje *length*, ki opisuje trajanje posamezne predstavitve članka.

Potrebni so še podatki za opis gručenja, ki ga ustvari samodejno razporejanje člankov. V ta namen se uporabijo polja *simple_cluster* in *cluster*, v katera se shrani zaporedna številka gruče, ki jo je gručenje priredilo posameznemu članku, ter polja *simple_visual_x*, *simple_visual_y*,

visual_x ter *visual_y*, v katera se shranijo koordinate, potrebne za vizualizacijo gručenja. Z uporabo polja *is_locked* je mogoče posamezne članke izločiti iz samodejnega razporejanja - samodejno razporejanje člankov se namreč izvede le nad članki, kjer velja *is_locked = FALSE*.

Celotna shema podatkovne baze (brez tabel, ki jih samodejno ustvari ogrodje Django), je prikazana na sliki 3.1.

3.3 Ročno upravljanje z urnikom

Vsebinsko je aplikacija razdeljena na dva ločena dela. Prvi je predstavljen v tem razdelku in omogoča uporabniku upravljanje s konferencami, ter ročno izdelavo urnika posamezne konference. Drugi del, ki uporabniku nudi možnost samodejnega razporejanja člankov v urnik, je predstavljen v razdelkih 3.4 in 3.5.

3.3.1 Prijava v sistem

Pred uporabo aplikacije se mora uporabnik prijaviti v sistem z uporabniškim imenom in geslom, ki si ga je izbral pri registraciji. To aplikaciji omogoča, da za vsakega uporabnika samodejno hrani vse ustvarjene konference in njihove urnike, do katerih lahko nato uporabnik dostopa iz poljubnega brskalnika.

3.3.2 Upravljanje s konferencami

Uporabnik najprej vidi stran za upravljanje s konferencami. V kolikor uporabnik še ni ustvaril nobene konference, bo imel na voljo le opcijo za ustvarjanje nove konference s poljubnim imenom (gumb **Create conference**). V primeru, ko uporabnik že ima ustvarjeno vsaj eno konferenco, lahko z uporabo gumba **Delete conference** izbriše posamezno konferenco, z gumbom **Rename conference** preimenuje konferenco in z gumbom **Copy conference structure** ustvari novo konferenco, pri čimer se za strukturo urnika

uporabi urnik že obstoječe konference. To je koristno pri vsakoletnih konferencah, pri katerih struktura urnika iz leta v leto ostane podobna. Vsakič, ko uporabnik ustvari ali izbriše konferenco, aplikacija posodobi podatkovno bazo in vanjo ali shrani novo konferenco, ali pa izbrano konferenco izbriše.

Uporabnik lahko s klikom na ime posamezne konference to konferenco izbere. Ko uporabnik izbere konferenco, se mu prikaže glavni vmesnik aplikacije. V njem lahko dostopa do ročnega razporejanja člankov, urejanja urnika, samodejnega razporejanja člankov in izvoza urnika.

3.3.3 Urejanje urnika

Na strani za urejanje urnika lahko uporabnik definira strukturo urnika konference. Najprej mora nastaviti število dni konference in dolžino posamezne **časovne rezine urnika**. Dolžino lahko pozneje spreminja za vsako časovno rezino posebej.

Časovne rezine urnika predstavljajo osnovno enoto urnika konference. Vsaka časovna rezina ima svojo tematiko in je sestavljena iz predstavitev člankov, ki so vezani na to tematiko. Uporabnik lahko nastavi začetno dolžino časovnih rezin preko polja *minutes per block*, pozneje pa lahko ročno spremeni dolžino posamezne časovne rezine.

Gumba *add slot* in *add parallel slots* služita za dodajanje časovnih rezin v urnik. Gumb *add slot* doda v urnik eno zaporedno časovno rezino, gumb *add parallel slots* pa v urnik doda več vzporednih časovnih rezin, ki bodo potekale istočasno. Uporabnik lahko z gumbi *delete* izbriše posamezne časovne rezine, z gumbom *change* pa spremeni njeno dolžino.

Poleg tega lahko uporabnik za vsak dan posebej nastavi uro začetka konference. Ta možnost je pomembna predvsem za izvoz urnika.

3.3.4 Uvoz člankov

Preden lahko uporabnik v urnik začne dodajati članke, jih mora najprej uvoziti v aplikacijo. To lahko stori na strani *manual program*. Ta stran je

razdeljena na dva dela - eden služi uvažanju člankov, drugi pa uporabniku dopušča ročno dodajanje člankov v urnik.

Članke lahko uporabnik v aplikacijo uvozi na dva načina:

- **Ročno dodajanje** - če uporabnik pritisne na gumb *Add paper* se mu prikaže obrazec za ročni vnos članka. Uporabnik lahko vnese naslov, povzetek članka in dolžino njegove predstavitve na konferenci ter ga s tem doda v aplikacijo.

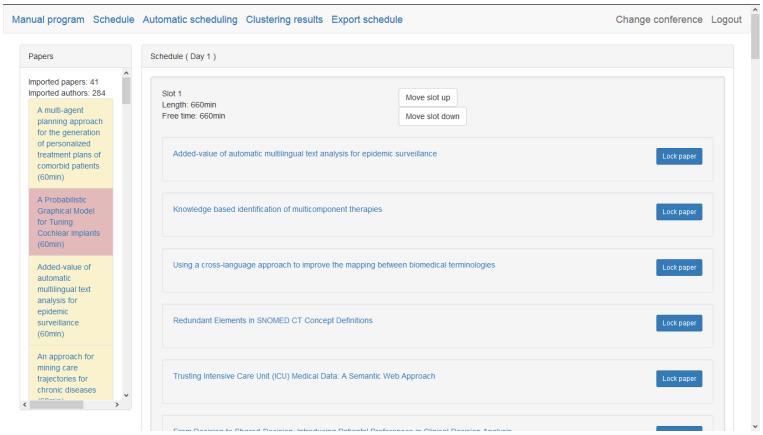
- **Uvoz podatkov iz sistema EasyChair** - aplikacija poleg ročnega dodajanja člankov podpira tudi uvoz člankov iz sistema EasyChair. EasyChair je eden izmed najpogosteje uporabljenih sistemov za upravljanje znanstvenih konferenc. Za uvoz člankov mora uporabnik najprej izvoziti članke iz sistema EasyChair. Uporabnik nato izbere datoteko, v katero je izvozil članke, in pritisne na gumb *Import papers*.

  Alternativno lahko uporabnik sam ustvari datoteko, ki je ekvivalentna datoteki, ki jo EasyChair ustvari ob izvozu člankov. Datoteka mora biti tipa .xls (format programa Microsoft Excel), v kateri vsaka vrstica predstavlja en članek, stolpci pa po vrsti hranijo podatke o identifikacijski številki članka, avtorjih (ločenih z vejico), naslovu in povzetku članka.

Vmesnik poleg uvoza člankov uporabniku omogoča tudi uvoz podatkov o dodelitvi člankov različnim ocenjevalcem. Ti podatki so koristni za samodejno razvrščanje člankov, ki je podrobneje opisano v poglavju 4.

3.3.5 Razporejanje člankov

Stran *manual program* uporabniku omogoča ročno razporejanje člankov v urniku. Vsi članki, ki jih je uporabnik predhodno dodal v aplikacijo, so prikazani v stranski vrstici. Uporabnik jih v urnik doda tako, da jih povleče v željno časovno rezino. To je možno le, če je v časovni rezini še kaj prostega časa. Uporabnik lahko dodane članke *zaklene* z gumbom *lock paper*. To



Slika 3.2: Zaslonska slika glavnega vmesnika.

pomeni, da jih samodejno razporejanje člankov ne bo premaknilo iz dodeljene časovne rezine.

Če uporabnik ob postavljanju člankov v urnik ugotovi, da mu postavitev časovnih rezin ne ustreza, jih lahko z gumboma *move slot up* in *move slot down* premakne na drugo mesto v urniku. Pri tem se skupaj s časovno rezino premaknejo tudi vsi članki, ki so bili dodeljeni časovni rezini.

Izgled vmesnika, ki uporabniku omogoča ročno razporejanje člankov je prikazan na sliki 3.2.

3.4 Samodejno razvrščanje člankov

Kot alternativa ročnemu urejanju urnika je uporabniku na voljo tudi samodejno razporejanje člankov. Samodejno razporejanje člankov v urniku je inovativna funkcionalnost naše aplikacije, ki ni na voljo v ostalih podobnih programih. Deluje na principu metod strojnega učenja in analize besedil. Te metode so podrobneje predstavljene v poglavju 4.

Uporabnik do samodejnega razvrščanja člankov dostopa preko strani *automatic scheduling*. Na tej strani lahko uporabnik upravlja z različnimi nastavitvami, ki vplivajo na samodejno razporejanje člankov. Glavne nastavitve,

ki jih uporabnik lahko spreminja, so sledeče:

Funkcija gručenja - Uporabnik lahko izbere, katera izmed ponujenih funkcij se bo uporabila za združevanje člankov v gruče. To lahko vpliva na kakovost končnega rezultata in na hitrost samodejnega razporejanja člankov. Uporabniku so glede na izbrano funkcijo na voljo različni parametri, s katerimi lahko spremeni delovanje funkcije.

Ključne besede - V primeru da uporabnik že ima okvirno idejo, v kakšne skupine bi rad članke grupiral, lahko pred začetkom samodejnega razvrščanja člankov definira skupino ključnih besed. Vpisane ključne besede se bodo pri analizi besedil štele za pomembnejše od ostalih besed. Z uporabo ključnih besed lahko uporabnik vpliva na tematiko posameznih skupin, ki bodo nastale po samodejnem razvrščanju člankov.

Uporabljeni podatki - Uporabnik lahko izbira, kateri podatki člankov bodo uporabljeni pri samodejnem razvrščanju člankov. Na voljo ima naslov in povzetek člankov ter podatke o ocenjevalcih člankov (le v primeru, če jih je predhodno uvozil v aplikacijo).

Ko uporabnik izbere želene nastavitve lahko s pritiskom gumba *Start automatic clustering* začne s postopkom samodejnega razporejanja člankov.

3.5 Vizualizacija rezultatov

Po končanem postopku samodejnega razporejanja člankov se uporabniku prikaže vizualizacija rezultatov. Do nje lahko pozneje dostopa tudi s pritiskom na povezavo *Clustering results*, ki je na voljo na navigacijski vrstici. Glavni del vizualizacije rezultatov predstavlja točkovni diagram, na katerem so prikazani vsi članki. Poleg tega je v stranski vrstici prikazan seznam člankov, ki so grupirani po gruči, v katero jih je razporedilo samodejno razvrščanje člankov.

Slika 3.3: Zaslonska slika vizualizacije rezultatov.

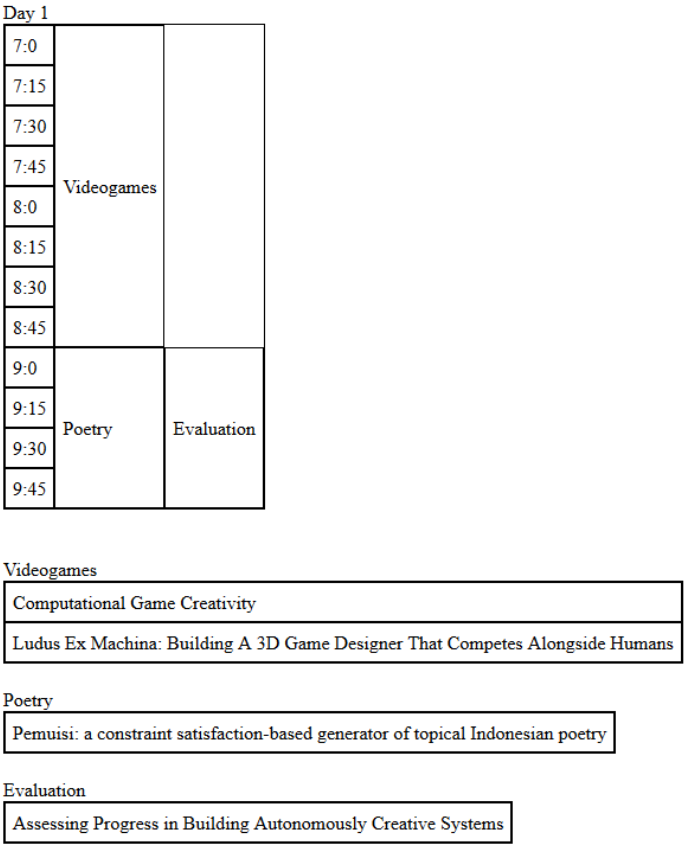
Točkovni diagram uporabniku ponuja dva različna pogleda. Prvi uporabniku prikaže vse članke in njihove prirejene gruče. Drugi pogled, do katerega lahko uporabnik dostopi s pritiskom na povezavo *Show only papers that were newly assigned to the schedule*, uporabniku prikaže le članke, ki so bili v postopku samodejnega razvrščanja na novo dodani v urnik. Pri tem pogledu so članki grupirani po časovnih rezinah urnika, v katere so bili dodeljeni. Izgled strani z vizualizacijo je prikazan na sliki 3.3.

3.6 Izvoz urnika

Ko uporabnik konča z izdelavo urnika, bo želel urnik izvoziti v obliki, ki jo bo lahko uporabil tudi zunaj naše aplikacije (recimo na spletni strani konference). V ta namen je v aplikaciji prisoten gumb *Export schedule*. S pritiskom na ta gumb se uporabniku prikaže stran, na kateri je najprej prikazana struktura urnika konference brez člankov, ki bodo na konferenci predstavljeni (prikažejo se le časovne rezine). Pod strukturo urnika se nahaja še seznam člankov grupiranih po časovni rezini, v kateri bodo predstavljeni.

Taka predstavitev urnika je bila izbrana zato, ker je bolj pregledna, kot če bi le preprosto prikazali urnik napolnjen z vsemi članki. Člankov je namreč

na konferencah lahko ogromno, zaradi česar taka predstavitev ne bi bila smiselna. Predstavitev naše aplikacije ustreza izgledu urnikov velikega števila znanstvenih konferenc in je prikazana na sliki 3.4.



Slika 3.4: Zaslonska slika izvoza urnika.

Poglavje 4

Samodejno razvrščanje člankov

Osrednja funkcionalnost, ki jo naša aplikacija ponuja uporabniku, je možnost samodejnega razvrščanja člankov v proste časovne rezine urnika glede na podobnost med članki. V tem poglavju podrobneje opišemo postopke in metode, ki smo jih uporabili pri implementaciji te funkcionalnosti.

Samodejno razvrščanje člankov v urnik se je izkazal za težak problem. Naša aplikacija mora za pravilno delovanje namreč rešiti dva ločena problema:

- Članke, ki jih je uporabnik uvozil v aplikacijo, je najprej potrebno razdeliti v večje število gruč glede na podobnost med članki. Za reševanje tega problema smo si pomagali z metodami analize besedil in metodami gručenja.

- Ko so članki razvrščeni v gruče, jih je potrebno dodati v proste časovne rezine urnika. Pri tem smo naleteli na problem, saj ustvarjene gruče po skupni dolžini člankov skoraj nikoli ne ustrezajo dolžinam prostih časovnih rezin. Za rešitev tega problema smo razvili preprost nov algoritem, ki z uporabo hevrističnih metod iz gruč izbere ustrezne članke za posamezno časovno rezino.

4.1 Izgradnja podatkovne množice

Za rešitev prvega problema mora aplikacija najprej zgraditi podatkovno množico, ki mora biti ustrezna za obdelavo z metodami gručenja. Aplikacija mora za vsak primer v množici (pri naši aplikaciji je primer ekvivalenten članku, ki ga je v aplikacijo uvozil uporabnik) zgraditi vektor, v katerem vsaka komponenta opisuje neko lastnost primera. Podatkovno množico dobimo tako, da zgrajene vektorje združimo v matriko. Matriko posredujemo algoritmom gručenja.

Pri uvozu člankov v aplikacijo lahko določimo dva glavna vira informacij - naslov članka in besedilo povzetka članka. Ta podatka sta aplikaciji na voljo le v besedilni obliki, ki neposredno ni primerna za uporabo v metodah gručenja. Lahko bi uporabili še podatke o avtorjih besedila, vendar je bilo po pregledu podatkov razvidno, da si članki z iste konference zelo redko delijo avtorje. To pomeni, da iz podatkov o avtorjih skoraj nikoli ni mogoče dobiti koristnih informacij za gručenje. Najprej je potrebno torej pretvoriti podatke o naslovu in povzetku članka v ustrezno vektorsko predstavitev.

Pred pretvorbo primerov v vektorje nad besedilom izvedemo preprosto predobdelavo. Najprej pretvorimo vse velike začetnice v male črke in posebne črke kot so č/ć v njihovo osnovno obliko (c). Nato odstranimo besede brez večjega vpliva na pomen besedila, kot so *and*, *or*, *a* in *the*. Pri tem uporabimo slovar nepotrebnih besed v angleščini (*stop words*), saj so konference večinoma mednarodne, to pa pomeni, da je večina člankov v angleškem jeziku.

Nato izvedemo pretvorbo besedil v vektorsko obliko. To storimo z uporabo standardnih metod analize besedil, ki besedilno predstavitev besedila pretvorijo v **vektor tf-idf** [30] (*term frequency-inverse document frequency*). To je vektor, kjer vsak element vektorja predstavlja frekvenco pojavitve posamezne besede v originalnem besedilu, vsaka frekvenca pojavitve pa je utežena s številom pojavitev besede v celotni množici besedil, kar zagotovi, da so bolj pogoste besede manj utežene kot slovnično redkejše, saj slabše ločijo med podobnimi besedili.

V primeru, da je uporabnik pri nastavitvah avtomatskega razvrščanja člankov definiral ključne besede, bo vektor tf-idf zgrajen le iz izbranih ključnih besed. Prav tako lahko izbere, kateri deli člankov se bodo uporabili za izgradnjo vektorja tf-idf - naslov, povzetek ali oboje.

4.2 Podatki o ocenjevalcih člankov

Poleg besedilnih podatkov o naslovu in povzetku članka lahko za gručenje člankov uporabimo tudi podatke o procesu ocenjevanja člankov. Ta proces poteka pred začetkom večine konferenc. Med tem procesom se določijo recenzenti, ki za vsak članek ocenijo ali je primeren za predstavitev na konferenci. Za lažje delo pred začetkom ocenjevanja vsak ocenjevalec izrazi mnenje o tem, katere članke želi oceniti in katerih raje ne bi. Podatke o tem, katere članke je posamezen ocenjevalec želel oceniti, lahko uporabimo za gručenje člankov, saj ti podatki namigujejo na določeno povezanost med članki. Ocenjevalci namreč raje ocenjujejo članke, ki so povezani z njihovo raziskovalno tematiko.

Če želimo te podatke uporabiti v algoritmih gručenja, jih je najprej potrebno pretvoriti v ustrezno obliko. Podatke o ocenjevalcih uvozimo iz csv datoteke (datoteka vrednosti ločenih z vejico), kjer vsaka vrstica opisuje en članek in kjer vsako polje v vrstici opisuje mnenja ocenjevalcev o teh člankih. Ocenjevalci lahko za vsak članek izberejo eno izmed naslednjih mnenj:

- hočem oceniti ta članek,
- lahko ocenim ta članek,
- članka ne morem oceniti zaradi konflikta interesov,
- članka ne želim oceniti.

Ko uvozimo datoteko z mnenji ocenjevalcev jo najprej pretvorimo v graf. V tem grafu vozlišča predstavljajo posamezne članke. Članka sta v grafu povezana, če je o njiju isti ocenjevalec podal mnenje, ki ni *članka ne želim*

oceniti ali *članka ne morem oceniti zaradi konflikta interesov*. Povezave med vozlišči so v grafu neusmerjene in so utežene po formuli:

$U_1 * U_2$, kjer je U_x konstanta posameznega članka določena po formuli:

$$U_x = \begin{cases} 1 \text{ za mnenje Lahko ocenim članek} \\ 2 \text{ za mnenje Hočem oceniti ta članek} \end{cases} \tag{4.1}$$

Tak graf je pri gručenju zelo koristen, saj lahko iz njega ugotovimo podobnost člankov glede na ekspertno mnenje ocenjevalcev in ne le glede na podobnost besedilne analize. V grafu bodo članki, ki jih isti ocenjevalec želi oceniti, med seboj povezani. To pomeni, da povezani članki verjetno spadajo k isti tematiki in so si torej med seboj podobni. Poleg tega bosta članka, ki si ju avtor močno želi oceniti (in je zanju podal mnenje *hočem oceniti ta članek*) povezana z večjo utežjo kot članka, za katera je avtor podal mnenje *lahko ocenim ta članek*. Predpostavimo namreč lahko, da članki za katere je avtor podal le mnenje *lahko ocenim ta članek* niso tako močno povezani z njegovo raziskovalno tematiko kot tisti, ki jih močno želi oceniti.

Graf na žalost ni neposredno primeren za uporabo v algoritmih gručenja. Prav tako kot je to veljalo za besedilne podatke, je tudi podatke o mnenjih ocenjevalcev potrebno najprej pretvoriti v vektorsko obliko. Za to pretvorbo smo uporabili metodo predstavljeno v članku Grčar et al. [20], ki pretvori poljuben utežen graf v vektorsko predstavitev z uporabo algoritma personalized PageRank [26]. Metoda iz vhodnega grafa za vsak članek zgradi vektor, ki opisuje njegove povezave z drugimi članki. Taki vektorji so ustrezni za uporabo v algoritmih gručenja, poleg tega pa se jih lahko zelo preprosto doda vektorjem, ki jih aplikacija dobi iz besedilne analize naslova in osnutka člankov (potrebno jih je samo pripeti na konec originalnega vektorja). To pomeni, da lahko brez težav za gručenje uporabimo poljubno kombinacijo vektorjev naslova, osnutka in mnenja ocenjevalcev.

4.3 Metode gručenja

Ko zgradimo ustrezno podatkovno množico, lahko nad njo poženemo metode gručenja. Te metode kot vhod sprejmejo matriko, kjer vsaka vrstica opisuje en primer (v naši aplikaciji je to posamezen članek), vsak stolpec pa opisuje eno lastnost primera. Na podlagi vhodne matrike metode gručenja nato izračunajo podobnost med podanimi primeri in jih na podlagi različnih parametrov združijo v skupine (gruče). Te gruče v nadaljevanju uporabimo za samodejno razvrščanje člankov.

Aplikacija uporabniku pred začetkom samodejnega razvrščanja člankov ponudi izbiro med sledečimi metodami gručenja:

Affinity propagation [17] - Gre za metodo, ki vhodne primere razvrsti v skupine tako, da med njimi poišče primere, ki najbolje predstavljajo določeno skupino (*exemplarje*) in nato poveže njim podobne primere v gručo. Prednost te metode je v tem, da ji ni potrebno predhodno podati števila gruč.

DBSCAN [14] - DBSCAN je ena izmed najpogosteje uporabljenih metod gručenja. Sposobna je vračati dobre rezultate (tudi če vhod vključuje gruče poljubnih oblik, kjer druge metode ne delujejo vedno najbolje) in je hitra tudi pri veliki količini vhodnih primerov. Prav tako kot *affinity propagation* ne potrebuje vnaprej podanega števila gruč, namesto tega kot argument sprejme minimalno število primerov v gruči.

Na žalost se pri naši domeni ta metoda ni izkazala za uporabno. Slabost metode DBSCAN je namreč v tem, da težko razlikuje med podatki z velikim številom atributov. To pomeni, da ima težave, če kot vhod dobi vektorje z velikim številom dimenzij. Takim vektorjem se pri besedilni analizi z uporabo matrik tf-idf ni mogoče izogniti.

Hierarchical clustering [24] je ena starejših metod gručenja. Deluje tako, da na začetku vsak članek razvrsti v lastno gručo in nato združuje gruče, dokler ne doseže števila gruč, ki ga uporabnik poda kot para-

meter. Na naši domeni deluje precej dobro, vendar je počasna. To je lahko pri spletnih aplikacijah težava, še posebej, če aplikacijo hkrati uporablja veliko število uporabnikov.

K-Means [22] je metoda, ki vhodne primere razporedi v N skupin tako, da je seštevek kvadratov razdalj primerov od sredine posameznih skupin čim manjši. Je zelo splošna metoda ki ne ponuja veliko fleksibilnosti, zaradi česar ne da vedno dobrih rezultatov. Zahteva, da uporabnik število gruč poda v naprej, kar oteži delo.

V povezavi z metodo K-means aplikacija ponuja tudi variacijo **mini batch K-means** [31]. Ta metoda izboljša hitrost, vendar lahko vrne nekoliko slabše rezultate kot osnovna metoda K-means.

Mean-shift [18] - Mean-shift je jedrna (*kernel*) metoda strojnega učenja. Na začetku ustvari določeno število jeder. Vsako jedro vključuje tiste vhodne primere, ki se nahajajo na določeni razdalji okoli sredine jedra. Metoda nato spreminja meje jeder, dokler premiki niso več koristni.

Metoda je dovolj splošna, da jo lahko uporabimo v skoraj vsaki domeni. Slabost metode je v tem, da za dobro delovanje potrebuje natančno nastavljene parametre - predvsem parameter pasovne širine (*bandwidth*). Ta parameter mora biti prilagojen vhodnim podatkom, ki pri naši aplikaciji ne bodo vedno enaki. To pomeni, da mora aplikacija vrednost parametra izračunati vsakič, ko uporabnik zažene samodejno razvrščanje člankov. Pri tem si pomaga s pomožno funkcijo (`sklearn.cluster.estimate_bandwidth`), ki je na voljo v paketu *scikit-learn*.

4.4 Razvrščanje člankov v urnik

Preprosta uporaba metod gručenja ni dovolj za dobro razporeditev člankov. Opisane metode gručenja namreč ne upoštevajo strukture urnika konference, v katerega želimo razporediti članke, in dolžin člankov. Metode vhodne

članke, ki so si med sabo podobni, le združijo v isto gručo. Aplikacija mora te gruče preslikati v skupine člankov tako, da vsaka skupina ustreza določeni časovni rezini urnika.

Da bi to dosegli, smo razvili preprost algoritem, ki z iterativnim gručenjem ustvari skupine, ki ustrezajo strukturi urnika, in vsebujejo podobne članke. Osnovna struktura algoritma je predstavljena na sliki 4.1.

```
1.) Zaženi metodo gručenja, ki jo je izbral uporabnik,
        nad vhodnimi članki
2.) Iz strukture urnika sestavi seznam SLOTS,
        kjer vsak element ustreza eni časovni rezini urnika
3.) Ponavljaj, dokler ni SLOTS prazen seznam:
    4.) Izberi časovno rezino iz SLOTS
            Pri tem najprej izberi večje časovne rezine
    5.) Izberi največjo gručo
            Pri vzporednih časovnih rezinah izberi
            gruče tako, da bo vsaka rezina sestavljena
            iz člankov, ki prihajajo iz različnih gruč
    6.) Iz izbrane gruče izberi članke, ki so si med seboj
            najbolj podobni in ki zapolnijo izbrano časovno
            rezino, in jih dodaj v časovno rezino
        7.) V kolikor taka gruča ne obstaja ponovi gručenje
                s takimi parametri, da bo gručenje vrnilo
                manj gruč.
            Pri metodi affinity propagation ročno združi
                članke.
    8.) Izbrano časovno rezino odstrani iz seznama SLOTS
```

Slika 4.1: Struktura algoritma za razvrščanje člankov.

Algoritem temelji na iterativnem gručenju in je namenoma preprost. Gre namreč za spletno aplikacijo, kar pomeni, da bi bolj napredne metode (kot

so na primer metode gručenja z omejitvami [29]) verjetno zahtevale preveč časa.

Težava je v tem, da je težko ustvariti gruče člankov, ki so si med seboj podobni in točno zapolnijo dane časovne rezine urnika. Predstavitve člankov in časovne rezine na konferencah imajo lahko namreč različne dolžine. To pomeni, da je že sama postavitev člankov v urnik težak problem. Če hočemo zadostiti še podobnosti med članki v isti časovni rezini, je problem še mnogo težji. Naš preprost algoritma sicer ne zagotavlja popolnih rezultatov, vendar deluje dovolj hitro za uporabo v spletni aplikaciji.

4.5 Združevanje gruč

Naša aplikacija želi pri razporejanju člankov v časovne rezine vedno zagotoviti, da so članki v isti časovni rezini izbrani iz iste gruče. Tega ni vedno mogoče doseči le z uporabo ene metode gručenja. Lahko se namreč zgodi, da so vse gruče, ki jih je vrnila metoda gručenja, manjše od izbrane časovne rezine. V tem primeru aplikacija začne s postopkom združevanja gruč.

Postopek se razlikuje glede na metodo gručenja, ki jo je izbral uporabnik. Najlažjo rešitev ponujajo metode, ki kot parameter sprejmejo število izhodnih gruč (te metode so *k-means*, *mini batch k-means* in *hierarchical clustering*). Pri teh metodah aplikacija preprosto zmanjša število izhodnih gruč in na novo požene metodo gručenja. Pri tem ne vključuje člankov, ki so že bili uspešno dodeljeni gručam. Podobno rešitev uporabi tudi pri metodi *mean shift*. Razlika je le v tem, da namesto spremembe števila izhodnih gruč spremeni parameter *bandwidth*, ki vpliva na število izhodnih gruč.

Težava je metoda *affinity propagation*, kjer je rezultat težje dobiti le z spremembo vhodnih parametrov. Pri tej metodi aplikacija ročno združi dve najmanjši gruči, ki vsebujeta še nedodeljene članke. S tem sicer lahko izbere dve gruči, ki si med seboj nista podobni, vendar take gruče večinoma vključujejo majhno število člankov (večinoma le enega ali dva). Zaradi tega opisan postopek združevanja ne bi smel predstavljati težav.

4.6 Vizualizacija rezultatov

Po končanem samodejnem razvrščanju člankov uporabniku prikažemo vizualizacijo rezultatov. To storimo tako, da vsakemu članku priredimo dvodimenzionalne koordinate in ga obarvamo z barvo, ki ustreza njegovi dodeljeni gruči. Če želimo članke predstaviti v dvodimenzionalnem koordinatnem sistemu moramo najprej ustrezno spremeniti njihove atribute. Atributi člankov so njihovi vektorji tf-idf in podatki o ocenjevalcih. Vektorji imajo veliko število dimenzij in niso primerni za vizualizacijo. Pretvorimo jih v dvodimenzionalne vektorje.

Za to pretvorbo smo uporabili dve metodi - **TruncatedSVD** [21] in **t-SNE** [33] (t-distributed stochastic neighbor embedding). Metoda t-SNE je v našem primeru boljša, saj poskuša ohraniti sosednost med originalnimi vektorji in daje dobre rezultate na tako imenovanih *redkih* podatkih (podatki, kjer ima večina komponent vektorja vrednost 0). Taki podatki so pri analizi besedil zelo pogosti in se skoraj vedno pojavijo tudi v naši aplikaciji. Metodo TruncatedSVD smo uporabili zato, ker je metoda t-SNE lahko precej počasna, predvsem kadar kot vhod dobi vektor z velikim številom dimenzij. Zaradi tega najprej uporabimo metodo TruncatedSVD, s katero znižamo število dimenzij na 50 (seveda le, če ima originalni vektor več kot 50 dimenzij). Nato uporabimo t-SNE, ki vrne vektor z dvema dimenzijama.

V aplikaciji je prisotna tudi alternativna metoda vizualizacije. Ta se uporabi, ko vhodni podatki v metode gručenja vsebujejo nizko število dimenzij (pod 10). V takih primerih metodi TruncatedSVD in t-SNE nista najboljša izbira in aplikacija namesto njiju uporabi metodo **PCA** [27]. Prednost metode PCA je v tem, da deluje bolje na gostih podatkih. Res je, da takih podatkov ne dobimo zelo pogosto - to se lahko zgodi le, če uporabnik uporabi gručenje na ključnih besedah. Metodo smo vseeno implementirali, saj v omenjenem primeru deluje bolje kot metodi t-SNE in TruncatedSVD.

Dobljeno vizualizacijo nato posredujemo spletni strani, kjer se uporabniku prikaže vizualizacija. Spletna stran za prikaz vizualizacije uporabi knjižnico javascripta z imenom NVD3 [6].

Poglavje 5

Rezultati samodejnega razvrščanja

5.1 Pregled testne množice podatkov

Za testiranje aplikacije smo imeli na voljo podatke o petih konferencah: *Artificial Intelligence in Medicine* (AIME) 2011 ter 2013, *The Ninth International Conference on Discovery Science* (DS-2006), *Fifth International Conference on Computational Creativity* (ICCC 2014) in *Planning to Learn and Service-Oriented Knowledge Discovery* (PlanSoKD 2011). Gre za manjše znanstvene konference, ki se nanašajo na tematiko informatike, računalništva in strojnega učenja. Vsaka konferenca razen DS-2006 je vsebovala podatke o sprejetih člankih v obliki .xls datoteke izvožene iz sistema EasyChair in podatke o mnenjih recenzentov v obliki .csv datoteke, prav tako izvožene iz sistema EasyChair.

Od naštetih konferenc DS-2006 in PlanSoKS 2011 nista bili zanimivi za podrobnejše testiranje. Konferenca PlanSoKS 2011 je vsebovala zelo majhno število člankov, pri konferenci DS-2006 pa nismo imeli na voljo podatkov o ocenjevalcih člankov. Obe konferenci AIME in konferenca ICCC so vsebovale dovolj podatkov za podrobnejšo analizo. Izmed njih smo se odločili podrobneje testirati podatke iz konferenc AIME 2013 in ICCC 2014.

Ime konference	Število člankov	Število ocenjevalcev
AIME 2011	16	60
AIME 2013	41	57
ICCC 2014	36	47
DS-2006	24	Ni podatka
PlanSoKD 2011	5	16

Tabela 5.1: Primerjava podatkov o konferencah.

Iz tabele 5.1 je razvidno, da smo imeli za testiranje aplikacije na voljo le manjše znanstvene konference s pod 50 članki. To sicer ni predstavljalo težav pri ocenjevanju točnosti gručenja, je pa pomenilo, da nismo mogli preveriti delovanja aplikacije na večjem številu podatkov. Članke bi sicer lahko vnašali ročno, vendar bi to za veliko število člankov (največje konference imajo lahko tudi več kot 10.000 člankov) vzelo ogromno časa.

Če bi pri tako velikih konferencah želeli samodejno razporediti vse članke, bi to naši aplikaciji vzelo veliko časa (metode gručenja imajo visoko časovno zahtevnost, naša aplikacija pa jih med samodejnim razporejanjem požene večkrat). Kljub temu menimo, da bi tudi pri takih konferencah bila koristna. Pri uporabi aplikacije namreč ne zahtevamo, da uporabnik v enem samodejnem razvrščanju razporedi vse članke. To pomeni, da bi v praksi uporabniki verjetno že v naprej razdelili članke v večje skupine in nato z našo aplikacijo članke iz teh skupin grupirali v manjše gruče.

5.2 Primerjava uporabljenih metod gručenja

Pri samodejnem razvrščanju člankov uporabniku ponudimo izbiro med različnimi algoritmi gručenja. Vsak od teh algoritmov ima različne lastnosti za gručenje, zaradi česar uporabniku pustimo možnost izbire. Algoritme smo preizkusili nad testnimi podatki, iz česar smo pridobili sledeče ugotovitve:

K-means in mini batch k-means sta v splošnem članke grupirali v do-

bro ločene skupine. Možnost definiranja števila gruč se je izkazala za koristno, saj je mogoče, da uporabnik že v naprej ve, koliko različnih tematik bo zajemala konferenca in lahko na podlagi tega ustrezno nastavi število gruč. Poleg tega je pri omenjenih metodah preprosto izračunati razdalje člankov od sredine gruče. To aplikaciji omogoča, da pri zapolnjevanju časovnih rezin izbere iz posamezne gruče tiste članke, ki so najbližji sredini gruče in so si torej verjetno najbolj podobni.

Affinity propagation se je izkazal kot dober splošen algoritem, ki lahko brez predhodnega nastavljanja parametrov poišče ustrezne gruče v vseh testnih primerih. Težava algoritma je, da mu je težko spremeniti parametre in ga je zato težko prilagoditi posamezni konferenci.

Hierarchical clustering se je izkazal kot dobra alternativa algoritmu k-means. V praksi deluje hitreje, vendar ne ponuja preprostega izračuna razdalje člankov od sredine njihove gruče. Tako kot k-means uporabniku ponuja nastavitev končnega števila gruč.

Mean shift se je izkazal kot precej slaba izbira. Oblike gruč, ki jih vrača, se zdijo zelo naključne, poleg tega pa mu je težko nastaviti pravilne parametre.

DBSCAN se je izkazal kot slabša verzija algoritma mean-shift. Iz tega razloga tega algoritma v končni verziji aplikacije uporabniku ne ponudimo na izbiro.

Glede porabe časa se je izkazalo, da so si algoritmi med sabo zelo podobni. Pri tem je edina izjema algoritem *affinity propagation*, ki je izrazito hitrejši od ostalih. Razlog tega je v tem, da samodejno razvrščanje člankov večino časa porabi na drugih metodah, kot so na primer iskanje člankov, ki zapolnijo izbrano časovno rezino in združevanje gruč. To razloži hitrost algoritma affinity propagation - to je namreč edini algoritem, ki se pri združevanju gruč ne požene ponovno (namesto tega se gruče združijo ročno). Rezultati hitrosti algoritmov gručenja so na voljo v tabeli 5.2. Rezultate smo dobili tako, da

Algoritem	Povprečen čas izvajanja
Mean shift	4.96s
K-means	4.81s
Mini batch k-means	4.64s
Hierarchical clustering	4.31s
Affinity propagation	1.76s

Tabela 5.2: Primerjava hitrosti algoritmov gručenja.

smo nad testno množico enainštiridesetih člankov s konference AIME 2013 vsako metodo pognali dvajsetkrat in izmerili povprečni čas delovanja.

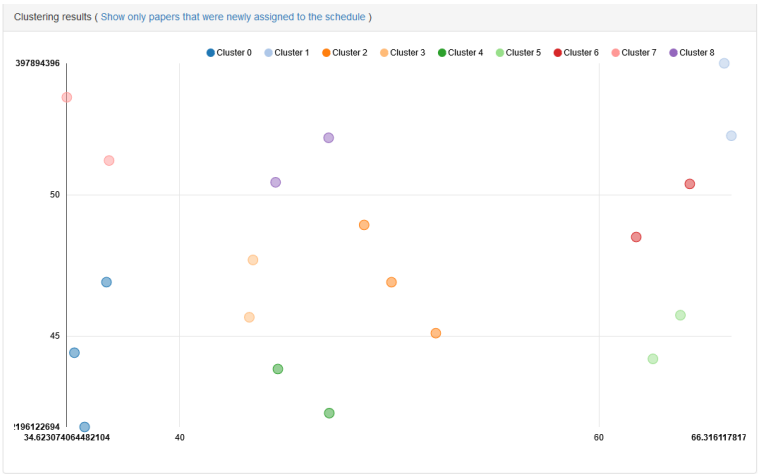
5.3 Analiza rezultatov

Ocenjevanje uspešnosti algoritmov gručenja je bilo težavno. Večina ocen, ki se pogosto uporabijo pri algoritmih gručenja namreč predpostavi, da že v naprej vemo v katere gruče spadajo vhodni podatki. Tega v našem primeru nismo imeli - najbližja stvar, ki bi jo lahko uporabili so dejanski urniki konferenc, iz katerih smo imeli vhodne podatke. Težava je, da razporeditev člankov na dejanski konferenci ni edina razporeditev, ki ustrezno združi članke glede na njihovo podobnost. Poleg tega se članki iz naših testnih primerov ne ujemajo popolnoma s tistimi, ki so bili dejansko predstavljeni na konferenci.

Ker uporaba takih metod za ocenjevanje uspešnosti gručenja ni bila možna, smo gručenje ocenili ročno na dva načina. Najprej smo primerjali vizualizacije različnih algoritmov gručenja, kjer smo iskali očitne napake različnih algoritmov. Nato smo na podlagi tega izbrali najboljšo metodo in jo pognali na primeru, ki je imel isto strukturo urnika in iste članke kot konferenca AIME 2013. Nato smo s pomočjo strokovnjaka s področja konference ročno ocenili, kako uspešno smo grupirali članke in rezultate primerjali z grupiranjem na dejanski konferenci.

5.3.1 Analiza s pomočjo vizualizacije

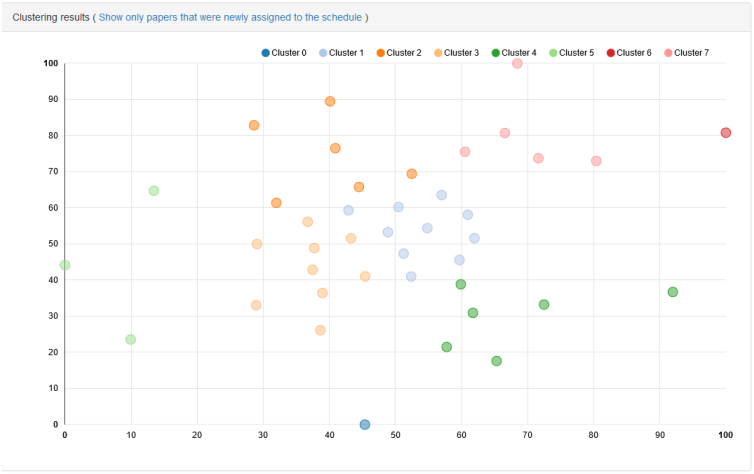
Analiza gručenja s pomočjo vizualizacije rezultatov je zasilna rešitev. Vseeno smo med potekom te analize na nekaterih algoritmih odkrili ključne pomanjkljivosti, zaradi katerih ti algoritmi niso primerni za uporabo. Pomanjkljivosti smo odkrili pri algoritmih *mean shift* in *DBSCAN*. Pri obeh je že iz preprostega pogleda na vizualizacijo razvidno, da vračata slabe rezultate. Oba namreč ali ustvarita veliko število gruč, ki vsebujejo zelo majhno število člankov (le 2 do 3), kot je vidno na sliki 5.1, ali ustvarita le eno veliko gručo z nekaj osamelci (odvisno od nastavitve parametrov). Pri takih gručah deluje samodejno razvrščanja člankov slabo, kar pomeni, da sta omenjena algoritma slaba izbira.



Slika 5.1: Gručenja metode mean shift.

Pri ostalih algoritmih ni vidnih večjih razlik. Vsi razporedijo članke v ustrezno velike gruče z jasnimi mejami, kot na sliki 5.2. Edina težava, ki je pri njih vidna je prisotnost osamelcev - posameznih člankov, ki se nahajajo daleč stran od vseh ostalih. Ker so osamelci prisotni pri vseh algoritmih gručenja je razlog zanje verjetno v podatkovni množici nad katero jih poženemo. To bi se dalo izboljšati z uporabo naprednejših metod analize besedil, ki jih pri izgradnji podatkovne množice zaradi pomanjkanja časa nismo uporabili. Te

izboljšave so podrobneje opisane v razdelku 6.2.



Slika 5.2: Gručenja metode affinity propagation.

5.3.2 Analiza na podlagi mnenja strokovnjaka

	Razmerje ustreznih člankov v gruči	
	Samodejna razporeditev	Dejanska konferenca
Skupina 1	8/11	11/11
Skupina 2	3/6	6/6
Skupina 3	3/3	2/3
Skupina 4	3/5	4/6
Skupina 5	3/5	3/3
Skupina 6	4/5	3/5
Skupina 7	4/5	4/5

Tabela 5.3: Rezultati ocene strokovnjaka.

Po končani analizi s pomočjo vizualizacije rezultatov smo podrobneje analizirali algoritem affinity propagation, saj je ob ponovljenih poskusih dajal

najbolj dosledne rezultate. To smo storili tako, da smo rezultate postopka samodejnega razvrščanja člankov z uporabo algoritma affinity propagation podali strokovnjaku s področja konference, ki je ocenil njihovo kvaliteto. Poleg tega je ocenil tudi postavitev člankov iz dejanskega urnika konference. Strokovnjak pred ocenjevanjem ni vedel, katera razporeditev je bila dejansko uporabljena. Rezultati te analize so podani v tabeli 5.3.

Iz tabele je razvidno, da samodejna razporeditev člankov ni tako dobra kot ročna razporeditev na dejanski konferenci. Težave so predvsem pri velikih skupinah, kjer moramo pri samodejnem razvrščanju združevati gruče, ki nam jih vrnejo algoritmi gručenja. Kljub temu so rezultati še vedno dovolj dobri, da lahko uporabnikom koristijo pri ustvarjanju urnikov konferenc.

5.4 Učinkovitost uporabe ključnih besed

Naša aplikacija uporabniku poleg običajnega gručenja člankov omogoča tudi gručenje glede na skupino vnaprej definiranih ključnih besed. Če se uporabnik odloči za to možnost bo besedilna analiza člankov upoštevala le besede iz skupine ključnih besed. To je zelo specifičen način gručenja in v splošnem ni tako učinkovit kot druge metode gručenja, v določenih primerih pa se izkaže za koristnega. To je predvsem takrat, ko uporabnik dobro pozna večino člankov na konferenci in ve, da jih lahko z uporabo ključnih besed razdeli na dobre skupine.

Primer, kjer je tako razvrščanje koristno je konferenca ICCC 2014. Ta namreč vsebuje časovne rezine, kjer vsi naslovi člankov vsebujejo edinstvene ključne besede. Če pri tej konferenci poženemo gručenje s ključnima besedama *game* in *poetry* dobimo tri ločene gruče, od katerih dve vsebujeta zelo podobne članke. Uporabnik lahko nato nad ostalimi članki požene splošno gručenje.

Poglavje 6

Možne izboljšave

6.1 Uporabniški vmesnik

Uporabniški vmesnik aplikacije je za zdaj precej preprost. Uporabniku sicer ponuja vse potrebne funkcionalnosti, vendar ne vsebuje navodil ali pomoči uporabniku. Smiselno bi bilo na primer dodati opis glavnih značilnosti vsakega od algoritmov gručenja in njihovih parametrov.

Izgled uporabniškega vmesnika je preprost. Temelji na privzetem stilu ogrodja *bootstrap*, kar pomeni, da mu manjka edinstvenosti. Poleg tega je spletni del aplikacije statičen, kar v modernih spletnih straneh ni zaželeno.

6.2 Samodejno razvrščanje člankov

Samodejno razvrščanje člankov trenutno ne deluje najbolje in bi ga lahko izboljšali na več različnih načinov. Pri gradnji podatkovne množice bi lahko uporabili bolj napredne metode besedilne analize, kot so lematizacija (pretvorba besed z istim jedrom vendar različnim sklonom ali množino v isto besedo) in bolj napredna odstranitev besed, ki ne vplivajo na pomen besedila. Poleg tega bi lahko poskusili izboljšati analizo podatkov o ocenah člankov tako, da bi spremenili uteži v grafu, čeprav je brez učinkovitega postopka za oceno kvalitete rezultatov težko oceniti, če bi to res pomagalo.

Množico podatkov bi lahko izboljšali tudi tako, da bi v seznam besed poleg besed dodali še izraze (*term*), ki so lahko sestavljeni iz več kot ene besede. Pravilno izbrani izrazi bi povečali učinkovitost gručenja [16].

Največje izboljšave pri samodejnem razvrščanju člankov so možne pri algoritmu, ki rezultate algoritmov gručenja preslika v proste časovne rezine urnika. Trenutno je ta algoritem preprost in bi ga lahko izboljšali s sledečimi popravki:

Boljše združevanje gruč - Trenutno pri združevanje gruč ne preverjamo podobnosti med gručami, ki jih združimo. Bolje bi bilo, če bi za vsako gručo izračunali koordinate njene sredine in nato združili gruče, ki so si najbolj blizu.

Boljša izbira člankov iz gruč - Ko iz gruče premikamo članke v časovno rezino, bi bilo bolje, če bi izbrali članke, ki so si med sabo najbolj podobni. Trenutno to storimo le pri metodah *k-means* in *mini-batch k-means*. Pri drugih metodah bi lahko ročno izračunali podobnosti med članki in s tem dobili boljše rezultate.

Boljši algoritmi gručenja - Pri gručenju bi lahko uporabili boljše algoritme gručenja, predvsem algoritma *mixture modeling* [32] in *banded matrix clustering* [19], kot tudi njuno kombinacijo [12]. Te algoritmi vračajo dobre rezultate, vendar nismo imeli časa za njihovo implementacijo.

Uporaba gručenja z omejitvami - Namesto trenutnih algoritmov gručenja, bi lahko uporabili gručenje z omejitvami. Gre za skupino algoritmov, ki primere razvrstijo v gruče, pri čimer upoštevajo v naprej definirane omejitve. Problem je, da je področje gručenja z omejitvami še dokaj nerazvito, zaradi česar nam ni uspelo najti ustreznih algoritmov za uporabo na naši domeni.

6.3 Evaluacija rezultatov

Rezultate smo ocenili le na podlagi vizualizacije in mnenja strokovnjaka. Taka evaluacija je preprosta in ne vrača vedno zanesljivih rezultatov. Za bolj zanesljivo evaluacijo bi morali ročno vnesti podatke o gručenju člankov iz več različnih konferenc in njihovo razporeditev primerjati z razporeditvijo samodejnega razvrščanja člankov, pri čimer bi uporabili algoritem *adjusted rand index* [23]. Algoritem med seboj primerja dva različna rezultata gručenja in kot rezultat vrne podobnost med njima.

Tega postopka nismo uporabili zaradi časovnih omejitev. Ročno vnašanje podatkov o konferencah bi vzelo ogromno časa, bi pa ocenjevanje s pomočjo algoritma dalo bolj zanesljive rezultate kot ocenjevanje na podlagi mnenja strokovnjaka.

Poglavje 7

Zaključek

V diplomskem delu smo razvili spletno aplikacijo, ki uporabniku omogoča organizacijo urnika konference in člankov, ki bodo predstavljeni na izbrani konferenci. Preko spletnega vmesnika smo omogočili dostop do sistema, v katerem lahko uporabnik definira strukturo urnika in uvozi ali ročno vnese članke konference, ki jih lahko potem doda v urnik. Preko sistema lahko več uporabnikov hkrati organizira več konferenc, ki so vse shranjene v centralnem strežniku.

Razvili smo postopek, ki samodejno razvrsti članke v urnik glede na podobnost med njimi in predstavlja osrednjo funkcionalnost aplikacije. Uporabnik lahko izbere algoritem gručenja in nastavi njegove parametre ter nato s pritiskom na gumb samodejno razvrsti članke v urnik glede na njihovo podobnost. Za izračun podobnosti smo uporabili podatke o naslovu in povzetku članka ter o grafu mnenj recenzentov člankov. Za implementacijo postopka smo uporabili algoritme s področij besedilne analize, gručenja in analize grafov.

Pri analizi rezultatov je bilo razvidno, da razvit postopek vrača rezultate, ki so dovolj dobri, da lahko koristijo organizatorjem konferenc. Kljub temu bi dobljene rezultate lahko še izboljšali. Uporabili bi lahko bolj izpopolnjene metode besedilne analize in izboljšali algoritem samodejnega razvrščanja. Izboljšali bi lahko tudi spletni vmesnik aplikacije, ki je trenutno preprost.

Menimo, da ustvarjena aplikacija deluje dovolj dobro, da lahko olajša delo organizatorjem konferenc. Pri konferencah, ki vsebujejo več kot 1.000 člankov, bi lahko prišlo do težav s časovno zahtevnostjo algoritma, kar pa lahko rešimo tako, da ne razporejamo vseh člankov hkrati.

Literatura

[1] (2015) About python. Dostopno na: https://www.python.org/about/

[2] (2015) Bootstrap. Dostopno na: http://getbootstrap.com/

[3] (2015) Cascading style sheets. Dostopno na: http://www.w3.org/Style/CSS/Overview.en.html

[4] (2015) Easychair home page. Dostopno na: http://easychair.org/

[5] (2015) Html5. Dostopno na: http://www.w3.org/TR/html5/

[6] (2015) Nv3d. Dostopno na: http://nvd3.org/index.html

[7] (2015) Openconf conference management system and peer-review software. Dostopno na: http://www.openconf.com/

[8] (2015) Past and future annual meetings. Dostopno na: http://www.sfn.org/annual-meeting/past-and-future-annual-meetings/

[9] (2015) PostgreSQL: About. Dostopno na: http://www.postgresql.org/about/

[10] (2015) Society for neuroscience. Dostopno na: http://www.sfn.org/annual-meeting/neuroscience-2015/sessions-and-events

[11] (2015) The web framework for perfectionists with deadlines — django. Dostopno na: https://www.djangoproject.com/

[12] P. R. Adhikari, A. Vavpetič, J. Kralj, N. Lavrač, in J. Hollmén, "Explaining mixture models through semantic pattern mining and banded matrix visualization," v *Discovery Science*, 2014, str. 1–12.

[13] A. Berson, *Client/Server Architecture*. New York: Mcgraw-Hill, 1992.

[14] M. Ester, H.-P. Kriegel, J. Sander, in X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," v *KDD*, zv. 96, št. 34, 1996, str. 226–231.

[15] D. Flanagan, *JavaScript: the definitive guide*. Sebastopol, Calif.: O'Reilly Media, Inc., 2006.

[16] B. Fortuna, N. Lavrač, in P. Velardi, "Advancing topic ontology learning through term extraction," v *Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence*. Springer-Verlag, 2008, str. 626–635.

[17] B. J. Frey in D. Dueck, "Clustering by passing messages between data points," *Science*, zv. 315, št. 5814, str. 972–976, 2007.

[18] K. Fukunaga in L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *Information Theory, IEEE Transactions on*, zv. 21, št. 1, str. 32–40, 1975.

[19] G. C. Garriga, E. Junttila, in H. Mannila, "Banded structure in binary matrices," *Knowledge and Information Systems*, zv. 28, št. 1, str. 197–226, 2011.

[20] M. Grčar in N. Lavrač, "A methodology for mining document-enriched heterogeneous information networks," v *Discovery Science*, 2011, str. 107–121.

[21] P. C. Hansen, "The truncated SVD as a method for regularization," *BIT Numerical Mathematics*, zv. 27, št. 4, str. 534–553, 1987.

[22] J. A. Hartigan in M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Applied statistics*, str. 100–108, 1979.

[23] L. Hubert in P. Arabie, "Comparing partitions," *Journal of classification*, zv. 2, št. 1, str. 193–218, 1985.

[24] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, zv. 32, št. 3, str. 241–254, 1967.

[25] G. E. Krasner, S. T. Pope *et al.*, "A description of the model-view-controller user interface paradigm in the smalltalk-80 system," *Journal of object oriented programming*, zv. 1, št. 3, str. 26–49, 1988.

[26] L. Page, S. Brin, R. Motwani, in T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web." Stanford InfoLab, Stanford, CA, 1999-66, November 1999.

[27] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, zv. 2, št. 11, str. 559–572, 1901.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, zv. 12, str. 2825–2830, 2011.

[29] V. Podpečan, M. Grčar, in N. Lavrač, "Semi-supervised constrained clustering: an expert-guided data analysis methodology," v *PRICAI 2010: Trends in Artificial Intelligence*, 2010, str. 219–230.

[30] J. Ramos, "Using TF-IDF to determine word relevance in document queries," v *Proceedings of the first instructional conference on machine learning*, 2003.

[31] D. Sculley, "Web-scale k-means clustering," v *Proceedings of the 19th international conference on world wide web*, 2010, str. 1177–1178.

[32] J. Tikka, J. Hollmén, in S. Myllykangas, "Mixture Modeling of DNA copy number amplification patterns in cancer," v *Proceedings of the 9th International Work-Conference on Artificial Neural Networks*, zv. 4507. Springer-Verlag, 2007, str. 972–979.

[33] L. Van der Maaten in G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, zv. 9, št. 85, str. 2579–2605, 2008.