Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Borja Bovcon

Analiza podobnosti besedil z metodami razvrščanja

diplomsko delo

interdisciplinarni univerzitetni študijski program
prve stopnje računalništva in matematike

Mentor: doc. dr. Zoran Bosnić

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00027/2013
Datum: 09.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko ter Fakulteta za matematiko in fiziko izdaja naslednjo nalogo:

Kandidat: **BORJA BOVCON**

Naslov: **ANALIZA PODOBNOSTI BESEDIL Z METODAMI RAZVRŠČANJA**
CLUSTERING-BASED ANALYSIS OF TEXT SIMILARITY

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Z rastjo semantičnega spleta se večajo tudi potrebe po zmožnostih za kakovovostno iskanje podobnih dokumentov v velikih dokumentnih zbirkah.
Kandidat naj v diplomski nalogi opravi pregled metod za merjenje podobnosti besedil, ki temeljijo na nenadzorovanem strojnem učenju (razvrščanju). V delu naj preizkusi delovanje različnih mer podobnosti in algoritmov za razvščanje dokumentov. Delovanje naj evalvira na več izbranih testnih dokumentnih zbirkah, v sklepu pa naj poda smernice oz. priporočila za uporabo najboljših kombinacij algoritmov.

Mentor:

doc. dr. Zoran Bosnić

Dekan Fakultete za računalništvo in informatiko:

prof. dr. Nikolaj Zimic

Dekan Fakultete za matematiko in fiziko:

akad. prof. dr. Franc Forstnerič

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Borja Bovcon, z vpisno številko **63100228**, sem avtor diplomskega dela z naslovom:

Analiza podobnosti besedil z metodami razvrščanja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Zorana Bosnića,
- 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 v zbirki "Dela FRI".

V Ljubljani, dne 16. septembra 2013 Podpis avtorja:

Povzetek

Diplomska naloga se ukvarja z metodami za merjenje podobnosti besedil z metodami razvrščanja. Začnemo s predstavitvijo problema in opisom dveh najpogosteje uporabljenih načinov reprezentacije podatkovnih množic besedil. V istem poglavju spoznamo tudi načine filtriranja in njihovo pomembnost, Porterjev algoritem ter algoritem tf-idf, namenjen uteževanju izrazov. Vse to apliciramo na izbrane podatkovne množice različnih velikosti in gostot. S pomočjo algoritmov razvrščanja in različnih mer razdalj, kategoriziramo besedila podatkovne množice v grozde. Delo zaključimo z evalvacijo in analizo rezultatov ter sklepom, v katerem izberemo kombinacijo najprimernejših opisanih metod za primerjavo besedil.
Ključne besede: razvrščanje, besedila, tekstovni dokumenti, analiza, kmeans, kmeans++, kmedoids++, mere razdalj, podobnost, primerjava

Abstract

The focus of this thesis is comparison of analysis of text-document similarity using clustering algorithms. We begin by defining main problem and then, we proceed to describe the two most used text-document representation techniques, where we present words filtering methods and their importance, Porter's algorithm and tf-idf term weighting algorithm. We then proceed to apply all previously described algorithms on selected data-sets, which vary in size and compactness. Fallowing this, we categorize documents in different clusters using clustering algorithms and similarity/distance measures. In final chapter we evaluate obtained clusters and analyse results of evaluation. As a conclusion, we hand-pick the best possible combination of described methods for determining text-document similarity.
Keywords: clustering, text, documents, analysis, kmeans, kmeans++, kmedoids++, distance measures, similarity, dissimilarity, comparison

Zahvala

Zahvaljujem se mentorju doc. dr. Zoranu Bosniću za usmerjanje, pomoč in nasvete pri pisanju diplomske naloge.

Za tehnične nasvete pri uporabi oradja LaTeX se zahvaljujem prijatelju in sošolcu Blažu Sovdatu.

Za posredno pomoč pri pisanju diplome pa se zahvaljujem staršema.

Kazalo

Poglavje 1

Uvod

1.1 Predstavitev problema

Hiter razvoj komunikacijskih tehnologij v zadnji letih je omogočil ljudem ustvarjati in izmenjevati velike količine informacij, še posebej v tekstovni obliki. Večina ustvarjenih in objavljenih besedil ni označena z vnaprej podanim naborom kategorij, saj je ljudem nemogoče slediti hitro naraščujočim informacijam. Lahko pa si pri tem pomagamo z algoritmi za razvrščanje. Razvrščanje dokumentov (angl. document clustering) je avtomatičen proces organizacije besedil v skupine, glede na njihovo podobnost, kar vključuje tudi pripadajočo temo. Razvrščanje se najpogosteje uporablja

- v iskalnikih (angl. search engines), za prikaz rezultatov, ločenih po kategorijah,
- pri analizi uporabnikovega zanimanja,
- pri analizi primerjalne književnosti, ...

1.2 Struktura diplomske naloge

Pri razvrščanju dokumentov imamo na razpolago izbrati različne implementacije naslednjih postopkov:

- Izbira reprezentacije tekstovnih dokumentov: Uporabili bomo dve različni metodi reprezentacije tekstovnih dokumentov, ki sta *vreča besed* in *bigrami*. Pri obeh metodah predstavimo tekstovni dokument kot vektor, katerega komponente ustrezno utežimo z opisanim postopkom.
- Izbira mere podobnosti: Ocena podobnosti temelji na medsebojni razdalji elementov. Za računanje razdalje elementov obstaja več mer. V tem poglavju je predstavljenih sedem mer razdalj — *Evklidska razdalja, kosinusna razdalja,*

Pearsonov korelacijski koeficient, Bray-Curtisova neenakost, Jeffreyjeva divergenca, razdalja Canberra in *Hellingerjeva razdalja.*

- Izbira metode razvrščanja: V tem poglavju so opisani trije algoritmi za razvrščanje — *KMeans, KMeans++* in *KMedoids++*. Poleg delovanja, so predstavljene tudi njihove omejitve, slabosti in časovne zahtevnosti.

Poglavje 2

Reprezentacija tekstovnih dokumentov

2.1 Uvod

Velik izziv pri razvrščanju tekstovnih dokumentov je izbira modela, s katerim bomo tekstovni dokument predstavili. Obstaja več različno zahtevnih modelov, vendar v praksi večinoma uporabljamo model *vreče besed* (angl. bag of words). V tem poglavju bomo, poleg *vreče besed*, opisali še predstavitev z *bigrami*.

2.2 Določanje pomena atributov

2.2.1 Vreča besed

Vreča besed je eden izmed najpogosteje uporabljenih modelov prav zaradi njegove preprostosti, kar pa ne pomeni, da je slabši od njemu podobnih, računsko zahtevnejših modelov [CB05].
Predpostavimo, da imamo množico tekstovnih dokumentov z n elementi, ki jo označimo s T. Označimo posamezen tekstovni dokument iz množice T z d_i. Sedaj sestavimo slovar D (angl. dictionary), moči m, v katero dodamo vse besede t_j, ki se pojavijo v naši množici tekstovnih dokumentov T. V D so vsi elementi paroma različni, kljub temu pa D ne ustreza matematični definiciji množice, saj so njeni elementi oštevilčeni. Število elementov množice D določa dimenzijo vektorjev izrazov $\vec{t_i}$, ki jih priredimo tekstovnim dokumentom d_i. Vektor izrazov $\vec{t_i}$ nam pove, kolikokrat se pojavi posamezna beseda t_j znotraj dokumenta d_i. Pojavitev besede t_j znotraj dokumenta d_i zapišemo kot funkcijo $tf(d_i, t_j)$, torej vektor izrazov za tekstovni dokument d_i zapišemo kot:

$$\vec{t_i} = (tf(d_i, t_1), tf(d_i, t_2), \cdots, tf(d_i, t_m)) \tag{2.1}$$

Na ta način lahko sklepamo, katere besede so pomembnejše od ostalih in katere zaznamujejo/ločijo dokument d_i od ostalih dokumentov.

Primer delovanja

Vzemimo množico tekstovnih dokumentov T, moči 2, ki jo sestavljata tekstovna dokumenta:

- d_1: *"To be, or not to be, that is the question."*
- d_2: *"Answer those questions."*

Pripadajoča vreča besed D moči 12 se glasi:

$$\begin{aligned} D = \{&To\,[1]\,, be, [2]\,, or\,[3]\,, not\,[4]\,, to\,[5]\,, that\,[6]\,, \\ &is\,[7]\,, the\,[8]\,, question.\,[9]\,, Answer\,[10]\,, \\ &those\,[11]\,, questions.\,[12]\} \end{aligned} \tag{2.2}$$

ter ustrezna vektorja izrazov:

$$\vec{t_1} = (1, 2, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0) \tag{2.3}$$

$$\vec{t_2} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1) \tag{2.4}$$

Iz primera lahko vidimo, da to ne deluje, kot bi si želeli. Beseda "to" se pojavi dvakrat zaradi velike začetnice, poleg besede "be" nastopa tudi ločilo — vejica, prav tako se dvakrat pojavi beseda "question", ker je zapisana enkrat v ednini, drugič pa v množini. Da bi se takšnim in podobnim problemom izognili, moramo tekstovne dokumente ustrezno filtrirati.

2.2.2 n-Gram

Tekstovni dokument lahko predstavimo tudi z množico *n-gram*-ov — to so sekvence n-tih elementov, ki so lahko:

- črke,
- zlogi,
- besede,
- fonemi.

V našem primeru bo *n-gram* sestavljen iz sekvence besed. V primeru, da bi izbrali $n = 1$, bi dobili t.i. *unigram* in s tem klasično reprezentacijo vreče besed. Mi si bomo podrobneje ogledali delovanje *bigram*-ov (*n-gram* z vrednostjo $n = 2$), katerih učinkovitost bomo tudi testirali.
Vzemimo množico n-tih tekstovnih dokumentov, ki jo označimo s T. Posamezen tekstovni dokument iz množice T označimo z d_i, kjer $i \in [1, n]$. Izgradnjo množice *bigram*-ov za tekstovni dokument d_i si lahko predstavljamo kot prekrivanje besedila z oknom, skozi katerega lahko vidimo le dve sosednji besedi. Dobljene pare besed shranimo v množico M_i. Končna množica *bigram*-ov M je unija posameznih množic M_i.

$$M = \bigcup_{i=1}^{n} M_i \tag{2.5}$$

Primer delovanja

Vzemimo tekstovni dokument d_1 z besedilom:

- *"To be, or not to be, that is the question."*

in zgradimo zanj množico *bigram*-ov:

$$\begin{aligned} M_i = \{&Tobe\,[1]\,, be, or\,[2]\,, ornot\,[3]\,, \\ &notto\,[4]\,, tobe,\,[5]\,, be, that\,[6]\,, \\ &thatis\,[7]\,, isthe\,[8]\,, thequestion.\,[9]\} \end{aligned} \tag{2.6}$$

Sedaj lahko tekstovni dokument d_1 zapišemo z vektorjem dolžine 9:

$$\vec{t_1} = (1, 1, 1, 1, 1, 1, 1, 1, 1) \tag{2.7}$$

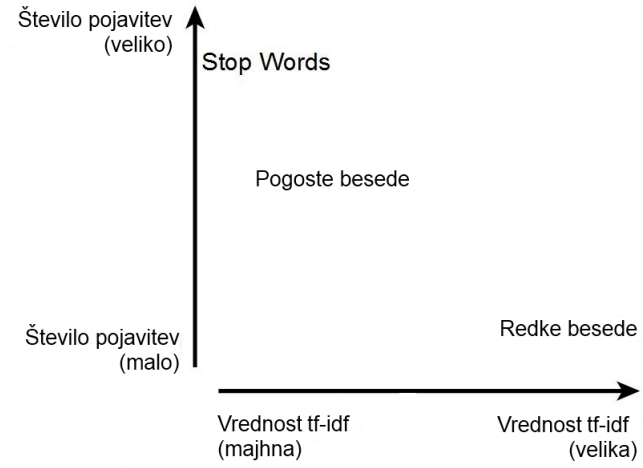
Za poljubno besedilo, dolgo n besed, dobimo množico *bigram*-ov moči $n - 1$ ali manj. Moč manj od $n - 1$ dobimo le v primeru, ko se *bigram*-i v besedilu ponavljajo.

2.3 Določanje vrednosti atributov

Bodisi, če z atributi predstavimo *bigrame* ali pa dokument predstavimo kot *vrečo besed*, moramo za obe reprezentaciji izračunati tudi vrednosti atributov.

Za vsako besedo ali besedno zvezo lahko izračunamo njeno frekvenco znotraj tekstovnega dokumenta, v katerem se pojavi. Na podlagi teh frekvenc lahko besedam priredimo ustrezno utež in s tem ločimo pomembne besede od manj pomembnih. Manjšo utež kot ima beseda, manj je pomembna in s tem ima tudi manjši vpliv

na računanje podobnosti dokumentov. Eden izmed najprimernejših algoritmov za uteževanje besed je algoritem *tf-idf (term frequency - inverse document frequency)*. Idejna zasnova mere *tf-idf* je prikazan na sliki 2.1.



Slika 2.1: Odvisnost frekvence pojavitev besede od vrednosti mere *tf-idf*

Mero tf-idf lahko razčlenimo na tri dele:

1. računanje frekvence besede,
2. računanje frekvence dokumenta,
3. računanje obratne (inverzne) frekvence dokumenta.

Računanje frekvence besede (TF)

Z računanjem frekvence besede ugotovimo, katere besede se pogosto pojavljajo znotraj posameznega tekstovnega dokumenta. Naša predpostavka je, da pogoste besede opisujejo in določajo temo tekstovnega dokumenta, in na podlagi te predpostavke lahko kasneje klasificiramo različne tekstovne dokumente v razrede.
Na računanje frekvence vpliva dolžina tekstovnega dokumenta, saj:

- se v kratkih besedilih določena iskana beseda ne pojavi dovolj pogosto — dobi manjšo utež, kot si jo zasluži,
- v daljših besedilih pride do šuma, ko se določene naključne besede pojavljajo pogosto — besede, ki se ne nanašajo na temo, dobijo veliko utež.

Prav to je razlog, da frekvenco besede normaliziramo. To lahko naredimo na več različnih načinov, vendar se najpogosteje uporabljata:

- logaritmično skaliranje,
- razširjena frekvenca.

Med njima prevladuje uporaba logaritmičnega skaliranja, zato jo bomo tudi uporabili. Njeno delovanje je sledeče:
Vzemimo poljubno besedo b, ki nastopa v tekstovnem dokumentu d. Njeno frekvenco izračunamo po formuli:

$$1 + \log_{10} tf_{b,d} \tag{2.8}$$

kjer je $tf_{b,d}$ število vseh pojavitev besede b v tekstovnem dokumentu d.

Računanje frekvence dokumenta (DF)

Ta komponenta nam pove število dokumentov d, v katerih nastopa beseda b. Označimo jo z df_b. Matematično lahko definicijo zapišemo kot:

$$d \in D : b \in T \tag{2.9}$$

kjer je D množica vseh dokumentov in T množica vseh besed. Besede, ki se pojavijo v majhnem številu besedil (takim besedam rečemo, da so redke), nosijo več informacije kot ostale, zato jim tudi priredimo večjo utež, saj bomo na podlagi njih kasneje besedila ločili v razrede.

Računanje obratne (inverzne) frekvence dokumentov (IDF)

Za izračun komponente IDF moramo poznati število dokumentov v zbirki, ki ga označimo z $|D|$ in frekvenco dokumenta za besedo b, kar označimo z df_b. Formula se glasi:

$$idf = \log_{10} \frac{|D|}{df_b} \tag{2.10}$$

kjer je osnova logaritma lahko poljubna, saj logaritem uporabimo le kot orodje za glajenje funkcije. Večja kot je vrednost komponente IDF, bolj je beseda pomembna. Z večanjem števila dokumentov v zbirki pridobiva IDF komponenta na vplivu, saj ločuje dokumente s pomembnimi besedami od dokumentov z nepomembnimi besedami.

Skupna mera tf-idf

Če vse tri dele združimo, dobimo končno formulo za izračun uteži besede b dokumenta d s tf-idf:

$$w_{b,d} = (1 + \log_{10} tf_{b,d}) \cdot \log_{10} \frac{|D|}{df_b} \tag{2.11}$$

kjer je $w_{b,d}$ vedno pozitivna.
V okolju R [pro] utežimo izraze in ustvarimo matriko vektorjev izrazov z ukazoma:

$$\begin{aligned} data.tfidf \leftarrow & DocumentTermMatrix(procZbirka, control = \\ & list(weighting = weightTfIdf)); \\ m \leftarrow & as.matrix(data.tfidf); \end{aligned} \tag{2.12}$$

kjer je spremenljivka *procZbirka* podatkovna zbirka z filtriranimi besedili.
Za uteževanje izrazov in izgradnjo matrike vektorjev izrazov *bigramov* uporabimo ukaza:

$$\begin{aligned} BigramTokenizer \leftarrow & function(x) NGramTokenizer(x, \\ & Weka\_control(min = 2, max = 2)); \\ data.tfidf \leftarrow & DocumentTermMatrix(procZbirka, control = \\ & list(tokenize = BigramTokenizer, weighting = weightTfIdf)); \end{aligned} \tag{2.13}$$

Pri tem moramo predhodno naložiti knjižnico *RWeka* [RWe].

Poglavje 3

Filtriranje tekstovnih dokumentov

V tekstovnih dokumentih nastopajo tudi besede, ki ne povedo nič o temi in vsebini besedila. Primer takšnih besed so vezniki, mašila, nepolnopomenske besede, itd. Ne samo, da takšni tipi besed ne pripomorejo k ustrezni klasifikaciji besedila, temveč tudi povečajo dimenzijo vektorja izrazov tekstovnega dokumenta in s tem povečajo časovno zahtevnost klasifikacijskega algoritma. Temu se lahko izognemo, če besedilo filtriramo, preden zgradimo zanj reprezentativni vektor izrazov. S filtriranjem iz tekstovnega dokumenta izluščimo le besede, za katere mislimo, da opisujejo temo in vsebino besedila. V nadaljevanju opisujemo uporabljeno programsko opremo za izvedbo filtriranja in postopek filtriranja s parametri.

3.1 Uporabljene knjižnice

Za filtriranje bomo uporabili knjižnici:

- TM *(Text Mining)* za programski jezik *R*.
- *BeautifulSoup* za programski jezik *Python*.

Knjižnica TM *(Text Mining)*

Knjižnica *TM* [CBKM+] je namenjena podatkovnemu rudarjenju. Ponuja nam možnosti:

- uvoza podatkov — Glavna struktura za delo z dokumenti je t.i. korpus (angl. corpus), zanj obstaja znotraj *TM*, več različnih implementacij. Privzeta implementacija je t.i. *VCorpus* (volatile corpus), katero bomo tudi uporabili. Posebnost *VCorpus*-a je, da zbirko dokumentov v celoti preberemo v objekt znotrja *R* in z njo delamo, samo dokler obstaja ta objekt. Takšen korpus lahko ustvarimo iz zbirke podatkov s pomočjo ukaza: Corpus(DirSource('pot-do-podatkovne-zbirke')), pri čemer morajo biti dokumenti v enem izmed naslednjih tipov formatov: .txt, .pdf, .xml, .doc.

- upravljanja zbirk podatkov — Nad tekstovnimi dokumenti v korpusu lahko izvajamo različne transformacije. To počnemo s funkcijo *tm_map()*, ki prejme dva parametra — ime zbirke, na kateri bomo izvedli transformacijo, in tip transformacije. S pomočjo funkcije *tm_filter()* lahko iz zbirke podatkov odstranimo želene dokumente na podlagi meta podatkov.

- predstavitve korpusa z matriko vektorjev izrazov — To je najpogosteje uporabljen pristop k podatkovnemu rudarjenju. Nad matriko vektorjev izrazov lahko izvajamo različne operacije — npr. poiščemo najpogosteje uporabljene besede, poiščemo asociacije na poljubno besedo, ... V splošnem so matrike vektorjev izrazov velikih dimenzij. Ta problem lahko delno odpravimo s filtriranjem, uporabo funkcije *removeSparseTerms()* in uporabo slovarja. Slovar je množica besed, za katere menimo, da so pomembne. Lahko ga uporabimo pri izdelavi matrike vektorjev izrazov, pri čemer se bodo upoštevale le besede iz slovarja — ostale se spregleda.

Knjižnica *BeautifulSoup*

BeautifulSoup [Bea] je knjižnica za programski jezik *Python*. Namenjena je obravnavanju datotek tipa *HTML* in *XML*. Za izgradnjo razčlenitvenega drevesa (angl. parse tree), moramo na prebrani datoteki izvesti funkcijo *BeautifulSoup()*. Po razčlenitvenem drevesu se lahko navigiramo z uporabo metode *find_all()*, ki sprejme za parameter značko *HTML*, katere vsebino iščemo. Iz vozlišča, v katerem se trenutno nahajamo, se lahko premaknemo s pomočjo klica atributov *.parent, .next_sibling, .previous_sibling, .contents, .children, .next_element, .previous_element, ...* Če želimo pridobiti le besedilo *HTML* ali *XML* datoteke, potem uporabimo metodo *get_text()*. V dobljenem besedilu lahko nastopajo posebni znaki različnih abeced, ki jo lahko odstranimo s spremembo kodiranja znakov in uporabo metode *smart_str*.

3.2 Postopek filtriranja

Filtriranje podatkov e-sporočil

Za ekstrakcijo besedila iz e-sporočila smo uporabili *Python* skripto. Skripta prebere tekstovno datoteko oblike e-sporočila, nato pa z uporabo ukaza

$$email.message\_from\_file(datoteka) \tag{3.1}$$

razčlenimo njeno vsebino in z ukazom

$$get\_payload() \tag{3.2}$$

izberemo le naslov in telo e-sporočila.

Filtriranje značk *HTML*

Za odstranitev značk *HTML* iz prebrane tekstovne datoteke smo uporabili *Python* skripto z knjižnico *BeautifulSoup*. Glavna ukaza za doseg želenega sta:

$$get\_text() \tag{3.3}$$

$$smart\_str() \tag{3.4}$$

Filtriranje znakov

Kot smo opazili na primeru iz razdelka 2.2.1, obravnavamo isti besedi, zapisani z različno velikostjo črk, kot dve različni besedi. To pomankljivost lahko odpravimo tako, da spremenimo zapis vseh besed v zgolj male tiskane črke. V R naredimo želeno z ukazom:

$$korpus \leftarrow tm\_map(korpus, tolower) \tag{3.5}$$

Nato iz besedila odstranimo vsa ločila, saj smo že na primeru iz razdelka 2.2.1 spoznali enega izmed problemov, na katerega naletimo sicer. To izvedemo z ukazom:

$$korpus \leftarrow tm\_map(korpus, removePunctuation) \tag{3.6}$$

Nakar odstranimo iz besedila še vsa števila, saj ne vplivajo na vsebino:

$$korpus \leftarrow tm\_map(korpus, removeNumbers) \tag{3.7}$$

V tekstovnem dokumentu se lahko pojavijo tudi posebni znaki, ki ne sodije ne med ločila in ne med števila. Takšne znake imenujemo prazni zanki (angl. whitespaces) in v R jih lahko izločimo z ukazom:

$$korpus \leftarrow stripWhitespace(korpus) \tag{3.8}$$

oziroma

$$korpus \leftarrow tm\_map(korpus, stripWhitespace) \tag{3.9}$$

Filtriranje besed

Iz besedila želimo izločiti čimveč besed, ki ne vplivajo na temo in vsebino. V veliki meri so to z angleško besedo imenovane *stopwords* besede. Odstranimo jih lahko s pomočjo ukaza:

$$korpus \leftarrow tm\_map(korpus, removeWords, stopwords('english')) \tag{3.10}$$

$$korpus \leftarrow tm\_map(korpus, removeWords, stopwords('SMART')) \tag{3.11}$$

kjer prvi ukaz odstrani besede iz seznama *Snowball*, drugi ukaz pa odstrani besede iz seznama *SMART*. Knjižnica *TM* ima več različnih seznamov *stopwords* besed, ki jih delimo na tri skupine:

- *SMART* [sma] seznam angleških besed,
- *Catalan* [Sup10] seznam katalonskih besed,
- seznam projekta *Snowball* [Porb], ki podpira angleške, italijanske, finske, nemške, nizozemske, francoske, švedske, ruske, španske, portugalske, madžarske in avstrijske besede.

Ker so izbrane podatkovne množice besedila v angleškem jeziku, je naša izbira omejena na seznama *Snowball* z 218 besedami in *SMART* z 517 besedami. Uporabili bomo seznam *SMART*, saj vsebuje več besed.
Na primeru iz razdelka 2.2.1 smo videli, da se "question" in "questions" obravnavata kot dve različni besedi, čeprav je njun pomen enak. Temu problemu se izognemo, če vse besede skrčimo na njihov koren. Vzemimo za primer besede *"fish", "fishing", "fisher", "fished"*, katere skrčimo na koren *"fish"* in iz štirih besed dobimo en izraz. S pomočjo knjižnice *TM* lahko besede skrčimo na koren z ukazom:

$$korpus \leftarrow tm\_map(korpus, stemDocument, language = "english") \tag{3.12}$$

Pri tem je za krčenje besed uporabljen Porterjev algoritem.

Porterjev algoritem

Porterjev algoritem je bil prvič objavljen leta 1980, kljub temu pa je še zmeraj eden izmed najpogosteje uporabljenih algoritmov za krčenje besed in to prav zaradi njegovega razmerja med hitrostjo in učinkovitostjo.
Sestavljen je iz ene iteracije šestih korakov, ki skupaj vsebujejo skoraj 60 pravil. Koraki se delijo na:

1. Znebi se pripone množine ("-s") ter pripon "-ed" in "-ing".
2. Spremeni končnico "y" v "i", če je v besedi prisoten samoglasnik.
3. Spremeni dvojno pripono v enojno (npr.: "optional" $\rightarrow$ "option").
4. Obravnavaj pripone tipa "-full", "-ness", $\cdots$
5. Odstrani pripone "-ant", "-ence", $\cdots$
6. Odstrani morebitno končnico "-e".

Lahko se zgodi, da dobimo za koren nekaj, kar sploh ni prava beseda, vendar je to razumljivo, saj algoritem ne krči besed na morfeme.

Poglavje 4

Mere podobnosti

V tem poglavju si bomo ogledali sedem različnih mer podobnosti, s pomočjo katerih določimo razdaljo med dvema tekstovnima dokumentoma. Uvodoma pojasnimo osnovno terminologijo:

- **Podobnost (angl. similarity)** - mera bližine med dvema objektoma. Bliže kot sta si objekta, večja je mera podobnosti.
- **Neenakost (angl. dissimilarity)** - mera, ki določa, kako različna sta si dva objekta. Bolj kot se razlikujeta, večja je mera neenakosti.

Mero neenakosti lahko pretvorimo v mero podobnosti in obratno. Predpostavimo, da imamo mero podobnosti (označimo z $s(x, y)$), katere vrednosti so iz intervala $[0, 1]$. To mero lahko pretvorimo v mero podobnosti (označeno z $d(x, y)$) na enega izmed sledečih načinov:

- $d(x, y) = 1 - s(x, y)$
- $d(x, y) = \sqrt{1 - s(x, y)}$

V splošnem lahko uporabimo katerokoli monotono padajočo transformacijo za pretvorbo mere podobnosti v mero neenakosti in katerokoli monotono naraščajočo transformacijo za pretvorbo mere neenakosti v mero podobnosti.
Metrika je posebna oblika mere neenakosti med dvema objektoma. Če želimo, da je mera metrika, mora ustrezati sledeči definiciji:

Definicija 1. *Označimo z x in y dva poljubna objekta, ki pripadata množici M ter z $d(x, y)$ razdaljo med tema dvema objektoma. Mera d bo metrika natanko tedaj, ko bo izpolnjevala naslednje štiri pogoje:*

1. *Je vedno nenegativna, t.j.* $d(x, y) \geq 0$.
2. *Je enaka* 0 *natanko tedaj, ko sta objekta identična, t.j.* $d(x, y) = 0 \Leftrightarrow x = y$.
3. *Je simetrična, t.j.* $d(x, y) = d(y, x)$.
4. *Zadostuje trikotniški neenakosti, t.j.* $d(x, z) \leq d(x, y) + d(y, z)$.

V nadaljevanju podajamo podrobni opis sedmih mer podobnosti.

4.1 Evklidska razdalja

Evklidska razdalja ustreza vsem štirim pogojem metrike, poleg tega pa je tudi računsko relativno preprosta. Ravno zato je velikokrat uporabljena v algoritmih, ki so namenjeni razvrščanju besedil - nastavljena je kot privzeta metrika algoritma K-Means.

Recimo, da imamo podana dokumenta d_a in d_b, katera predstavimo z ustreznima vektorjema terminov $\vec{t_a}$ in $\vec{t_b}$, dolžine m. Potem bo enačba za evklidsko razdaljo:

$$D_{euc}(\vec{t_a}, \vec{t_b}) = \sqrt{\sum_{t=0}^{m} (w_{t,a} - w_{t,b})^2} \tag{4.1}$$

Kjer je:

$$w_{t,a} = tfidf(d_a, t) \tag{4.2}$$
$$w_{t,b} = tfidf(d_b, t) \tag{4.3}$$

Enake oznake bomo uporabili tudi v nadaljevanju.

4.2 Kosinusna podobnost

Recimo, da imamo podana vektorja terminov $\vec{t_a}$ in $\vec{t_b}$. Pri kosinusni podobnosti opazujemo kot med vektorjema, ki predstavljata besedili. Manjši kot je kot, bolj sta si besedili podobni. Če je kot med vektorjema enak 0, potem sta besedili identični, pri 1 pa sta si popolnoma različni. Posebnost kosinusne podobnosti je, da dolžina vektorja ne vpliva na podobnost. Enačba za računanje kosinusne podobnosti se glasi:

$$sim_{cos}(\vec{t_a}, \vec{t_b}) = \frac{\sum_{t=0}^{m} w_{t,a} \cdot w_{t,b}}{\sqrt{\sum_{t=0}^{m} w_{t,a}^2} \cdot \sqrt{\sum_{t=0}^{m} w_{t,b}^2}} \tag{4.4}$$

ker pa je to mera podobnosti, ki zavzame vrednosti iz intervala $[0, 1]$, jo moramo še pretvoriti v mero razdalje:

$$D_{cos}(\vec{t_a}, \vec{t_b}) = 1 - \frac{\sum_{t=0}^{m} w_{t,a} \cdot w_{t,b}}{\sqrt{\sum_{t=0}^{m} w_{t,a}^2} \cdot \sqrt{\sum_{t=0}^{m} w_{t,b}^2}} \tag{4.5}$$

Vendar kosinusna podobnost ne ustreza popolnoma definiciji metrike.

Primer: Vzemimo dokument d. Če dokument d kombiniramo in s tem ustvarimo nov dokument d', potem velja:

$$sim_{cos}(\vec{t_d}, \vec{t_d'}) = 1 \tag{4.6}$$

Prav tako velja tudi:

$$sim_{cos}(\vec{t_l}, \vec{t_d}) = sim_{cos}(\vec{t_l}, \vec{t_d'}) \tag{4.7}$$

kar pa je v protislovju z našo definicijo metrike, saj dokumenta d in d' nista identična. V praksi to sicer ne povzroča težav, saj delamo z normiranimi vektorji.

4.3 Pearsonov korelacijski koeficient

Enačba za izračun Pearsonovega koeficienta:

$$sim_{per}(\vec{t_a}, \vec{t_b}) = \frac{\sum_{t=0}^{m} (w_{t,a} - \bar{w_{t,a}}) \cdot (w_{t,b} - \bar{w_{t,b}})}{\sqrt{\sum_{t=0}^{m} (w_{t,a} - \bar{w_{t,a}})^2} \cdot \sqrt{\sum_{t=0}^{m} (w_{t,b} - \bar{w_{t,b}})^2}} \tag{4.8}$$

Tudi Pearsonov koeficient je mera podobnosti. V mero razdalje jo lahko pretvorimo s sledečim postopkom:

$$D_{per}(\vec{t_a}, \vec{t_b}) = \begin{cases} 1 - sim_{per}(\vec{t_a}, \vec{t_b}) & \text{če } sim_{per} \geq 0 \\ \left| sim_{per}(\vec{t_a}, \vec{t_b}) \right| & \text{če } sim_{per} < 0 \end{cases} \tag{4.9}$$

4.4 Bray-Curtisova neenakost

Bray-Curtisova neenakost krši pogoj trikotniške neenakosti metrike. Kljub temu deluje v določenih primerih dobro. Vrednosti vrača iz intervala $[0, 1]$, kjer 0 pomeni, da sta vektorja identična, 1 pa, da se popolnoma razlikujeta.
Enačba za računanje:

$$des_{bc}(\vec{t_a}, \vec{t_b}) = \frac{\sum_{t=0}^{m} w_{t,a} - w_{t,b}}{\sum_{t=0}^{m} w_{t,a} + w_{t,b}} \tag{4.10}$$

4.5 Jeffreyjeva divergenca

Jeffreyjeva divergenca temelji na Kullback-Leiblerjevi (okrajšava: KL) razdalji.
Enačba KL razdalje:

$$d_{kl}(\vec{t_a}, \vec{t_b}) = \sum_{t=0}^{m} w_{t,a} \cdot \log\left(\frac{w_{t,a}}{w_{t,b}}\right) \tag{4.11}$$

KL divergenca v splošnem ni metrika, saj ne ustreza pogoju simetričnosti. Jeffreyeva divergenca to pomankljivost odpravi in jo spremeni v popolno metriko. Enačba Jeffreyjeve divergence:

$$d_{jd}(\vec{t_a}, \vec{t_b}) = d_{kl}(\vec{t_a}, \vec{t_b}) + d_{kl}(\vec{t_b}, \vec{t_a}) \tag{4.12}$$

$$d_{jd}(\vec{t_a}, \vec{t_b}) = \sum_{t=0}^{m} w_{t,a} \cdot \log\left(\frac{w_{t,a}}{w_{t,b}}\right) + \sum_{t=0}^{m} w_{t,b} \cdot \log\left(\frac{w_{t,b}}{w_{t,a}}\right) \tag{4.13}$$

4.6 Razdalja Canberra

Podobna je Manhattanski razdalji, ki tudi izhaja iz posebne vrste razdalje Minhkovskega. Prvotna razdalja Canberra je bila namenjana računanju samo s strogo pozitivnimi števili. Po modifikaciji sprejme tudi negativna števila in vrne enoten rezultat. Zelo je občutljiva na vrednosti, ki so blizu 0, za razliko od Manhattanske razdalje pa ni občutljiva na velika števila.
Enačba razdalje Canberra:

$$d_{can}(\vec{t_a}, \vec{t_b}) = \sum_{t=0}^{m} \frac{|w_{t,a} - w_{t,b}|}{|w_{t,a}| + |w_{t,a}|} \tag{4.14}$$

4.7 Hellingerjeva razdalja

Hellingerjeva razdalja vrača vrednosti iz intervala $[0, 1]$. Lahko jo izrazimo s pomočjo Bhattacharyya-eve razdalje, ki je za diskretno porazdelitev definirana kot:

$$d_{bc}(\vec{t_a}, \vec{t_b}) = -\ln\left(BC(\vec{t_a}, \vec{t_b})\right) \tag{4.15}$$

$$BC(\vec{t_a}, \vec{t_b}) = \sum_{t=0}^{m} \sqrt{w_{t,a} \cdot w_{t,b}} \tag{4.16}$$

iz česar sledi tudi enačba Hellingerjeve razdalje:

$$d_{hel}(\vec{t_a}, \vec{t_b}) = \sqrt{1 - BC(\vec{t_a}, \vec{t_b})} \tag{4.17}$$

Poglavje 5

Metode razvrščanja

5.1 Uvod

Razvrščanje je v podatkovnem rudarjenju temeljni postopek združevanja podobnih elementov v skupine (angl. cluster). Za razvrščanje lahko uporabimo različne algoritme, ki se razlikujejo v načinu iskanja skupin. Za učinkovito ločitev skupin želimo razdaljo med elementi iste skupine minimizirati, razdaljo med elementi različnih skupin pa maksimizirati. To pomeni, da algoritmi za razvrščanje močno slonijo na merah podobnosti/razdalj. Poleg mere razdalj moramo definirati tudi prag še sprejemljive gostote in predvideno število ciljnih skupin. Kako bomo nastavili parametre, je odvisno od podatkovne zbirke, na kateri opravljamo meritve, in od želenih rezultatov. Za pridobitev želenih rezultatov se moramo učiti na napakah, kar pomeni, da opravimo več iteracij meritev z različnimi parametri na različno pred-procesiranih zbirkah podatkov.
V tem poglavju si bomo ogledali tri metode razvrščanja, ki temeljijo na centroidih, ter kasneje z njihovo pomočjo preizkušali mere podobnosti, opisane v poglavju 4.

5.2 KMeans

V podatkovnem rudarjenju se uporablja algoritem KMeans za razvrstitev n elementov v k skupin (kjer navadno velja $n \gg k$) na podlagi podobnosti.

5.2.1 Osnove

Algoritem KMeans sprejme tri vrednosti.

- k — število centroidov (razredov),
- M — matrika vektorjev izrazov velikosti $m \times n$,

- *maxI* — največje dovoljeno število iteracij (lahko tudi ni podano).

Naša naloga je izbrati k centroidov (posamezen centroid označimo z $\vec{c_i} \in C$), tako da minimiziramo:

$$\sum_{i=0}^{m} \min_{c \in C} \left|\vec{t_i} - \vec{c}\right|^2 \tag{5.1}$$

Izbrani centroidi nam pomagajo pri razvrščanju dokumentov v skupine. Vektor izrazov $\vec{t_i}$ priredimo centroidu c_j natanko tedaj, ko je vektor izrazov $\vec{t_i}$ najbližje centroidu c_j.

Izbira centroidov in razvrščanje dokumentov v skupine glede na izbrane centroide spada med NP-polne probleme, tudi v primeru, ko je $k = 2$. Semi-optimalen način izbire centroidov in razvrščanja dokumentov lahko dosežemo s pomočjo hevristične izboljšave, ki poskrbi za hitro konvergenco k lokalnemu optimumu.

5.2.2 Algoritem

KMeans je zaradi svoje hitrosti in preprostosti eden izmed najpogosteje uporabljenih algoritmov v nenadzorovanem strojnem učenju, njegova točnost pa je zelo odvisna od začetnega izbora centroidov. Za pridobitev natančnejših rezultatov lahko poženemo algoritem večkrat in dobljene rezultate povprečimo. Psevdokoda algoritma se nahaja v tabeli 1.

5.2.3 Kompleksnost

Časovna zahtevnost algoritma KMeans je $O(n \cdot k \cdot i \cdot d)$, kjer je:

- n — število vektorjev izrazov,
- k — število centroidov,
- i — število iteracij,
- d — dolžina vektorja izrazov (število izrazov, ki sestavlja vektor izrazov).

Časovno zahtevnost lahko izboljšamo z uporabo hevrističnih algoritmov, kot je *LLoydov* algoritem.

Algoritem 1 Delovanje algoritma KMeans

Vhod: Algoritem sprejme matriko vektorjev izrazov M, število centroidov k in (po želji) maksimalno število iteracij $maxI$.

Izhod: Vrne množico izračunanih centroidov C in množico G z informacijo, kateremu centroidu pripada posamezen vektor izrazov.

1: Naključno izberi $c_1 \cdots c_k \in M$, za katere velja $c_i \neq c_j$, če $i \neq j$
2: Odstrani izbrane vektorje izrazov iz matrike vektorjev izrazov
3: **while** C se spreminja ALI $i \leq maxI$ **do**
4: $i := 0$
5: **for** $i := 1$ to n **do** `// Sprehodi se skozi vse vektorje izrazov`
6: **for** $j := 1$ to k **do** `// Sprehodi se skozi centroide`
7: izračunaj razdaljo med $\vec{t_i}$ in $\vec{c_j}$
8: **end for**
9: dodeli $\vec{t_i}$ v grupo g_k, ki pripada najbližjemu centroidu $\vec{c_k}$
10: **end for**
11: na podlagi dobljenih grup okoli centroidov izračunaj nove centroide
12: posodobi množico centroidov C
13: $i := i + 1$
14: **end while**
15: **return** C `// Vrni množico centroidov`
16: **return** G `// Vrni množico podatkov, kateremu centroidu pripada posamezen vektor izrazov`

LLoydov algoritem

Gre za izboljšavo algoritma *KMeans* na področju časovne kompleksnosti.
Kot vhod sprejmemo matriko vektorjev izrazov, katere si lahko predstavljamo kot točke v visoko-dimenzionalnem prostoru. Nato vstopimo v zanko, ki se izvaja, dokler se vsebina skupine spreminja. Zanka je sestavljena iz treh ključnih korakov, ki so:

1. Izračunaj diagram *Voronoi-a*[1] (za izračun uporabimo metode *Monte Carlo*[2]).
2. Integriraj posamezno celico diagrama in izračunaj centroid.
3. Prerazporedi točke k centroidom celic.

Po vsakem koraku so točke v diagramu bolj enakomerno razporejene — razdalja med točkami, ki so si bile blizu, se poveča, razdalja med oddaljenimi točkami pa se zmanjša. Konvergenca je počasna zaradi numeričnih omejitev, zato se v praksi algoritem zaključi, ko je razporeditev 'dovolj dobra' glede na naše zahteve.

5.2.4 Slabosti in omejitve

Glavni dve slabosti algoritma KMeans sta:

1. dokazana super-polinomična zahtevnost v najslabšem primeru,
2. ob neoptimalni izbiri začetnih centroidov pa dobimo slabe rezultate.

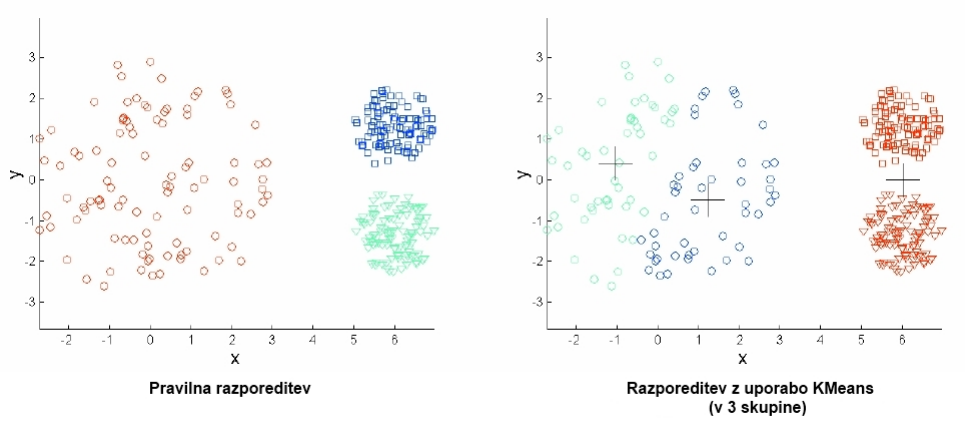
Omejitve

Algoritem KMeans ima probleme, če so razredi:

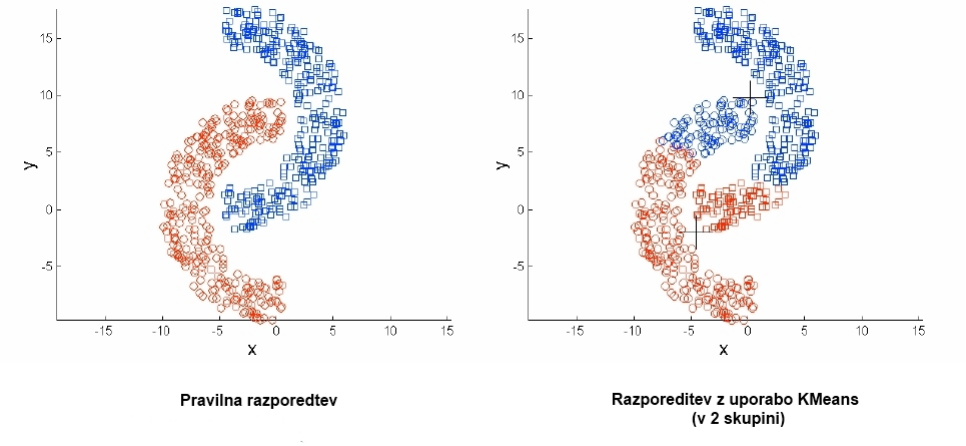
- različnih velikosti in/ali gostot *(slika 5.1)*,
- različnih nekroglastih oblik *(slika 5.2)*,
- prazni,
- vsebujejo osamelce (angl. outliers) — to so elementi, katerih razdalja od centroida je veliko večja kot pri ostalih.

[1] Z uporabo Voronoi-evega diagrama lahko razdelimo prostor na regije, ki jih imenujemo Voronoi-eve celice. Vsaki celici pripada ključna točka, ki je v našem primeru centroid skupine. [Vor]

[2] V razred Monte Carlo metod spadajo vsi algoritmi, ki pridobijo rezultate s pomočjo večkratnega naključnega vzorčenja — npr. ocenitev hevristik na podlagi večkratnih simulacij. [Mon]



Slika 5.1: Primer problema različnih gostot. Razred z rdeče predstavljenimi podatki je veliko bolj razkropljen kot ostali. Problem se pojavi pri izbiri centroidov, česar posledica so narobe razvrščeni podatki. [TG]



Slika 5.2: Primer problema različnih nekroglastih oblik. Centroidi so narobe določeni in zato je tudi skoraj polovica podatkov razvrščenih v napačen razred. [TG]

5.3 KMeans++

Uspešnost algoritma *KMeans* je v veliki meri odvisna od začetnega izbora centroidov, kar je tudi glavna ideja za izboljšavo v algoritmu *KMeans++*.

5.3.1 Osnove

Leta 2007 sta David Arthur in Sergei Vassilvitskii [kmea] prvič predstavila algoritem *KMeans++*, ki temelji na algoritmu *KMeans*. Posvetila sta se predvsem izboljšavi drugega problema, opisanega v razdelku 5.2.4.
Začetne centroide izbiramo s pomočjo verjetnosti. Kako pa določimo verjetnost posameznega elementa?

5.3.2 Računanje verjetnosti

Za določitev k centroidov potrebujemo $k-1$ iteracij. Kot prvi centroid si izberemo naključni vektor izrazov $\vec{t_i} \in M$, kjer je M matrika vektorjev izrazov, nato pa vstopimo v zanko $k-1$ korakov. Sedaj za vsak vektor izrazov $\vec{t_j}$ izračunamo njegovo verjetnost po formuli

$$P(\vec{t_j}) = \frac{D^2(\vec{t_j})}{\sum_{k=0}^{n} D^2(\vec{t_k})} \tag{5.2}$$

kjer je $D^2(\vec{t_i})$ razdalja med vektorjem izrazov $\vec{t_i}$ in njemu najbližjim, že določenim, centroidom. V vsakem koraku iteracije ponovno izračunamo verjetnosti vseh vektorjev izrazov, saj smo z dodajanjem novega centroida pridobili nove informacije.

5.3.3 Algoritem

Psevdokoda algoritma *KMeans++* je zapisana v tabeli 2.

5.4 Kombinacija KMeans++ in KMedoids

Algoritem je kombinacija pozitivnih lastnosti algoritmov *KMeans* in *KMedoids*.

5.4.1 Ideja

Osnovni algoritem *KMeans* izbira začetne centroide naključno in temelji na minimizaciji variance med podatki skupin. Varianco lahko minimiziramo tako, da minimiziramo evklidsko razdaljo med elementi skupin in njegovim centrom. V takšnem primeru algoritem vedno konvergira, če pa namesto evklidske razdalje uporabimo katero drugo razdaljo, to ne drži več, kar želimo tudi popraviti.

Algoritem 2 Delovanje algoritma KMeans++

Vhod: Algoritem sprejme matriko vektorjev izrazov M, število centroidov k in (po želji) maksimalno število iteracij $maxI$.

Izhod: Vrne množico izračunanih centroidov C in množico G z informacijo, kateremu centroidu pripada posamezen vektor izrazov.

1: Naključno izberi centroid $c_1 \in M$
2: **while** $|C| \leq k$ **do**
3: Izberi $\vec{t_i} \in M$ z verjetnostjo glede na $D^2(\vec{t_i})$ `// Verjetnost se računa glede na razdaljo med` $\vec{t_i}$ `in najbližjim že izbranim centroidom`
4: $C = C \cup \vec{t_i}$
5: Odstrani izbrani vektor izrazov $\vec{t_i}$ iz matrike vektorjev izrazov
6: **end while**
7: **while** C se spreminja ALI $i \leq maxI$ **do**
8: $i := 0$
9: **for** $i := 1$ to n **do** `// Sprehodi se skozi vse vektorje izrazov`
10: **for** $j := 1$ to k **do** `// Sprehodi se skozi centroide`
11: izračunaj razdaljo med $\vec{t_i}$ in $\vec{c_j}$
12: **end for**
13: dodeli $\vec{t_i}$ v grupo g_k, ki pripada najbližjemu centroidu $\vec{c_k}$
14: **end for**
15: na podlagi dobljenih grup okoli centroidov izračunaj nove centroide
16: posodobi množico centroidov C
17: $i := i + 1$
18: **end while**
19: **return** C `// Vrni množico centroidov`
20: **return** G `// Vrni množico podatkov, kateremu centroidu pripada posamezen vektor izrazov`

5.4.2 Delovanje

Algoritem *KMedoids++* sprejme tri parametre:

- k - *število centroidov*,
- M - *matrika vektorjev izrazov velikosti* $m \times n$,
- maxIter - *število dovoljenih iteracij*.

Verjetnost, da izberemo centroid za vektor izrazov $\vec{t_i}$, izračunamo po postopku, predstavljenem v razdelku 5.3.2. Nato si izberemo k naključnih vektorjev izrazov v skladu z verjetnostjo, ki jim je prirejena, in jih določimo kot začetne centroide. Ko imamo začetne centroide določene, vstopimo v zanko, ki se izvaja, dokler se skupine spreminjajo. Za obstoječe skupine izračunamo nov centroid, tako da izračunamo povprečno vrednost elmentov, ki mu pripadajo - dobimo nov element, ki pa v naši podatkovni zbirki ne obstaja. Kot pravi centroid si izberemo element, ki je glede na našo izbrano razdaljo najbližje dobljenemu navideznemu elementu. Algoritem bo vedno konvergiral za poljubno izbrano mero razdalje, ni pa zagotovljeno, da bomo dobili optimalno rešitev. Psevdokoda algoritma je vidna v tabeli 3.

5.4.3 Kompleksnost

Časovna kompleksnost izbire začetnih centroidov je: $O(k{\cdot}a{\cdot}n)$. Časovna kompleksnost algoritma je: $O(i \cdot (n \cdot k \cdot a + k \cdot (k \cdot + n \cdot a)))$. Če obe časovni kompleksnosti seštejemo in poenostavimo, dobimo $O(i \cdot n \cdot k \cdot a)$, kjer je:

- i — število iteracij,
- n — število vektorjev izrazov,
- k — število centroidov (skupin),
- a — dimenzija (dolžina) vektorja izrazov.

5.4.4 Slabosti

Tako kot algoritma *KMeans* in *KMeans++*, tudi ta algoritem ni odporen na:

- razrede različnih gostot in velikosti,
- razrede različnih ne-kroglastih oblik,
- prazne razrede.
- osamelce

Algoritem 3 Delovanje algoritma KMedoids++

Vhod: Algoritem sprejme matriko vektorjev izrazov M, število centroidov k, želeno mero razdalje r in (po želji) maksimalno število iteracij $maxI$.

Izhod: Vrne množico izračunanih centroidov C in množico G z informacijo, kateremu centroidu pripada posamezen vektor izrazov.

1: Naključno izberi centroid $c_1 \in M$
2: **while** $|C| \leq k$ **do**
3: Izberi $\vec{t_i} \in M$ z verjetnostjo glede na $D^2(\vec{t_i})$ `// Verjetnost se računa glede na razdaljo med` $\vec{t_i}$ `in najbližjim že izbranim centroidom. Za računanje razdalje uporabimo izbrano mero razdalje` r
4: $C = C \cup \vec{t_i}$
5: Odstrani izbrani vektor izrazov $\vec{t_i}$ iz matrike vektorjev izrazov `// Tako preprečimo morebitno podvajanje centroidov`
6: **end while**
7: $iter := 0$
8: **while** C se spreminja ALI $iter \leq maxI$ **do**
9: **for** $i := 1$ to n **do** `// Sprehodi se skozi vse vektorje izrazov`
10: **for** $j := 1$ to k **do** `// Sprehodi se skozi centroide`
11: izračunaj razdaljo med $\vec{t_i}$ in $\vec{c_j}$
12: **end for**
13: dodeli $\vec{t_i}$ v grupo g_k, ki pripada najbližjemu centroidu $\vec{c_k}$
14: **end for**
15: **for** $j := 1$ to k **do** `// Izracunaj povprečnje skupin`
16: $\vec{c_j}$:= povprečje elementov skupine k
17: poišči element $\vec{t_i}$, ki je glede na mero razdalje r najbližje centroidu $\vec{c_j}$
18: $\vec{c_j} := \vec{t_i}$
19: **end for**
20: posodobi množico centroidov C
21: $iter := iter + 1$
22: **end while**
23: **return** C `// Vrni množico centroidov`
24: **return** G `// Vrni množico podatkov, kateremu centroidu pripada posamezen vektor izrazov`

Poglavje 6

Metodologija evalvacije

Uspešnost mer podobnosti iz poglavja 4 in metod iz poglavja 5, bomo preizkušali na podatkovnih zbirkah besedil, opisanih v nadaljevanju.
Ker je rezultat algoritma *KMeans* odvisen od začetne izbire centroidov, bomo vsak test ponovil 20 krat in dobljene rezultate povprečili, pri tem moramo še poskrbeti, da bo začetna izbira centroidov testov neponovljiva. V primeru, da bi katera od metod imela težave s konvergenco pri izbrani meri podobnosti, bomo omejili število dovoljenih iteracij na 30.

6.1 Uporabljene množice podatkov

V tem razdelku predstavimo podatkovni množici, ki sta bili uporabljeni pri eksperimentu, ter njune lastnosti.

besedila

Zbirka vsebuje 1015 različnih besedil s področja ekonomije. Besedila lahko razdelimo v dva razreda: splošna besedila in tržna besedila. V nobenem od besedil ne nastopajo meta-podatki, *HTML* koda ali katera druga koda. Po filtriranju vsebuje zbirka 5259 različnih izrazov, ki jih utežimo z *tf-idf*. Utežene podatke razvrstimo padajoče po prirejeni uteži ter za eksperiment izberemo prvih 2000, s čimer dobimo matriko vektorjev izrazov velikosti 1015×2000.

classic-docs [Tun10]

Podatkovna zbirka vsebuje 7097 akademskih člankov. Razdelimo jih lahko v 4 razrede: CACM (3204 besedil), CISI (1460 besedil), CRAN (1400 besedil) in MED (1033 besedil). Ta zbirka besedil vsebuje moteče informacije (npr. glava e-sporočila, ...), zato besedila v fazi predprocesiranja ustrezno prečistimo. Število vseh izrazov v podatkovni

zbirki je 12009, po predprocesiranju in nastavitvi praga pojavitev nam ostane še 806 izrazov in 7095 besedil — dveh tekstovnih dokumentov ne moremo predstaviti s preostalimi izrazi — ki jih utežimo s *tf-idf* in tako dobimo matriko vektorjev izrazov velikosti 7095×806.

6.2 Mere za uspešnost učenja

Na opisanih podatkovnih množicah izvedemo razvrščanje z metodami razvrščanja, opisanimi v poglavju 5. Metode razvrščanja nam elemente podatkovne množice razvrstijo v izbrano število skupin. Skupina pripada razredu, katerega elementov vsebuje največ. Lahko se zgodi, da več skupin pripada istemu razredu. Kako dobro smo elemente razvrstili v skupine, ocenimo s pomočjo mer čistoče in entropije, ki ju opisujemo v nadaljevanju.

Čistoča (angl. purity)

Čistočo izračunamo tako, da preštejemo število pravilno klasificiranih dokumentov. Označimo s C_i i-to skupino velikosti n_i, tedaj se bo enačba za čistočo skupine C_i glasila:

$$P(C_i) = \frac{1}{n_i} \cdot \max(n_i^k) \tag{6.1}$$

kjer z $max(n_i^k)$ označimo število prevladujočih dokumentov znotraj skupine.
Čistoča vrača vrednosti iz intervala $[0, 1]$. V idealnem primeru, ko bodo skupini pripadali le dokumenti istega razreda, bo zavzela vrednost 1. Torej višja kot je čistoča zbirke dokumentov, optimalneje so le-ti razporejeni v skupine.
Čistočo zbirke dokumentov zapišemo kot uteženo vsoto posameznih skupin zbirke:

$$P = \sum_{i=1}^{k} \frac{n_i}{n} \cdot P(C_i) \tag{6.2}$$

Entropija (angl. entropy)

Entropija je ena izmed mer nedoločenosti dinamičnega, naključnega diskretnega sistema [eva], ki ovrednoti razporeditev kategorij v dane skupine. Označimo poljubno skupino velikosti n_i s C_i, potem se bo njena mera entropije glasila:

$$E(C_i) = -\frac{1}{\log(c)} \cdot \sum_{j=1}^{k} \frac{n_i^j}{n_i} \cdot \log(\frac{n_i^j}{n_i}) \tag{6.3}$$

kjer s c označimo število vseh kategorij množice dokumentov, z n_i^h število primerov v skupini C_i, ki pripadajo j-temu razredu, in s k število vseh razredov. Pri tem upoštevamo, da je število kategorij c enako številu razredov.
V splošnem entropija dodeli boljšo oceno primerom, v katerih nastopa večje število skupin, kar pomeni, da če v skrajnem primeru vsakemu dokumentu namenimo svojo skupino, je vrednost entropije optimalna. V našem eksperimentu bi nam to povzročalo težave, saj imamo primera z različnim številom razredov in s tem tudi različnim številom skupin. To težavo lahko odpravimo tako, da entropije normaliziramo — delimo jih z logaritmom števila vseh razredov množice dokumentov, kar je razvidno tudi v enačbi 6.3. Entropija po normalizaciji vrne rezultat iz množice $[0, 1]$, kjer 0 ozačuje popolno skupino, v kateri nastopajo zgolj dokumenti enega razreda, torej manjša kot je vrednost entropije, kvalitetnejša je opazovana skupina. Povprečno entropijo množice tekstovnih dokumentov zapišemo kot uteženo vsoto entropij posameznih skupin:

$$E = \sum_{i=1}^{k} \frac{n_i}{n} \cdot E(C_i) \tag{6.4}$$

kjer n označuje število vseh tekstovnih dokumentov znotraj zbirke.

6.3 Ocenjevanje optimalnega števila skupin (k)

Pomemben parameter, ki ga moramo podati opisanim metodam razvrščanja, je parameter števila skupin. V primeru, da poznamo število kategorij/rezredov, to ni problem, saj nastavimo $k = stRazredov$. Vendar v praksi navadno ne poznamo števila kategorij, zato moramo optimalen k oceniti s pomočjo različnih metod. Metode, s katerimi bomo ocenili optimalen k, so opisane v nadaljevanju.

6.3.1 Pravilo palca (angl. rule of thumb)

Pri tem pravilu določimo k po enačbi:

$$k \approx \sqrt{\frac{n}{2}} \tag{6.5}$$

kjer n predstavlja število tekstovnih dokumentov.

6.3.2 Metoda iskanja števila skupin tekstovnih množic (angl. finding number of clusters in text databases)

Vzemimo matriko vektorjev izrazov D velikosti $m \times n$, kjer m predstavlja število tekstovnih dokumentov in n predstavlja število izrazov. Število skupin ocenimo z

enačbo:

$$k \approx \frac{m \cdot n}{t} \tag{6.6}$$

kjer t predstavlja število neničelnih elementov v matriki D.

6.3.3 Metoda komolca (angl. elbow method)

Pri tej metodi izračunamo razporeditve pri različnem številu skupin k (navadno kar za naravna števila iz intervala $[2, 20]$). Za optimalni k izberemo tisto število k, od katerega dalje ne izboljšamo rezultatov z dodajanjem skupin več bistveno. Tako število k imenujemo *komolec*.

6.3.4 Metoda silhouette (angl. silhouette method)

Metodo je prvič opisal Peter J. Rousseeuw [Wik12] leta 1986. S pomočjo te metode ocenimo, kako dobro se prilega posamezen element v dodeljeno skupino. Torej je potrebno najprej izračunati ločitev na skupine. To storimo z uporabo metode *KMeans* ali katere druge metode razvrščanja. Mi bomo razvrstitev izračunali z uporabo metode *KMeans*, za vsak k iz intervala naravnih števil $[1, 20]$. Nato za vsak element e znotraj skupine C, izračunamo povprečno oddaljenost $a(e)$ do vseh ostalih elementov znotraj iste skupine. Za izračun razdalje lahko uporabimo poljubno mero razdalje, ki je metrika. Nato izračunamo povprečno oddaljenost od elementa e do elementov ostalih skupin in najkrajšo razdaljo označimo z $b(e)$. S tem določimo skupino, ki je sosednja skupini C. Sedaj lahko zapišemo enačbo:

$$s(e) = \begin{cases} 1 - \frac{a(e)}{b(e)} & \text{če } a(e) < b(e) \\ 0 & \text{če } a(e) \equiv b(e) \\ \frac{b(e)}{a(e)} - 1 & \text{če } a(e) > b(e) \end{cases} \tag{6.7}$$

Iz enačbe je razvidno, da $s(e)$ zavzame vrednosti iz intervala $[-1, 1]$. V primeru, da je $s(e) = 1$, pomeni, da se element e dobro prilega v skupino C. V primeru, da je $s(e) = -1$, pa nam pove, da bi bilo bolje, če bi element e uvrstili v sosednjo skupino. Povprečje $s(e)$ vseh elementov podatkovne množice nam pove, kako dobro se prilegajo v izbrane skupine. Izbrali bomo tisti k, pri katerem se elementi najbolje prilegajo v skupine — torej k, pri katerem je povprečen $s(e)$ najbližje 1.

6.3.5 Pristop z informacijskim kriterijem (angl. information criterion approach)

Pri tem pristopu lahko uporabimo različne metode za izračun informacijskega kriterija skupine. V našem eksperimentu bomo uporabili *Bayesov* informacijski kriterij (angl.

Bayesian information criterion *(BIC)*). Enačba po kateri izračunamo BIC je:

$$BIC \approx -2 \cdot \log(p(D\|M)) \tag{6.8}$$

kjer je D matrika vektorjev izrazov in $p(D\|M)$ integrirana funkcija verjetja matrike vektorjev izrazov D, pri dani razporeditvi M.

6.3.6 Statistika "gap" (angl. gap statistics)

Izračunajmo *KMeans* razporeditev za vsak k z intervala naravnih števil $[1, 20]$. Nato izračunajmo razdaljo med elementi d_{ij} skupine C_r, velikosti n_r pri vsaki razporeditvi k. Vsoto razdalj med pari znotraj skupine zapišemo kot:

$$D_r = \sum_{i,j} d_{ij}. \tag{6.9}$$

Definirajmo sedaj W_k kot:

$$W_k = \sum_{r=1}^{k} \frac{1}{2 \cdot n_r} \cdot D_r. \tag{6.10}$$

Statistiko "gap" lahko zapišemo z enačbo:

$$Gap_n(k) = E_n^*(\log(W_k)) - log(W_k), \tag{6.11}$$

kjer $E_n^*(\log(W_k))$ ocenimo s postopkom *bootstrapping*. Izbrali si bomo k, pri katerem statistika "gap" doseže maksimum.

Poglavje 7

Rezultati

V tem razdelku so zbrane meritve opisanih metod z uporabo različnih razdalj. Razdeljene so po podatkovnih množicah, na katerih so bile opravljene. Za vsako podatkovno množico smo izračunali optimalno število skupin z metodami, opisanimi v razdelku 6.3.

7.1 Analiza množice *besedila*

Znano število kategorij podatkovne množice je 2. V tabeli 7.1 so zbrane ocenjene vrednosti k z različnimi metodami. Poglejmo si sedaj podrobneje ocene k z izbranimi metodami.

- Število vseh dokumentov znotraj množice je 1015, torej lahko z uporabo metode palca, izračunamo optimalni k kot $k \approx \sqrt{\frac{1015}{2}} \approx 23$.
- Po filtriranju podatkovne množice imamo 2000 izrazov in 1015 tekstovnih dokumentov, kar določa velikost matrike vektorjev izrazov z 2000×1015. Število neničelnih vnosov je 39527. Sedaj lahko ocenimo k kot $k \approx \frac{2000 \cdot 1015}{39527} \approx 51$.
- Na sliki 7.1 je prikazan graf števila k v odvisnosti od vsote korenov razdalj znotraj skupin. S pomočjo tega grafa želimo poiskati število k, ki predstavlja *komolec*. Opazimo, da takšnega števila k ni, torej si z metodo *komolca* ne moremo pomagati pri oceni števila skupin.
- Z uporabo metode *Silhouette* dobimo grafa prikazana na slikah 7.2 in 7.3. Iz grafov odčitamo, da nam ta metoda svetuje uporabo 10 skupin.
- Z uporabo metode *BIC* dobimo grafa prikazana na slikah 7.4 in 7.5. Na prvem grafu lahko opazimo, da je maksimum dosežen pri številu 3, torej izberemo

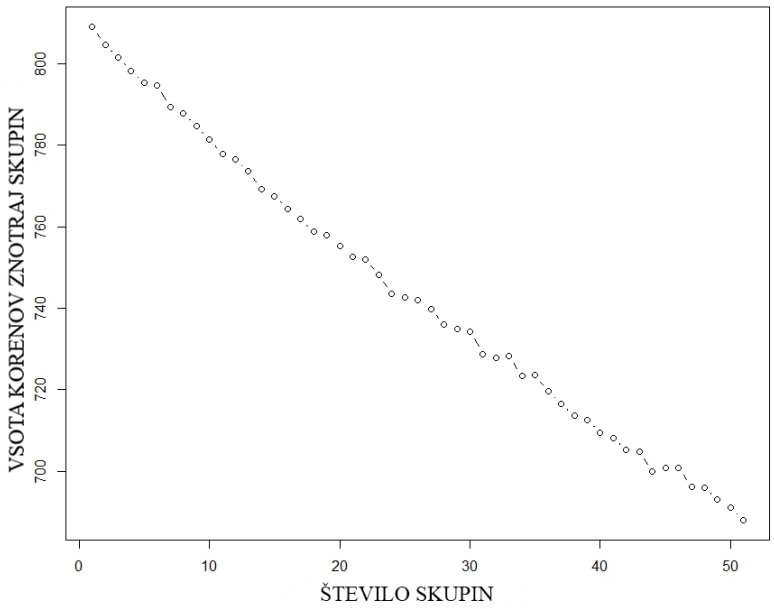
$k = 3$. Na drugem grafu je prikazana razvrstitev besedil podatkovne množice v 3 skupine.

- Z izračunom statistike "gap" dobimo graf 7.6. Iz grafa odčitamo, da je optimalno število skupin 1, takoj za tem pa 3.

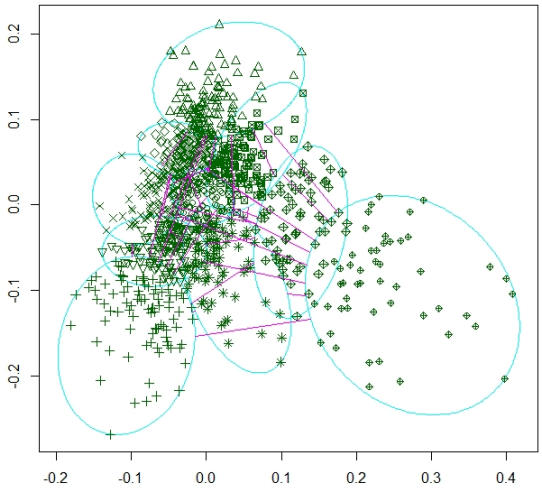
Izračunane čistoče in entropije, pri uporabi metod *KMeans, KMeans++* in *KMedoids++*, v kombinaciji z merami razdalj, opisanimi v poglavju 4, pri različnem številu k, so zbrane v tabelah 7.2, 7.5, 7.8 in 7.11.

metoda	k
metoda palca	23
metoda 2	51
metoda komolca	/
metoda *silhouette*	10
metoda BIC	3
statistika "gap"	3

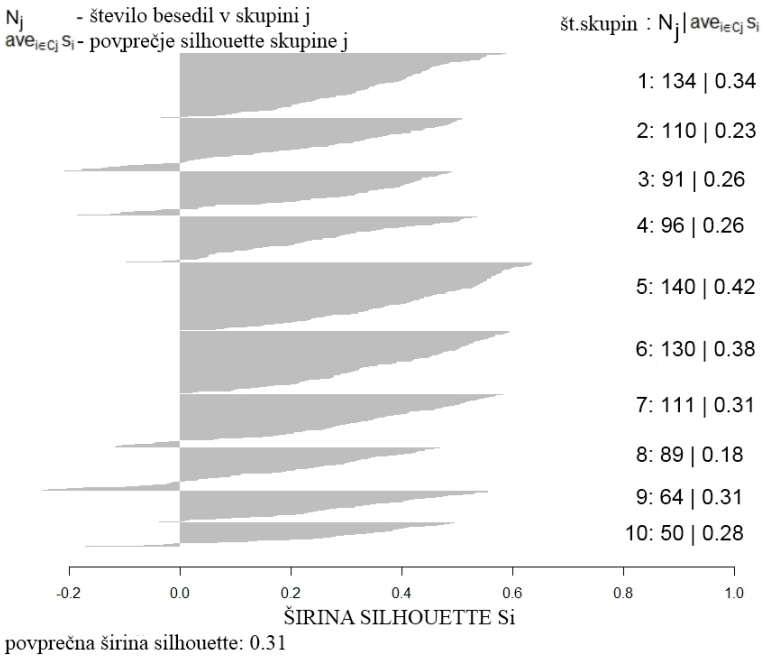
Tabela 7.1: Vrednosti k pri različnih metodah.



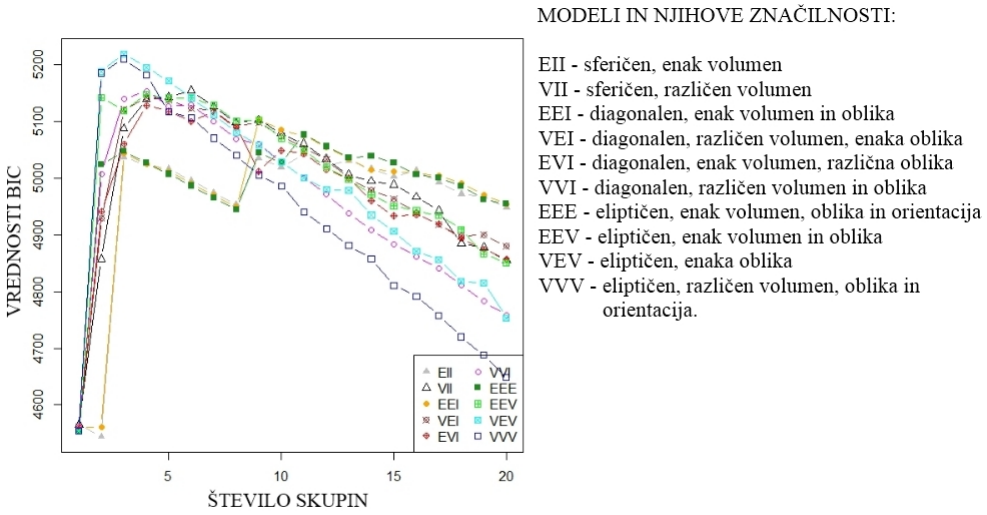
Slika 7.1: Prikaz podatkov, potrebnih za določitev števila skupin po metodi komolca.



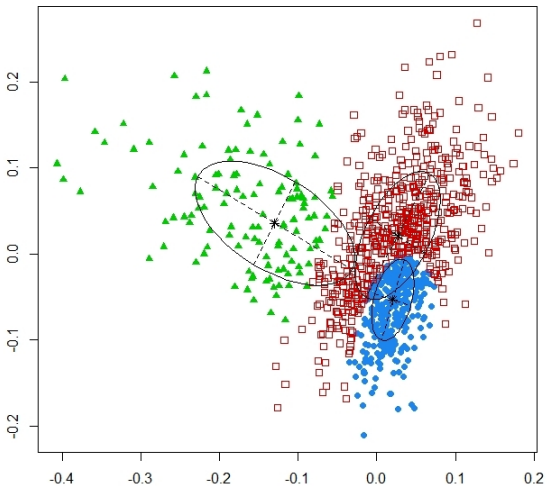
Slika 7.2: Graf prikazuje delitev podatkovne množice v 10 skupin z uporabo metode *silhouette*. Podatki visoko-dimenzionalnega prostora, ki ga določa matrika vektorjev izrazov, množice *besedila*, so preslikani v 2-dimenzionalni prostor za lažjo predstavitev.



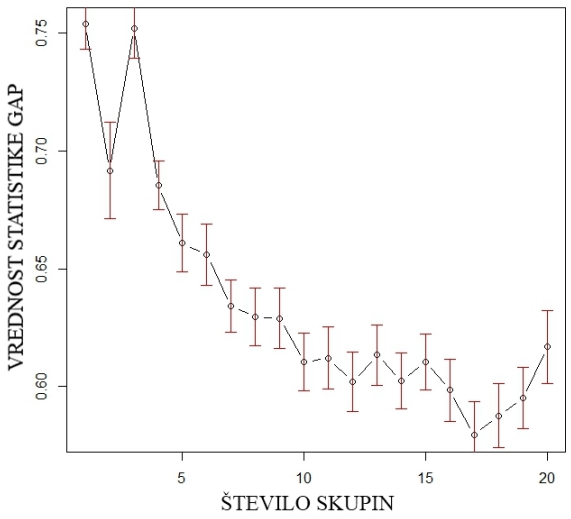
Slika 7.3: Graf prikazuje prileganje elementov v izbrane skupine.



Slika 7.4: Graf prikazuje *BIC* vrednosti pri različnem številu skupin k.



Slika 7.5: Graf prikazuje razvrstitev besedil podatkovne množice v 3 skupine, izbrane s pomočjo metode *BIC*. Podatki visoko-dimenzionalnega prostora, ki ga določa matrika vektorjev izrazov, množice *besedila*, so preslikani v 2-dimenzionalni prostor za lažjo predstavitev.



Slika 7.6: Graf prikazuje izračunane statistike “gap” pri različnem številu skupin.

Metoda KMeans

št.skup.	*Evkl. r.*	*Cos. r.*	*P. k. k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 2$	**0.9044**	0.7143	0.7300	0.7118	0.8167	0.8138	0.7073	**0.7712**
$k = 3$	0.6246	0.7143	0.7123	0.7133	0.6246	0.6266	**0.7823**	0.6854
$k = 10$	0.6266	0.6700	0.7163	**0.7547**	0.6374	0.6246	0.7438	0.6819
$k = 23$	0.6150	**0.7724**	0.7113	0.7330	0.7291	0.6246	0.7488	0.7052
$k = 51$	0.7211	0.7064	0.6709	**0.7438**	0.6250	0.6502	0.7251	0.6918
povp.	0.6983	0.7212	0.7082	0.7313	0.6866	0.6680	**0.7415**	*0.7071*

Metoda KMeans++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 2$	0.6276	0.6601	0.6246	**0.6995**	0.6246	0.6256	0.6670	0.6470
$k = 3$	0.6680	**0.7251**	0.6956	0.6246	0.6246	0.6305	0.7025	0.6673
$k = 10$	0.7192	0.7330	0.7113	**0.7773**	0.6246	0.6828	0.7241	**0.7103**
$k = 23$	0.7172	0.7163	0.6621	**0.7557**	0.6473	0.6246	0.7133	0.6929
$k = 51$	0.7015	0.7310	0.6689	0.7468	0.6246	0.6246	**0.7517**	0.6927
povp.	0.6853	0.7131	0.6725	**0.7208**	0.6327	0.6376	0.7117	*0.6820*

Metoda KMedoids++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 2$	0.6246	0.6404	0.6246	0.6246	0.6246	0.6246	**0.6286**	0.6274
$k = 3$	0.6246	0.6660	0.6246	**0.6887**	0.6246	0.6246	0.6423	0.6422
$k = 10$	0.6256	0.6640	0.6246	0.6650	0.6246	0.6246	**0.6680**	0.6423
$k = 23$	0.6365	**0.7084**	0.6246	0.7074	0.6246	0.6246	0.7074	0.6619
$k = 51$	0.6700	0.7251	0.6246	**0.7340**	0.6246	0.6325	0.7291	**0.6771**
povp.	0.6363	0.6808	0.6246	**0.6799**	0.6246	0.6262	0.6751	***0.6502***

Tabela 7.2: Tabela izračunanih *čistoč* na podatkovni zbirki *besedila* z uporabo reprezentacije *vreče besed*.

Evkl.r.	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*
0.6733	0.7050	0.6684	**0.7107**	0.6480	0.6439	0.7094

Tabela 7.3: Povprečja čistoč glede na uporabljeno razdaljo na podatkovni množici *besedila* z uporabo reprezentacije *vreče besed*.

$k = 2$	$k = 3$	$k = 10$	$k = 23$	$k = 51$
0.6819	0.6650	0.6782	0.6867	**0.6872**

Tabela 7.4: Povprečja čistoč glede na izbrani k na podatkovni množici *besedila* z uporabo reprezentacije *vreče besed*.

Metoda KMeans

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 2$	**0.4419**	0.8261	0.8359	0.8228	0.9454	0.4991	0.8676	**0.7484**
$k = 3$	0.8815	0.8275	0.8579	0.8247	0.9239	0.9532	**0.7314**	0.7527
$k = 10$	0.9455	0.8679	0.8077	**0.7585**	0.9259	0.9524	0.7916	0.8642
$k = 23$	0.8917	**0.7042**	0.8319	0.7741	0.7873	0.9547	0.7439	0.8125
$k = 51$	0.7967	0.8135	0.8802	**0.7429**	0.9547	0.9233	0.7886	0.8428
povp.	0.7915	0.8078	0.8427	**0.7846**	0.9074	0.8566	**0.7846**	***0.8041***

Metoda KMeans++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 2$	0.9523	0.8754	0.9449	**0.8158**	0.9209	0.9539	0.8499	0.9019
$k = 3$	0.9157	**0.8220**	0.8625	0.9020	0.9547	0.9357	0.8403	0.8904
$k = 10$	0.8411	0.7683	0.8305	**0.7098**	0.9544	0.8500	0.7922	**0.8209**
$k = 23$	0.8231	0.7917	0.8798	**0.7362**	0.9083	0.9542	0.8056	0.8427
$k = 51$	0.8122	0.7806	0.8823	**0.7316**	0.9120	0.9529	0.7431	0.8307
povp.	0.8689	0.8076	0.8800	**0.7791**	0.9301	0.9293	0.8062	*0.8573*

Metoda KMedoids++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 2$	0.9504	0.8889	0.9546	0.9434	0.9540	0.9507	**0.8775**	0.9314
$k = 3$	0.9545	**0.8770**	0.9521	0.8777	0.9540	0.9506	0.9191	0.9264
$k = 10$	0.9413	0.9055	0.9892	**0.8886**	0.9540	0.9467	0.8946	0.9314
$k = 23$	0.9185	0.8436	0.9489	**0.8065**	0.9540	0.9458	0.8543	0.8959
$k = 51$	0.8579	0.7873	0.9444	**0.7705**	0.9540	0.9245	0.7802	**0.8598**
povp.	0.9245	0.8605	0.9578	**0.8573**	0.9540	0.9437	0.8651	*0.9090*

Tabela 7.5: Tabela izračunanih *entropij* na podatkovni zbirki *besedila* z uporabo reprezentacije *vreče besed*.

Evkl. r.	*Cos. r.*	*P. k. k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*
0.8616	0.8253	0.8935	**0.8070**	0.9305	0.9100	0.8186

Tabela 7.6: Povprečja entropij glede na uporabljeno razdaljo na podatkovni množici *besedila* z uporabo reprezentacije *vreče besed*.

$k = 2$	$k = 3$	$k = 10$	$k = 23$	$k = 51$
0.8606	0.8565	0.8722	0.8504	**0.8444**

Tabela 7.7: Povprečja entropij glede na izbrani k na podatkovni množici *besedila* z uporabo reprezentacije *vreče besed*.

Metoda KMeans

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 2$	0.6266	0.6246	0.6246	**0.6473**	0.6246	0.6246	0.6246	0.6281
$k = 3$	0.6246	**0.6287**	0.6246	0.6246	0.6246	0.6246	0.6246	0.6252
$k = 10$	0.6246	0.6325	0.6276	0.6522	0.6355	0.6433	**0.6778**	0.6419
$k = 23$	0.6374	0.6581	0.6246	**0.6601**	0.6246	0.6315	0.6394	0.6394
$k = 51$	0.6424	**0.6906**	0.6374	**0.6906**	0.6374	0.6867	0.6788	**0.6663**
povp.	0.6311	0.6469	0.6278	**0.6550**	0.6293	0.6421	0.6490	*0.6402*

Metoda KMeans++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 2$	**0.6256**	0.6246	0.6246	0.6246	0.6246	0.6246	0.6246	0.6248
$k = 3$	0.6246	0.6246	0.6246	**0.6414**	0.6246	0.6246	0.6345	0.6284
$k = 10$	0.6266	**0.6404**	0.6246	0.6344	0.6305	0.6276	0.6246	0.6298
$k = 23$	0.6335	**0.6680**	0.6246	**0.6680**	0.6365	0.6453	0.6640	0.6486
$k = 51$	0.6424	0.7015	0.6404	0.6977	0.6384	0.6995	**0.7044**	**0.6749**
povp.	0.6305	0.6518	0.6278	**0.6532**	0.6309	0.6443	0.6504	***0.6413***

Metoda KMedoids++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 2$	0.6246	0.6246	0.6246	0.6246	0.6246	0.6246	0.6246	0.6246
$k = 3$	0.6246	0.6246	0.6246	0.6246	0.6246	0.6275	**0.6286**	0.6256
$k = 10$	0.6266	0.6581	0.6384	0.6483	0.6246	**0.6591**	0.6493	0.6435
$k = 23$	0.6344	0.6433	0.6404	0.6493	0.6246	**0.6661**	0.6414	0.6428
$k = 51$	0.6414	0.6970	0.6364	0.7025	0.6246	0.6709	0.6857	**0.6655**
povp.	0.6303	0.6495	0.6329	**0.6499**	0.6246	0.6496	0.6456	*0.6404*

Tabela 7.8: Tabela izračunanih *čistoč* na podatkovni zbirki *besedila* z uporabo reprezentacije *bigramov*.

Evkl. r.	*Cos. r.*	*P. k. k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*
0.6306	0.6494	0.6295	**0.6527**	0.6283	0.6453	0.6483

Tabela 7.9: Povprečja čistoč glede na uporabljeno razdaljo na podatkovni množici *besedila* z uporabo reprezentacije *bigramov*.

$k = 2$	$k = 3$	$k = 10$	$k = 23$	$k = 51$
0.6258	0.6264	0.6384	0.6436	**0.6689**

Tabela 7.10: Povprečja čistoč glede na izbrani k na podatkovni množici *besedila* z uporabo reprezentacije *bigramov*.

Metoda KMeans

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 2$	0.9530	0.9473	0.9546	**0.9040**	0.9547	0.9348	0.9535	0.9431
$k = 3$	0.9520	0.9471	0.9541	0.9343	0.9514	**0.9328**	0.9446	0.9452
$k = 10$	0.9502	0.9313	0.9496	0.9088	0.9093	0.8986	**0.8717**	0.9171
$k = 23$	0.9264	**0.8989**	0.9392	0.8805	0.9433	0.9442	0.9189	0.9216
$k = 51$	0.9017	0.8522	0.9236	**0.8462**	0.9163	0.8536	0.8519	**0.8779**
povp.	0.9367	0.9154	0.9442	**0.8948**	0.9350	0.9128	0.9081	*0.9210*

Metoda KMeans++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 2$	0.9533	0.9486	0.9547	0.9512	0.9546	**0.9123**	0.9399	0.9449
$k = 3$	0.9527	0.9338	0.9500	**0.9101**	0.9492	0.9316	0.9253	0.9361
$k = 10$	0.9445	**0.9192**	0.9492	0.9233	0.9412	0.9218	0.9289	0.9326
$k = 23$	0.9324	**0.8831**	0.9375	0.8901	0.9220	0.9118	0.8851	0.9089
$k = 51$	0.8963	0.8353	0.9225	0.8529	0.9092	0.8310	**0.8303**	**0.8682**
povp.	0.9358	0.9040	0.9428	0.9055	0.9352	**0.9017**	0.9019	***0.9181***

Metoda KMedoids++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 2$	0.9540	0.9550	0.9531	0.9537	0.9547	**0.9514**	0.9542	0.9537
$k = 3$	0.9513	**0.9473**	0.9546	0.9533	0.9546	0.9504	0.9513	0.9518
$k = 10$	0.9425	0.9074	0.9329	**0.9068**	0.9390	0.9126	0.9090	0.9215
$k = 23$	0.9307	0.9045	0.9245	0.8961	0.9409	**0.8919**	0.9057	0.9135
$k = 51$	0.9052	0.8261	0.9086	**0.8294**	0.9276	0.8607	0.8353	**0.8704**
povp.	0.9367	0.9081	0.9347	**0.9079**	0.9434	0.9134	0.9111	*0.9222*

Tabela 7.11: Tabela izračunanih *entropij* na podatkovni zbirki *besedila* z uporabo reprezentacije *bigramov*.

Evkl. r.	*Cos. r.*	*P. k. k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*
0.9364	0.9092	0.9406	**0.9027**	0.9379	0.9093	0.9070

Tabela 7.12: Povprečja entropij glede na uporabljeno razdaljo na podatkovni množici *besedila* z uporabo reprezentacije *bigramov*.

$k = 2$	$k = 3$	$k = 10$	$k = 23$	$k = 51$
0.9472	0.9444	0.9237	0.9147	**0.8722**

Tabela 7.13: Povprečja entropij glede na izbrani k na podatkovni množici *besedila* z uporabo reprezentacije *bigramov*.

7.2 Analiza množice *classic-docs*

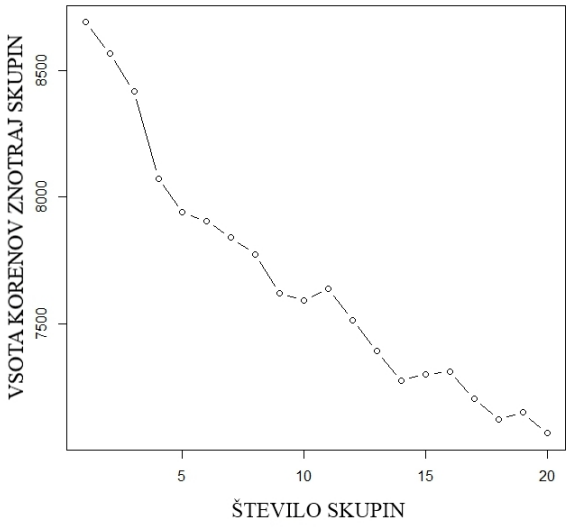
Znano število kategorij podatkovne množice je 4. V tabeli 7.14 so zbrane ocene optimalnega števila skupin z različnimi metodami. Poglejmo si sedaj podrobneje ocene števila k glede na izbrano metodo.

- Število vseh dokumentov znotraj množice je 7095 torej lahko z uporabo metode palca, zapišemo optimalen $k \approx \sqrt{\frac{7095}{2}} \approx 60$
- Matriko vektorjev izrazov podatkovne množiče *classics-docs* sestavlja 7095 vektorjev izrazov dolžine 806. Neničelnih vnosov v matriki je 169942. Oceno za k lahko sedaj zapišemo kot $k \approx \frac{7095 \cdot 806}{169942} \approx 34$.
- Tudi pri tej podatkovni množici težko razberemo iz grafa 7.7 kje je *komolec*. Najbližje *komolcu* je pri vrednosti $k = 4$.
- Z uporabo metode *Silhouette* dobimo grafa, prikazana na slikah 7.8 in 7.9. Iz prvega grafa odčitamo, da je maksimum dosežen pri številu 4, torej izberemo $k = 4$. Na drugem grafu je prikazana razvrstitev besedil podatkovne množice v 4 skupine.
- Z uporabo metode *BIC* dobimo grafa, prikazana na slikah 7.10 in 7.11. Iz grafov lahko odčitamo, da nam tudi ta metoda svetuje uporabo 3 skupin.
- Z izračunom statistike "gap" dobimo graf 7.12. Iz pridobljenega grafa ne moremo odčitati, kateri k je optimalen, saj graf narašča.

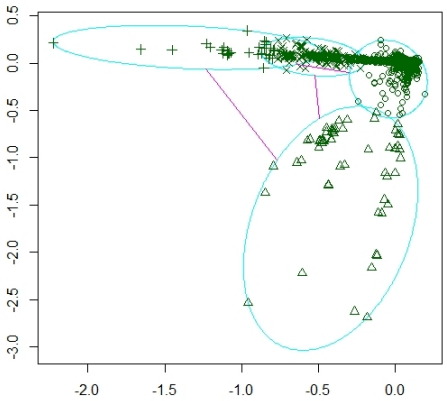
Izračunane entropije in čistoče pri uporabi metod *KMeans, KMeans++* in *KMedoids++*, v kombinaciji z merami razdalj, opisanimi v poglavju 4, pri različnem številu skupin, so zbrane v tabelah 7.15, 7.18, 7.21 in 7.24.

metoda	k
metoda palca	60
metoda 2	34
metoda komolca	4
metoda *silhouette*	4
metoda BIC	4
statistika "gap"	/

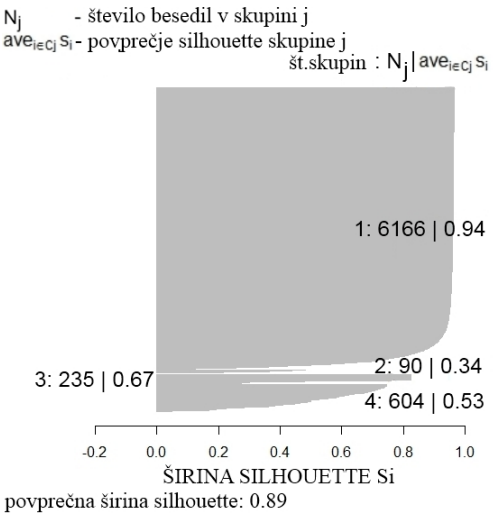
Tabela 7.14: Vrednosti k pri različnih metodah.



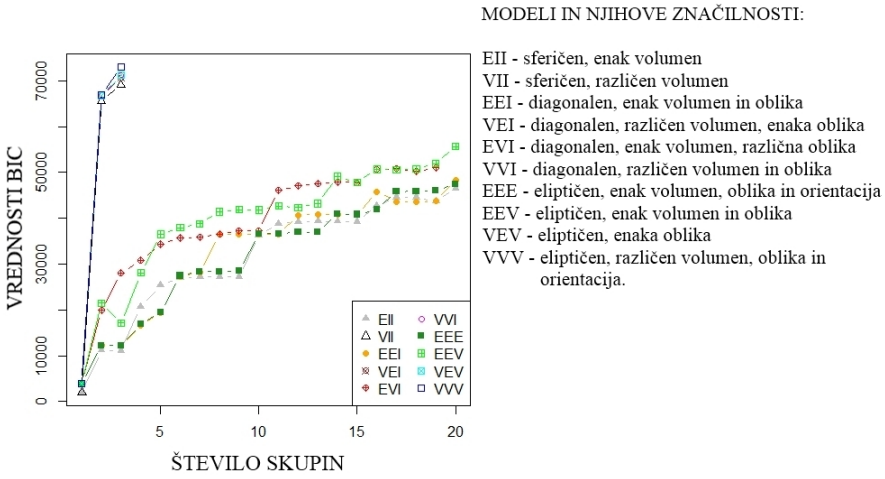
Slika 7.7: Prikaz podatkov, potrebnih za določitev števila skupin po metodi komolca.



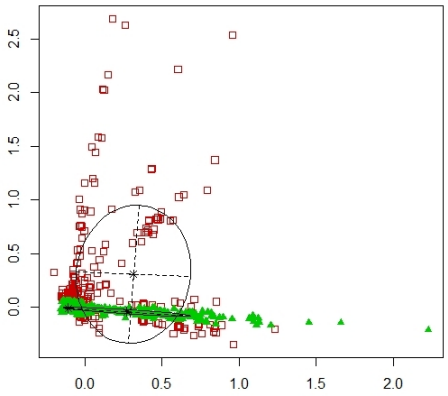
Slika 7.8: Graf prikazuje delitev podatkovne množice v 4 skupine z uporabo metode *silhouette*. Podatki visoko-dimenzionalnega prostora, ki ga določa matrika vektorjev izrazov, množice *classic-docs*, so preslikani v 2-dimenzionalni prostor za lažjo predstavitev.



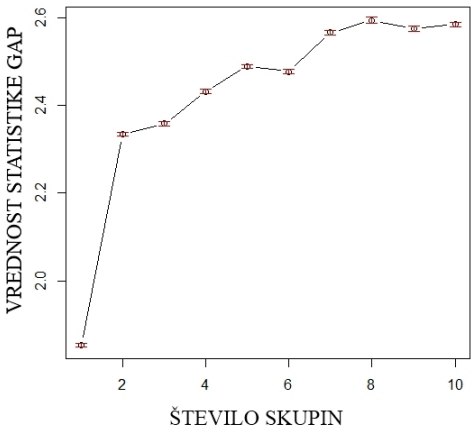
Slika 7.9: Graf prikazuje prileganje elementov v izbrane skupine.



Slika 7.10: Graf prikazuje *BIC* vrednosti pri različnem številu skupin k.



Slika 7.11: Graf prikazuje razvrstitev besedil podatkovne množice v 3 skupine, izbrane s pomočjo metode *BIC*. Podatki visoko-dimenzionalnega prostora, ki ga določa matrika vektorjev izrazov, množice *classic-docs*, so preslikani v 2-dimenzionalni prostor za lažjo predstavitev.



Slika 7.12: Graf prikazuje izračunane statistike “gap” pri različnem številu skupin.

Metoda KMeans

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.r*	*povp.*
$k = 3$	0.4516	0.4516	0.4640	**0.7393**	0.6221	0.6270	0.5834	0.5627
$k = 4$	0.4516	0.7531	0.4543	0.7805	0.7354	0.7442	**0.7937**	0.6733
$k = 34$	0.8124	0.9104	0.4655	**0.9136**	0.4630	0.6058	0.8799	**0.7215**
$k = 60$	0.7786	0.9026	0.4786	**0.9141**	0.4651	0.5253	0.8768	0.7059
povp.	0.6236	0.7544	0.4656	**0.8369**	0.7431	0.6256	0.7834	***0.6659***

Metoda KMeans++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 3$	0.4516	0.4516	0.4516	**0.7841**	0.6155	0.5762	0.7500	0.5829
$k = 4$	0.4516	0.6164	0.4516	0.7777	0.7449	0.7507	**0.8681**	0.6659
$k = 34$	0.8406	0.8947	0.4744	**0.9228**	0.4644	0.5752	0.8906	**0.7232**
$k = 60$	0.7935	0.8771	0.4503	**0.9060**	0.4150	0.4979	0.8235	0.6805
povp.	0.6343	0.7100	0.4570	**0.8477**	0.5600	0.6000	0.8331	*0.6631*

Metoda KMedoids++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 3$	0.4516	0.4516	0.4602	**0.5356**	0.4516	0.4516	0.4516	0.4648
$k = 4$	0.4516	**0.5782**	0.4569	0.5395	0.4516	0.4516	0.4516	0.4830
$k = 34$	0.4891	0.7886	0.4744	**0.8293**	0.4516	0.4971	0.7106	**0.6058**
$k = 60$	0.4515	**0.7791**	0.4838	0.7037	0.4266	0.5074	0.6831	0.5765
povp.	0.4610	0.6494	0.4688	**0.6520**	0.4454	0.4769	0.5742	*0.5325*

Tabela 7.15: Tabela izračunanih *čistoč* na podatkovni zbirki *classic-docs* z uporabo reprezentacije *vreče besed.*

Evkl. r.	*Cos. r.*	*P. k. k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*
0.5730	0.7046	0.4638	**0.7789**	0.5828	0.5675	0.7302

Tabela 7.16: Povprečja čistoč glede na uporabljeno razdaljo na podatkovni množici *classic-docs* z uporabo reprezentacije *vreče besed.*

$k = 3$	$k = 4$	$k = 34$	$k = 60$
0.5701	0.6074	**0.6835**	0.6543

Tabela 7.17: Povprečja čistoč glede na izbrani k na podatkovni množici *classic-docs* z uporabo reprezentacije *vreče besed*.

Metoda KMeans

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 3$	0.8515	0.7626	0.8360	**0.4401**	0.5633	0.5473	0.5441	0.6493
$k = 4$	0.8495	0.4180	0.8233	**0.3144**	0.4170	0.3928	0.3928	0.5154
$k = 34$	0.3735	0.2236	0.8502	**0.1989**	0.8246	0.7019	0.2565	**0.4899**
$k = 60$	0.3724	0.2218	0.8359	**0.1811**	0.8178	0.7690	0.2596	0.4939
povp.	0.6117	0.4072	0.8364	**0.2836**	0.6557	0.6028	0.3633	***0.5372***

Metoda KMeans++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 3$	0.8482	0.8211	0.9073	**0.4138**	0.5777	0.5557	0.4214	0.7326
$k = 4$	0.9212	0.5205	0.8958	0.4269	0.4580	0.3856	**0.2704**	0.5541
$k = 34$	0.3059	0.2167	0.8360	**0.1868**	0.8303	0.7067	0.2487	0.4759
$k = 60$	0.3271	0.2259	0.8200	**0.1851**	0.8399	0.7532	0.2500	0.4859
povp.	0.6006	0.4461	0.8648	0.3032	0.6765	0.6003	**0.2976**	*0.5413*

Metoda KMedoids++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 3$	0.9239	0.8233	0.8885	**0.7780**	0.9223	0.9042	0.8780	0.8740
$k = 4$	0.9034	**0.6820**	0.9011	0.6372	0.8334	0.9125	0.8734	0.8204
$k = 34$	0.7951	0.4331	0.8227	**0.3561**	0.8334	0.7441	0.5520	0.6481
$k = 60$	0.4555	**0.4371**	0.8395	0.4418	0.8470	0.7090	0.5197	**0.6071**
povp.	0.7695	0.5939	0.8630	**0.5533**	0.8590	0.8175	0.7058	*0.7374*

Tabela 7.18: Tabela izračunanih entropij na podatkovni zbirki *classic-docs* z uporabo reprezentacije *vreče besed*.

Evkl. r.	*Cos. r.*	*P. k. k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*
0.6606	0.4824	0.8547	**0.3800**	0.7304	0.6735	0.4556

Tabela 7.19: Povprečja entropij glede na uporabljeno razdaljo na podatkovni množici *classic-docs* z uporabo reprezentacije *vreče besed*.

$k = 3$	$k = 4$	$k = 34$	$k = 60$
0.7520	0.6300	0.5380	**0.5290**

Tabela 7.20: Povprečja entropij glede na izbrani k na podatkovni množici *classic-docs* z uporabo reprezentacije *vreče besed*.

Metoda KMeans

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 3$	0.4882	0.4905	0.4515	0.5246	0.4516	**0.5608**	0.5271	0.4992
$k = 4$	0.5456	0.4674	0.4536	0.4661	0.5825	0.4516	**0.6416**	0.5155
$k = 34$	0.5440	**0.7590**	0.4815	0.7450	0.5618	0.6975	0.7264	0.6450
$k = 60$	0.5300	**0.7677**	0.4833	0.7390	0.5917	0.7184	0.7405	**0.6529**
povp.	0.5270	0.6212	0.4675	0.6187	0.5469	0.6071	**0.6589**	***0.5782***

Metoda KMeans++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 3$	0.4516	0.4586	0.4516	0.4595	0.5870	**0.6085**	0.5760	0.5133
$k = 4$	0.5366	0.4660	0.4566	**0.6674**	0.4610	0.6547	0.4658	0.5596
$k = 34$	0.4974	0.7040	0.4731	0.6374	0.5690	0.4516	**0.7180**	0.5786
$k = 60$	0.5607	**0.7433**	0.4757	0.5698	0.5920	0.4926	0.7393	**0.5962**
povp.	0.5116	0.5930	0.4643	0.5835	0.5523	0.5519	**0.6248**	*0.5619*

Metoda KMedoids++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 3$	0.4533	0.4516	0.4516	**0.4579**	0.4550	0.4554	0.4516	0.4538
$k = 4$	0.4564	0.4582	0.4569	0.4582	0.4561	**0.5636**	0.5106	**0.5490**
$k = 34$	0.4872	0.5691	0.4530	0.4792	0.5481	0.4516	**0.5710**	0.5085
$k = 60$	0.4872	**0.6390**	0.4734	0.4516	0.6050	0.4588	0.6113	0.5323
povp.	0.4710	0.5295	0.4587	0.4617	0.5161	0.4823	**0.5361**	*0.5109*

Tabela 7.21: Tabela izračunanih *čistoč* na podatkovni zbirki *classic-docs* z uporabo reprezentacije *bitermov*.

Evkl. r.	*Cos. r.*	*P. k. k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*
0.5032	0.5812	0.4635	0.5546	0.5384	0.5471	**0.6066**

Tabela 7.22: Povprečja čistoč glede na uporabljeno razdaljo na podatkovni množici *classic-docs* z uporabo reprezentacije *bitermov*.

$k = 3$	$k = 4$	$k = 34$	$k = 60$
0.4888	0.5414	0.5774	**0.5938**

Tabela 7.23: Povprečja čistoč glede na izbrani k na podatkovni množici *classic-docs* z uporabo reprezentacije *bitermov*.

Metoda KMeans

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 3$	0.8809	0.7935	0.8239	0.7907	0.8219	**0.7748**	0.8023	0.8126
$k = 4$	0.8038	0.8297	0.8900	0.8250	0.7407	0.9268	**0.6469**	0.8090
$k = 34$	0.7888	**0.4837**	0.8581	0.4895	0.7653	0.5611	0.5162	0.6375
$k = 60$	0.8130	**0.4727**	0.8322	0.5046	0.7164	0.5585	0.4994	**0.7218**
povp.	0.8216	0.6449	0.8511	0.6525	0.7611	0.7053	**0.6162**	***0.7071***

Metoda KMeans++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 3$	0.9249	0.8227	0.9018	0.8146	0.7327	**0.6581**	0.7447	0.7999
$k = 4$	0.8156	0.8265	0.8966	**0.6233**	0.8194	0.6431	0.8132	0.7768
$k = 34$	0.8439	0.5532	0.8629	0.6376	0.7555	0.9268	**0.5416**	0.7359
$k = 60$	0.7592	0.5094	0.8484	0.7288	0.7150	0.7971	**0.4972**	**0.6936**
povp.	0.8359	**0.6780**	0.8774	0.7011	0.7557	0.7563	0.7117	*0.7516*

Metoda KMedoids++

št.skup.	*Evkl.r.*	*Cos.r.*	*P.k.k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*	*povp.*
$k = 3$	0.9212	0.9235	0.9251	**0.9100**	0.9201	0.9109	0.9109	0.9174
$k = 4$	0.9157	0.9200	0.8701	0.9160	0.9126	**0.7780**	0.8415	0.8791
$k = 34$	0.8650	0.7666	0.8710	0.8488	0.7931	0.9207	**0.7400**	**0.8293**
$k = 60$	0.8559	**0.6798**	0.8541	0.9210	0.7230	0.9173	0.6968	0.8795
povp.	0.8895	0.8225	0.8801	0.8990	0.8372	0.8818	**0.7973**	*0.8763*

Tabela 7.24: Tabela izračunanih entropij na podatkovni zbirki *classic-docs* z uporabo reprezentacije *bitermov*.

Evkl. r.	*Cos. r.*	*P. k. k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*
0.8490	0.7151	0.8695	0.7509	0.7847	0.7811	**0.7084**

Tabela 7.25: Povprečja entropij glede na uporabljeno razdaljo na podatkovni množici *classic-docs* z uporabo reprezentacije *bitermov*.

$k = 3$	$k = 4$	$k = 34$	$k = 60$
0.8433	0.8216	**0.7342**	0.7650

Tabela 7.26: Povprečja entropij glede na izbrani k na podatkovni množici *classic-docs* z uporabo reprezentacije *bitermov*.

7.3 Analiza

V tem poglavju bomo najprej analizirali rezultate, pridobljene na podatkovni množici *besedila*, in nato še rezultate podatkovne množice *classic-docs*. Analizo bomo zaključili s posplošitvijo ugotovitev.

besedila

- Najprej si poglejmo, kateri tip reprezentacije podatkovne množice je najbolje uporabiti. V tabeli 7.27 so zbrane povprečne čistoče in entropije glede na uporabljeno reprezentacijo. Vidimo lahko, da *vreča besed* doseže boljše rezultate. Iz tabele 7.28 lahko odčitamo, da statistična razlika rezultatov obeh reprezentacij (na podlagi *t-testa* [1]) ni statistično značilna.

- Sedaj si poglejmo, katero razvrščevalno metodo je najbolje uporabiti. V tabeli 7.29 sta zbrani povprečna čistoča in entropija glede na uporabljeno razvrščevalno metodo. Vidimo lahko, da ima metoda *KMeans* boljšo povprečno čistočo in entropijo kot preostali metodi. Zanima nas še, če je metoda statistično značilna z ostalimi, kar preverimo s *t-testom.* V tabeli 7.30 so zbrane

[1] Pri *Studentovem t-testu* primerjamo dve hipotezi — začetno hipotezo proti alternativni hipotezi. Naš cilj je, da bi začetno hipotezo zavrnili. V našem eksperimentu bomo uporabili dvostranski *t-test*. Za začetno hipotezo bomo vzeli razliko opazovanih elementov. Rezultat je *p-vrednost*, ki nam pove najmanjšo stopnjo značilnosti, pri kateri hipotezo še lahko zavrnemo. Če je $p < 0.1$ rečemo, da je zač. hipoteza statistično značilna in jo zavrnemo, če je $p < 0.05$ rečemo, da je zač. hipoteza statistično zelo značilna in jo zavrnemo. [Wik13b]

p-vrednosti *t-testa* parov metod. Opazimo lahko, da rezultat metode *KMeans* statistično značilno odstopa od rezultatov metode *KMedoids++*. Razlika rezultatov metod *KMeans* in *KMeans++* ni statistično značilna.

- Izberimo sedaj najustreznejšo razdaljo. V tabeli 7.31 so zbrane povprečne čistoče in entropije glede na uporabljeno razdaljo. Vidimo, da razdalja *Bray-Curtis* zavzame najboljšo povprečno čistočo in entropijo in da je značilno boljša od *Evklidske* razdalje, *Kosinusne* razdalje, razdalje *Pearsonovega korelacijskega koeficienta*, *Jeffreyjeve* razdalje in razdalje *Canberra*, od *Hellingerjeve* razdalje pa ne.
- Sedaj želimo izbrati še najboljše število skupin (k). V tabeli 7.33 so prikazane povprečne čistoče in entropije glede na izbrano število skupin. V povprečju dosežemo najboljše rezultate pri $k = 51$. V tabeli 7.34 so zbrani rezultati *t-testa* skupine $k = 51$ proti ostalim skupinam. Vidimo lahko, da je rezultat v celoti (pri entropiji in čistoči) značilno boljši od rezultata pri $k = 10$, od ostalih rezultatov pa ne.

	Vreča besed	*Bitermi*
čistoča	**0.6798**	0.4060
entropija	**0.8568**	0.9204

Tabela 7.27: Povprečja čistoče in entropije glede na uporabljeno reprezentacijo za množico *besedila*.

	t-test p
čistoča	0.1411
entropija	0.1736

Tabela 7.28: p-vrednost *t-testa* za primerjavo čistoče in entropije med uporabljenima reprezentacijama za množico *besedila*.

	KMeans	*KMeans++*	*KMedoids++*
čistoča	**0.6737**	0.6616	0.6453
entropija	**0.8626**	0.8877	0.9156

Tabela 7.29: Povprečja čistoče in entropije glede na uporabljene metode razvrščanja za množico *besedila*.

	KMeans : KMeans++ (p-vrednost)	*KMeans : KMedoids++ (p-vrednost)*
čistoča	0.3870	0.0730
entropija	0.2407	0.0269

Tabela 7.30: p-vrednosti *t-testa* za primerjavo čistoče in entropije glede na uporabljene metode razvrščanja za množico *besedila*.

	Evkl. r.	*Cos. r.*	*P. k. k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*
čistoča	0.6520	0.6772	0.6490	**0.6817**	0.6382	0.6446	0.6788
entropija	0.8990	0.8673	0.9171	**0.8548**	0.9342	0.9097	0.8628

Tabela 7.31: Povprečje čistoč in entropij glede na uporabljeno razdaljo na podatkovni množici *besedila*

	Evkl. r.	*Cos. r.*	*P. k. k.*	*Jeff.*	*Canb.*	*Hell.*
čistoča	0.0005	0.0671	0.0034	0.0085	0.0449	0.3537
entropija	0.0150	0.0701	0.0051	0.0131	0.0735	0.1387

Tabela 7.32: p-vrednost *t-testa* za primerjavo razdalje *Bray-Curtis* v primerjavi z ostalimi glede na čistočo in entropijo na podatkovni množici *besedila*

	$k = 2$	$k = 3$	$k = 10$	$k = 23$	$k = 51$
čistoča	0.6539	0.6850	0.6583	0.6652	*0.6781*
entropija	0.9039	0.9005	0.8980	0.8826	*0.8583*

Tabela 7.33: Povprečje čistoče in entropije glede na število skupin za množico *besedila*.

	$k = 2$	$k = 3$	$k = 10$	$k = 23$
čistoča	0.2973	0.0028	0.0776	0.1111
entropija	0.1657	0.1741	0.0234	0.0981

Tabela 7.34: p-vrednosti *t-testa* za primerjavo čistoče in entropije s parametrom $k = 23$ proti ostalim številom skupin za množico *besedila*.

classic-docs

- Zopet si najprej pogljemo, kateri tip reprezentacije podatkovne množice je najbolje uporabiti. V tabeli 7.35 so zbrane povprečne čistoče in entropije glede na uporabljeno reprezentacijo. Vidimo lahko, da *vreča besed* doseže boljše rezultate. Iz tabele 7.36 lahko odčitamo, da statistična razlika rezultatov obeh reprezentacij ni statistično značilna.

- Sedaj si poglejmo, katero razvrščevalno metodo je najbolje uporabiti. V tabeli 7.37 sta zbrani povprečna čistoča in entropija glede na uporabljeno razvrščevalno metodo. Vidimo lahko, da ima metoda *KMeans* boljšo povprečno čistočo in entropijo kot preostali metodi. Zanima nas še, če je metoda statistično značilna z ostalimi, kar preverimo s *t-testom.* V tabeli 7.38 so zbrane *p-vrednosti t-testa* parov metod. Opazimo lahko, da rezultat metode *KMeans* statistično močno značilno odstopa od rezultata metode *KMedoids++* in statistično značilno odstopa od rezultata metode *KMeans++*.

- Izberimo sedaj najustreznejšo razdaljo. V tabeli 7.31 so zbrane povprečne čistoče in entropije glede na uporabljeno razdaljo. Vidimo, da razdalja *Bray-Curtis* zavzame najboljšo povprečno čistočo in entropijo. Rezultati razdalje *Bray-Curtis* ne statistično odstopaja od rezultatov ostalih razdalj.

- Sedaj želimo izbrati še najboljše število skupin (k). V tabeli 7.41 so prikazane povprečne čistoče in entropije glede na izbrano število skupin. V povprečju dosežemo najboljše rezultate pri $k = 34$. V tabeli 7.42 so zbrani rezultati *t-testa* skupine $k = 34$ proti ostalim skupinam. Rezultati skupine $k = 34$ statistično značilno odstopajo od rezultatov skupin $k = 3$ in $k = 4$, od ostalih pa ne.

	Vreča besed	*Bitermi*
čistoča	**0.6205**	0.5503
entropija	**0.6053**	0.7783

Tabela 7.35: Povprečja čistoče in entropije glede na uporabljeno reprezentacijo za množico *classic-docs*.

	t-test p
čistoča	0.1040
entropija	0.0140

Tabela 7.36: p-vrednost *t-testa* za primerjavo čistoče in entropije med uporabljenima reprezentacijama za množico *classic-docs*.

	KMeans	*KMeans++*	*KMedoids++*
čistoča	**0.6221**	0.6125	0.5217
entropija	**0.6222**	0.6465	0.8069

Tabela 7.37: Povprečja čistoče in entropije glede na uporabljene metode razvrščanja za množico *classic-docs*.

	KMeans : KMeans++ (p-vrednost)	*KMeans : KMedoids++ (p-vrednost)*
čistoča	0.0060	$5.22 \cdot e^{-5}$
entropija	0.0539	$1.122 \cdot e^{-5}$

Tabela 7.38: p-vrednosti *t-testa* za primerjavo čistoče in entropije glede na uporabljene metode razvrščanja za množico *classic-docs*.

	Evkl. r.	*Cos. r.*	*P. k. k.*	*B.C.*	*Jeff.*	*Canb.*	*Hell.*
čistoča	0.5943	0.6572	0.5558	**0.6712**	0.5991	0.5960	0.6711
entropija	0.8291	0.7375	0.8910	**0.7147**	0.8471	0.7973	0.7242

Tabela 7.39: Povprečje čistoč in entropij glede na uporabljeno razdaljo na podatkovni množici *classic-docs*

	Evkl. r.	*Cos. r.*	*P. k. k.*	*Jeff.*	*Canb.*	*Hell.*
čistoča	0.1752	0.5498	0.1890	0.1887	0.2078	0.9875
entropija	0.1405	0.4901	0.1804	0.1804	0.1476	0.6575

Tabela 7.40: p-vrednosti *t-test* za primerjavo razdalje *Bray-Curtis* v primerjavi z ostalimi glede na čistočo in entropijo na podatkovni množici *classic-docs*

	$k = 3$	$k = 4$	$k = 34$	$k = 60$
čistoča	0.5129	0.5744	**0.6304**	0.6241
entropija	0.7976	0.7258	**0.6378**	0.6470

Tabela 7.41: Povprečje čistoče in entropije glede na število skupin za množico *classic-docs*.

	$k = 3$	$k = 4$	$k = 60$
čistoča	0.0014	0.085	0.5863
entropija	0.0040	0.0242	0.6842

Tabela 7.42: p-vrednosti *t-testa* za primerjavo čistoče in entropije s parametrom $k = 34$ proti ostalim številom skupin za množico *classic-docs*.

Poglavje 8

Sklep

V delu smo se ukvarjali s problemom analize podobnosti besedil. Za rešitev tega problema smo uporabili pristop z nenadzorovanim strojnim učenjem. Za to smo potrebovali primerne algoritme razvrščanja, ki so *KMeans*, *KMeans++* in *KMedoids++*. Delovanje vsakega od naštetih algoritmov je močno odvisno od mere podobnosti in predvidenega števila skupin. Za določitev optimalnega števila predvidenih skupin, smo uporabili šest različno zahtevnih metod. Podobnost med elementi smo merili s pomočjo sedmih mer razdalje. Z uporabo dveh podatkovnih množic, ki smo jih predstavili na dva načina, smo želeli ugotoviti najboljšo kombinacijo algoritma za razvrščanje, mere razdalj in predvidenega števila skupin.
Iz meritev, opravljenih v prejšnjem poglavju, smo prišli do ugotovitev, da je podatkovno množico bolje predstaviti z *vrečo besed* kot pa z *bitermi*. Za razvrščevalni algoritem je najbolje izbrati *KMeans* v povezavi z *Bray-Curtisovo* neenakostjo, kot mero razdalje. Število skupin lahko izbiramo na dva načina. Če želimo ugotoviti, kateri k je najbližje dejanskemu številu kategorij, potem izberemo metodo *BIC*. V primeru, da želimo izbrati k, pri katerem bomo dosegli najboljše rezultate, pa izberemo raje metodo iskanja števil skupin tekstovnih množic.

Literatura

[AV06] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Proceedings of the twenty-second annual symposium on Computational geometry*, SCG '06, pages 144–153, New York, NY, USA, 2006. ACM.

[bag] Bag-of-words model. `http://en.wikipedia.org/wiki/Bag-of-words_model`. Dostopno: 08-09-2013.

[Bea] Beautiful soup. `http://www.crummy.com/software/BeautifulSoup/`. Dostopno: 08-09-2013.

[Can] Canberradistance. `http://reference.wolfram.com/mathematica/ref/CanberraDistance.html`. Dostopno: 08-09-2013.

[CB05] Mari Ostendorf Constantinos Boulis. Text classification by augmenting the bag-of-words representation with redundancy-compensated bigrams. 2005.

[CBKM$^+$] Kurt Hornik Christian Buchta, Bettina Grun, Robert Koblischke, David Meyer, Stefan Theussl, and Fridolin Wild. tm - text mining package. `http://tm.r-forge.r-project.org/`. Dostopno: 08-09-2013.

[Cha07] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions, 2007.

[clua] Document clustering. `http://en.wikipedia.org/wiki/Document_clustering`. Dostopno: 08-09-2013.

[club] Quick-r: Cluster analysis. `http://www.statmethods.net/advstats/cluster.html`. Dostopno: 08-09-2013.

[Cos] Cosine similarity. `http://en.wikipedia.org/wiki/Cosine_similarity`. Dostopno: 08-09-2013.

[CP05] Xiaohui Cui and Thomas E. Potok. Document clustering analysis based on hybrid pso+k-means algorithm. *Special Issue*, pages 27–33, 2005.

[Dek09] Alexander Dekhtyar. Distance/similarity measures. http://users.csc.calpoly.edu/~dekhtyar/560-Fall2009/lectures/lec09.466.pdf, 2009. Dostopno: 08-09-2013.

[dis] Distance matrix computation. http://stat.ethz.ch/R-manual/R-patched/library/stats/html/dist.html. Dostopno: 08-09-2013.

[DM] Christian Buchta David Meyer. Matrix distance/similarity computation. http://rss.acs.unt.edu/Rdoc/library/proxy/html/dist.html. Dostopno: 08-09-2013.

[doc] Document-term matrix. http://en.wikipedia.org/wiki/Document-term_matrix. Dostopno: 08-09-2013.

[DP00] Andrew Moore Dan Pelleg. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, San Francisco, 2000. Morgan Kaufmann.

[ELO08] Tamer Elsayed, Jimmy Lin, and Douglas W. Oard. Pairwise document similarity in large collections with mapreduce, 2008.

[eva] Evaluation of clustering. http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html. Dostopno: 08-09-2013.

[Evk] Euclidean distance. http://en.wikipedia.org/wiki/Euclidean_distance. Dostopno: 08-09-2013.

[FGA07] Daniel M. Ennis F. Gregory Ashby. Similarity measures. http://www.scholarpedia.org/article/Similarity_measures, 2007. Dostopno: 08-09-2013.

[FWH+08] Ingo Feinerer, Wirtschaftsuniversität Wien, Kurt Hornik, Wirtschaftsuniversität Wien, David Meyer, and Wirtschaftsuniversität Wien. Text mining infrastructure in r. *Journal of Statistical Software*, 2008.

[Gly05] Earl F. Glynn. Correlation "distances"and hierarchical clustering. http://research.stowers-institute.org/efg/R/Visualization/cor-cluster/, 2005. Dostopno: 08-09-2013.

[Gos] Sarah Goslee. Bray-curtis distance. http://rss.acs.unt.edu/Rdoc/library/ecodist/html/bcdist.html. Dostopno: 08-09-2013.

[Gre] Michael Greenacre. Measures of distance between samples: non-euclidean. http://www.econ.upf.edu/~michael/stanford/maeb5.pdf. Dostopno: 08-09-2013.

[Hel] Hellinger distance. http://en.wikipedia.org/wiki/Hellinger_distance. Dostopno: 08-09-2013.

[HMFW] Lan Huang, David Milne, Eibe Frank, and Ian H. Witten. Learning a concept-based document similarity measure.

[Hua08] Anna Huang. Similarity measures for text document clustering, 2008.

[kmea] K-means++. http://en.wikipedia.org/wiki/K-means%2B%2B. Dostopno: 08-09-2013.

[kmeb] k-medoids. http://en.wikipedia.org/wiki/K-medoids. Dostopno: 08-09-2013.

[LPW] Michael D. Lee, Brandon Pincombe, and Matthew Welsh. A comparison of machine measures of text document similarity with human judgments.

[MDM07] Donald Metzler, Susan Dumais, and Christopher Meek. Similarity measures for short segments of text. In *In Proc. of ECIR-07*, 2007.

[min] Data mining algorithms in r/clustering/k-means. http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/K-Means. Dostopno: 08-09-2013.

[Mon] Monte carlo method. http://en.wikipedia.org/wiki/Monte_Carlo_methods. Dostopno: 08-09-2013.

[MSMT+07] Roger Zhang Mahdi Shafiei, Singer Wang, Evangelos Milios, Bin Tang, Jane Tougas, and Ray Spiteri. Document representation and dimension reduction for text clustering. *Data Engeneering Workshop, 2007 IEEE 23rd International Conference on*, pages 770–779, 2007.

[ngr] n-gram. http://en.wikipedia.org/wiki/N-gram. Dostopno: 08-09-2013.

[NPp] Np-polnost. http://wiki.fmf.uni-lj.si/wiki/NP-polnost. Dostopno: 08-09-2013.

[ORSS06] Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *In 47th IEEE Symposium on the Foundations of Computer Science (FOCS*, pages 165–176, 2006.

[Pea] Pearson product-moment correlation coefficient. http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient. Dostopno: 08-09-2013.

[pora] The porter stemming algorithm. http://tartarus.org/martin/PorterStemmer/. Dostopno: 08-09-2013.

[Porb] Martin Porter. Snowball. http://snowball.tartarus.org/. Dostopno: 08-09-2013.

[pro] The r project for statistical computing. http://www.r-project.org/. Dostopno: 10-09-2013.

[RS13] Muhammad Rafi and Mohammad Shahid Shaikh. An improved semantic similarity measure for document clustering based on topic maps. *CoRR*, abs/1303.4087, 2013.

[RWe] Rweka: R/weka interface. http://cran.r-project.org/web/packages/RWeka/index.html. Dostopno: 10-09-2013.

[Sch07] Jan Schulz. Bray-curtis dissimilarity. http://www.code10.info/index.php?option=com_content&view=article&id=46:articlebray-curtis-dissim&catid=38:cat_coding_algorithms_data-similarity&Itemid=57, 2007. Dostopno: 08-09-2013.

[SKK00] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *In KDD Workshop on Text Mining*, 2000.

[sma] Stop word list - smart. http://www.lextek.com/manuals/onix/stopwords2.html. Dostopno: 08-09-2013.

[sto] Stop words. http://en.wikipedia.org/wiki/Stop_words. Dostopno: 08-09-2013.

[Sup10] SB Support. Catalan stop-words. http://dnnspeedblog.com/SpeedBlog/PostID/3181/Catalan-Stop-words, 2010. Dostopno: 08-09-2013.

[TG] Kumar Tan, Steinbach and Ghosh. The k-means algorithm. http://www.cs.uvm.edu/~xwu/kdd/Slides/Kmeans-ICDM06.pdf. Dostopno: 08-09-2013.

[Tib] Ryan Tibshirani. Clustering 1: K-means, k-medoids. http://www.stat.cmu.edu/~ryantibs/datamining/lectures/04-clus1-marked.pdf. Dostopno: 08-09-2013.

[Tri13] Craig Trim. Tf/idf with google n-grams and pos tags. http://trimc-nlp.blogspot.com/2013/04/tfidf-with-google-n-grams-and-pos-tags.html, 2013. Dostopno: 08-09-2013.

[Tun10] Volkan Tunali. Classic3 and classic4 datasets. http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets/, 2010. Dostopno: 08-09-2013.

[Vor] Voronoi diagram. http://en.wikipedia.org/wiki/Voronoi_diagram. Dostopno: 08-09-2013.

[Wae11] Daniel Waegel. The porter stemmer. http://www.eecis.udel.edu/~trnka/CISC889-11S/lectures/dan-porters.pdf, 2011. Dostopno: 08-09-2013.

[Wik12] Wikipedia. Peter rousseeuw — wikipedia, the free encyclopedia, 2012. [Online; accessed 14-September-2013].

[Wik13a] Wikipedia. Determining the number of clusters in a data set — wikipedia, the free encyclopedia, 2013. [Online; accessed 16-September-2013].

[Wik13b] Wikipedia. Student's t-test — wikipedia, the free encyclopedia, 2013. [Online; accessed 14-September-2013].

[wor] Word cloud in r. http://onertipaday.blogspot.com/2011/07/word-cloud-in-r.html. Dostopno: 08-09-2013.

[ZK02] Ying Zhao and George Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Data Mining and Knowledge Discovery*, pages 515–524. ACM Press, 2002.

Slike

Tabele

Algoritmi