Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Rok Rejc

Generiranje slovenskih besednih oblik s pomočjo strojnega učenja

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: izr. prof. dr. Marko Robnik Šikonja
Somentor: dr. Simon Krek

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Leksikon besednih oblik je eden temeljnih jezikovnih virov, ki služi pri več nalogah analize in razumevanja besedil. Zaradi stalnega spreminjanja jezika je nujno njegovo stalno posodabljanje, pri čemer nove besede zajemamo iz referenčnih korpusov. Razvijte prototip orodja, ki bo omogočalo avtomatsko širitev leksikona slovenskih besednih oblik Sloleks. Z gručenjem določite obstoječe besedotvorne paradigme, potem pa dobljene gruče uporabite za izgradnjo napovednega modela. Pristop ovrednotite na prosto dostopnem korpusu. Preverite smiselnost dobljenih paradigem in klasifikacijsko točnost naučenega modela strojnega učenja. Analizirajte napake in predlagajte njihovo odpravo.

Zahvaljujem se mentorjema izr. prof. dr. Marku Robniku Šikonji ter dr. Simonu Kreku, ki sta me usmerjala pri izdelavi diplomske naloge ter mi nudila strokovno pomoč.

Prav tako se zahvaljujem partnerici Maji ter hčerkama, ki so me podpirale in mi omogočile dokončanje študija. Nenazadnje se zahvaljujem sodelavcem v podjetju Amebis d.o.o., ki so mi omogočili izostanke od dela za študijske obveznosti.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CRAN	Comprehensive R Archive Network	/
CSV	Comma-Separated Values	vrednosti, ločene z vejico
ISO	International Organization for Standardization	Mednarodna organizacija za standardizacijo
JOS	/	Jezikovno označevanje slovenščine
LMF	Lexical Markup Framework	/
MSD	MorphoSyntactic Descriptions	oblikoskladenjska oznaka
PAM	Partitioning Around Medoids	gručenje z medoidi
XML	Extensible Markup Language	razširljivi označevalni jezik

Povzetek

Naslov: Generiranje slovenskih besednih oblik s pomočjo strojnega učenja

Sloleks je leksikon besednih oblik, ki v strukturirani bazi podatkov vsebuje slovenske besede in njihove pregibne oblike, skupaj z informacijo, v katero besedno vrsto spadajo in kakšne so njihove oblikoskladenjske lastnosti. Zaradi stalnega spreminjanja jezika ter naraščajočih potreb po strojnem procesiranju je Sloleks potrebno stalno posodabljati.

Cilj diplomske naloge je bil izdelati orodje, ki bo s strojnim učenjem omogočalo avtomatsko širitev leksikona besednih oblik Sloleks, pri čemer smo se osredotočili na samostalniško besedno vrsto, vendar pa je orodje mogoče uporabiti tudi za druge besedne vrste, kot sta glagol ali pridevnik. Problema smo se lotili z razvrščanjem samostalnikov v skupine s podobnimi oblikoskladenjskimi lastnostmi, pri čemer smo uporabili Gowerjevo razdaljo ter gručenje z medoidi. Na podlagi dobljenih skupin, ki predstavljajo oblikoskladenjske paradigme, smo z naivnim Bayesovim klasifikatorjem zgradili model za napovedovanje teh paradigem tudi za nove besede. Samostalnikom iz korpusa ccGigafida, ki jih manjkajo besedne oblike, smo z zgrajenim klasifikatorjem pripisali skupino ter glede na tipične predstavnike skupine ustrezno dopolnili manjkajoče besedne oblike. Pristop smo ovrednotili kvalitativno ter kvantitativno.

Ključne besede: leksikon besednih oblik, Sloleks, oblikoskladenjske paradigme, strojno učenje, naivni Bayesov klasifikator, gručenje z medoidi.

Abstract

Title: Generating Slovene word forms using machine learning

Sloleks is a lexicon of Slovene word forms which contains - in a structured database - Slovene words and all their word forms, their word class and morphosyntactic properties. Due to constant changing of the language and the growing needs for machine processing, Sloleks must be constantly updated.

The aim of the thesis was to create a tool using machine learning that will allow automated extension of lexicon of Slovene word forms Sloleks. We focused mainly on nouns, but the tool can also be used for other word classes such as verb or adjective. The problem was tackled with clustering of nouns into groups with similar morphosyntactic properties, where we used clustering around medoids. Based on the obtained groups which represent morphosyntactic paradigms, we build a model using naive Bayes classifier which predicts these paradigms for new words. For nouns from corpus ccGigafida, which have missing word forms, we predicted groups using build classifier and filled the paradigm with missing word form using typical representatives of classes. Approach was evaluated qualitatively and quantitatively.

Keywords: lexicon of word forms, Sloleks, morphosyntactic paradigms, machine learning, naive Bayes classifier, Partitioning Around Medoids.

Poglavje 1

Uvod

Opis oblikoslovnih paradigem je pri morfološko bogatih jezikih, kot je slovenščina, zelo pomemben. Opis pregibanja je vključil že Bohorič v svojo slovnico leta 1584 in se od takrat pojavljajo v večini kasnejših slovenskih slovnic, v pravopisnih priročnikih ter slovarjih. Po drugi strani so bili zaradi omejenosti s prostorom v knjigah morfološki podatki navadno močno okrajšani in/ali nepopolni (vsebovali so npr. le nekaj oblik) [3, str. 81].

Z razvojem računalništva in jezikovnih tehnologij so se pojavile potrebe po slovarjih in leksikonih besednih oblik, ki so tudi strojno berljivi in vsebujejo vse besedne oblike ter njihove lastnosti. Sodoben leksikon besednih oblik mora torej omogočati uporabo v jezikovnotehnoloških orodjih (črkovalnikih, slovničnih pregledovalnikih, sintetizatorjih govora, strojnih prevajalnikih itn.), hkrati pa je zaželeno, da so uporabni tudi kot samostojni pripomočki, namenjeni govorcem slovenskega jezika [3, str. 81].

Eden izmed prvih strojno berljivih leksikonov besednih oblik za slovenščino je ASES (Amebisov Skupni Elektronski Slovar), ki od začetka devetdesetih let nastaja v podjetju Amebis, d.o.o., Kamnik. Leksikon ni prosto dostopen, uporabljen pa je v različnih izdelkih podjetja, kot sta npr. slovnični pregledovalnik Besana ali strojni prevajalnik Presis [6, str. 38].

V okviru projekta "Sporazumevanje v slovenskem jeziku"[1], ki sta ga v

[1]Spletna stran projekta: http://www.slovenscina.eu.

letih 2008-2013 financirala Evropski socialni sklad in Ministrstvo za izobraževanje, znanost in šport je nastal sodoben leksikon besednih oblik Sloleks, ki v strukturirani bazi (XML format) vsebuje osnovne podatke o slovenskih besedah, njihovih pregibnih oblikah ter lastnostih [5]. Leksikon vsebuje 100.805 najpogostejših osnovnih oblik (lem) slovenskega besedišča in 2.792.006 posameznih besednih oblik z opisanimi slovničnimi lastnostmi ter pokriva doslej največji delež splošnega besedišča slovenskega jezika [3, str. 93].

1.1 Širitev Sloleksa

Zaradi stalnega spreminjanja jezika sodobne slovenščine in hkrati naraščajočih potreb strojnega procesiranja je nujno stalno posodabljanje leksikona besednih oblik. V okviru diplomske naloge sta pomembna predvsem dva vira posodabljanja. Prvi vir je referenčni korpus slovenskega jezika, ki vsebuje besedišče, ki se po merilih za zajem iztočnic ni uvrstilo v Sloleks, a je zaradi rabe pomembno za razvoj jezikovnih tehnologij. S širitvijo leksikona se lahko izboljša jezikovni model označevalnika ter označevanje korpusa. Drugi vir pa so specializirani deli besedišča, kot je narečno ali strokovno besedje [3, str. 93-94].

Želimo razviti orodje, ki bo omogočalo avtomatsko širitev leksikona besednih oblik Sloleks. V okviru naloge se bomo osredotočili na samostalniško besedno vrsto, vendar pa bo razvit model primeren tudi za druge besedne vrste. Razvoj smo razdelili na tri korake:

1. V prvem koraku bomo s pomočjo nenadzorovanega strojnega učenja samostalnike razvrstili v skupine gesel s podobnimi pregibnimi vzorci ter oblikoskladenjskimi lastnostmi.

2. Skupine bomo v drugem koraku uporabili kot razrede v klasifikacijskem modelu (nadzorovano strojno učenje), ki bo za neznane samostalnike napovedal skupino, v katero spadajo. Skupine bo mogoče napovedati

glede na osnovno obliko samostalnika (lemo) ali glede na osnovno obliko ter eno ali več pripadajočih oblik.

3. V zadnjem koraku bomo z napovedanimi skupinami ustrezno dopolnili manjkajoče podatke v korpusu.

1.2 Pregled vsebine

Diplomsko nalogo smo razdelili na pet poglavij. Drugo poglavje je namenjeno opisu Sloleksa, v njem opišemo njegovo zgradbo ter vsebino. V tretjem poglavju na kratko opišemo uporabljene metode strojnega učenja ter programske jezike in okolja, ki smo jih uporabili pri izdelavi diplomske naloge. V četrtem in osrednjem poglavju je podrobneje opisano orodje za širitev leksikona. Najprej opišemo razvrščanje besed v skupine, nato napovedovanje skupine ter na koncu še metodo za kreiranje oz. dopolnitev paradigme. Razvite metode ovrednotimo glede na ustreznost dobljenih gruč in izmerimo klasifikacijsko točnost razvitega modela. V petem poglavju povzamemo rezultate diplomske naloge ter predlagamo nekaj mogočih izboljšav.

Poglavje 2

Sloleks

Leksikon besednih oblik Sloleks je nastal v okviru projekta "Sporazumevanje v slovenskem jeziku". Leksikon zajema približno 100.805 gesel ter 2.792.006 besednih oblik, ki so združene z njihovimi osnovnimi oblikami (lemami). Zastopanost lem in besednih oblik glede na besedno vrsto je prikazana v tabeli 2.1.

Gesla (leme) so bile zajete v leksikon po naslednjih kriterijih [7][3, str. 83]:

- leme ročno označenega učnega korpusa ssj500k [1],
- vse leme zaprtih besednih vrst (predlog, veznik, zaimek, členek),
- izbrani težji primeri iz oblikoslovja (npr. tuja lastna imena),
- ostale leme so bile izbrane glede na pogostost pojavljanja v korpusu FidaPLUS [2].

Leksikon je zapisan v razširljivem označevalnem jeziku XML, in sicer v formatu LMF (ang. *Lexical Markup Framework*), ki je ISO standard za strojno berljive podatkovne zbirke za procesiranje naravnih jezikov [8].

Oblikoskladenjske lastnosti besednih oblik so v leksikonu zapisane skladno s specifikacijami JOS, ki so bile razvite v okviru projekta Jezikovno označevanje slovenščine [9].

besedna vrsta	**število lem**	**število oblik**
samostalniki	54.260	924.268
pridevniki	26.612	1.571.970
glagoli	10.242	260.826
prislovi	6.906	9.931
števniki	2.240	18.448
zaimki	169	6.182
predlogi	96	101
medmeti	85	85
okrajšave	70	70
členki	68	68
vezniki	54	54
večbesedne enote	3	3
Skupaj	**100.805**	**2.792.006**

Tabela 2.1: Število lem in oblik v leksikonu glede na besedno vrsto.

Oblikoskladenjske lastnosti so zapisane v obliki MSD (ang. *morphosyntactic descriptions*) oznak, kjer prva črka predstavlja besedno vrsto, naslednje črke pa predstavljajo vrednosti dodatnih lastnosti. V tabeli 2.2 so navedene besedne vrste ter njihove lastnosti, pri čemer se besedna vrsta "neuvrščeno" ne pojavlja v leksikonu.

Oblikoskladenjske lastnosti besedne oblike *mizi* so zapisane kot MSD oznaka Sozed ter pomenijo: *samostalnik*, *vrsta=občno ime*, *spol=ženski*, *število=ednina*, *sklon=dajalnik*. Vse možne oblike MSD oznak so naštete v priporočilih za oblikoslovno označevanje JOS [10].

besedna vrsta	**oznaka**	**lastnosti s številom vrednosti**
samostalnik	S	vrsta(2), spol(3), število(3), sklon(6), živost (2)
glagol	G	vrsta(2), vid(3), oblika(7), oseba(3), število(3), spol(3), nikalnost(2)
pridevnik	P	vrsta(3), stopnja(3), spol(3), število(3), sklon(6), določnost(2)
prislov	R	stopnja(3), deležje(2)
zaimek	Z	vrsta(9), oseba(3), spol(3), število(3), sklon(6), število svojine(3), spol svojine(3), oblika(2)
števnik	K	zapis(3), vrsta(4), spol(3), število(3), sklon(6), določnost(2)
predlog	D	sklon(6)
veznik	V	vrsta(2)
členek	L	brez lastnosti
medmet	M	brez lastnosti
okrajšava	O	brez lastnosti
neuvrščeno	N	vrsta(3)

Tabela 2.2: Besedne vrste ter njihove lastnosti.

2.1 Vsebina gesla

Geslo v leksikonu besednih oblik navadno vsebuje osnovno obliko ter vse pripadajoče oblike z lastnostmi, lahko pa tudi povezavo na druga gesla ali informacijo o pomenu gesla [4].

V nadaljevanju je prikazan del gesla za samostalnik *miza* (ne vsebuje vseh pripadajočih besednih oblik) iz leksikona:

```
<?xml version="1.0" encoding="utf-8"?>
<LexicalEntry id="LE_f04e4cc44cf7497b038fc5fc1fd3eb30">
  <feat att="ključ" val="S_miza"/>
```

```
<feat att="besedna_vrsta" val="samostalnik"/>
<feat att="SP2001" val="samostalnik"/>
<feat att="vrsta" val="občno_ime"/>
<feat att="spol" val="ženski"/>
<Lemma>
  <feat att="zapis_oblike" val="miza"/>
</Lemma>
<WordForm>
  <feat att="msd" val="Sozei"/>
  <feat att="število" val="ednina"/>
  <feat att="sklon" val="imenovalnik"/>
  <FormRepresentation>
    <feat att="zapis_oblike" val="miza"/>
    <feat att="pogostnost" val="15509"/>
  </FormRepresentation>
</WordForm>
<WordForm>
  <feat att="msd" val="Sozer"/>
  <feat att="število" val="ednina"/>
  <feat att="sklon" val="rodilnik"/>
  <FormRepresentation>
    <feat att="zapis_oblike" val="mize"/>
    <feat att="pogostnost" val="21037"/>
  </FormRepresentation>
</WordForm>
<!-- ... -->
<WordForm>
  <feat att="msd" val="Sozeo"/>
  <feat att="število" val="ednina"/>
  <feat att="sklon" val="orodnik"/>
  <FormRepresentation>
```

```
      <feat att="zapis_oblike" val="mizo"/>
      <feat att="pogostnost" val="6194"/>
    </FormRepresentation>
  </WordForm>
  <WordForm>
    <feat att="msd" val="Sozdi"/>
    <feat att="število" val="dvojina"/>
    <feat att="sklon" val="imenovalnik"/>
    <FormRepresentation>
      <feat att="zapis_oblike" val="mizi"/>
      <feat att="pogostnost" val="101"/>
    </FormRepresentation>
  </WordForm>
  <WordForm>
    <feat att="msd" val="Sozmo"/>
    <feat att="število" val="množina"/>
    <feat att="sklon" val="orodnik"/>
    <FormRepresentation>
      <feat att="zapis_oblike" val="mizami"/>
      <feat att="pogostnost" val="2307"/>
    </FormRepresentation>
  </WordForm>
</LexicalEntry>
```

Krovni element v geslu je LexicalEntry, ki mu je dodana enolična identifikacijska oznaka. Nato sledijo lastnosti ter vrednosti, ki so skupne vsem besednim oblikam. V elementu Lemma se nahaja zapis osnovne oblike - leme.

Besedne oblike z enakimi oblikoskladenjskimi lastnostmi se nahajajo v istem WordForm elementu. Znotraj elementa WordForm se navadno nahaja en element FormRepresentation, v katerem je zapis besedne oblike, lahko pa se nahaja tudi več elementov FormRepresentation, če obstaja več variant zapisov (npr. *Pierru* - *Pierreu*).

Poglavje 3

Uporabljene tehnologije in orodja

Pri snovanju orodja za avtomatsko širitev leksikona besednih oblik Sloleks smo uporabili metode strojnega učenja. V tem poglavju bomo pogledali kaj strojno učenje je in dve njegovi najbolj razširjeni obliki - **nadzorovano** ter **nenadzorovano** učenje. Na kratko bomo opisali tudi programsko okolje R in ostala orodja, ki smo jih uporabili pri izdelavi diplomske naloge.

3.1 Strojno učenje

SAS, eno izmed vodilnih podjetij za analitiko in poslovno inteligenco, strojno učenje definira kot metodo za analizo podatkov, ki avtomatizira izdelavo analitičnega modela. Z uporabo algoritmov, ki se iterativno učijo iz podatkov, lahko računalniki s pomočjo strojnega učenja najdejo skrite vzorce, ne da bi bili eksplicitno programirani [11].

Strojno učenje je danes uporabljeno na mnogih področjih, npr. filtriranje nezaželene pošte, samovozeča vozila, razpoznava govora in slik, odkrivanje zavarovalniških ali bančnih goljufij, avtomatski nakup ali prodaja delnic, napovedovanje verjetnosti srčnega infarkta.

Dve najbolj razširjeni obliki strojnega učenja sta **nadzorovano učenje**

(ang. *supervised learning*) in **nenadzorovano učenje** (ang. *unsupervised learning*).

3.1.1 Nadzorovano učenje

Cilj nadzorovanega učenja je naučiti se preslikati vhod x v izhod y, pri čemer imamo na voljo učno množico parov vhod-izhod $T = \{(x_i, y_i)_{i=1}^N\}$. T imenujemo **učna množica**, N pa število učnih primerov.

Vektor x_i navadno predstavlja D dimenzionalni vektor števil, ki predstavljajo lastnosti oziroma **značilke** učnega primera (npr. število sob, velikost, leto gradnje). Kadar y_i pripada končni množici vrednosti, $y_i \in \{1, ..., C\}$ (npr. benigen ali maligen tumor), problemu pravimo **razvrščanje** (ang. *classification*), kadar pa y_i pripada množici realnih števil (npr. cena nepremičnine), pa problemu pravimo **regresija** (ang. *regression*) [12, str. 2].

Nadzorovano učenje navadno uporabimo v primerih, ko podatke iz preteklosti uporabimo za napovedovanje v prihodnosti. V okviru diplomske naloge bomo nadzorovano učenje uporabili za napoved skupine, ki ji neznani samostalnik pripada.

3.1.2 Nenadzorovano učenje

Pri nenadzorovanem učenju nam kot vhod služi množica primerov $T = \{(x_i)_{i=1}^N\}$. Cilj učenja je v podatkih najti "zanimive vzorce" [12, str. 2] oziroma jih razvrstiti v gruče, glede na medsebojno podobnost.

Nenadzorovano učenje uporabimo lahko npr. za segmentiranje kupcev ali grupiranje podobnih časopisnih novic [11]. V diplomski nalogi bomo nenadzorovano učenje uporabili za tvorjenje skupin samostalnikov s podobnimi pregibnimi vzorci.

3.2 Orange

Kot glavno orodje pri izdelavi modelov za strojno učenje smo najprej izbrali sistem Orange. Orange je odprtokodno orodje za strojno učenje in podatkovno rudarjenje, ki ga lahko uporabimo kot python knjižico ali kot aplikacijo za vizualno programiranje [13].

Orange vsebuje različne komponente za predprocesiranje podatkov, razvrščanje v skupine, klasifikacijo, regresijo itn.

Med uporabo orodja Orange smo opazili nekaj pomanjkljivosti:

- Dokumentacija programskih knjižic je nekoliko pomanjkljiva - nekateri razredi niso opisani oz. niso opisani dovolj natančno.
- Kljub temu, da se na spletu najde že kar nekaj primerov uporabe Oranga, je v primerjavi z orodji scikit-learn ali R uporaba še vedno precej manjša. Pogosto na spletu zato ne uspemo najti informacij glede uporabe, ki nas zanimajo.
- Nekateri algoritmi (DBSCAN, Gower distance), ki smo jih želeli uporabiti v okviru diplomske naloge, v orodju Orange niso bili podprti.

Kljub temu, da bi zgornje pomanjkljivosti lahko odpravili z analizo programske kode oziroma lastno izdelavo komponent, smo se odločili, da bomo raje uporabili programsko okolje R.

3.3 R

R je programsko okolje in jezik za statistično računanje ter grafiko. Vsebuje različne statistične (linearno in nelinearno modeliranje, klasični statistični testi, klasifikacijo, razvrščanje v skupine) in grafične metode. Omogoča enostavno kreiranje grafov, vključno z matematičnimi simboli in formulami [14].

R je razširljiv s pomočjo paketov. Osnovni paketi so vključeni v distribucijo, več kot 7800 paketov pa je dosegljivih na omrežju CRAN (ang. *Comprehensive R Archive Network*) in pokrivajo velik del moderne statistike.

Programski jezik R je obsežno dokumentiran, dokumentacija je napisana v lastnem formatu Rd, ki je podoben formatu LaTeX[15]. R je prosto dostopen pod GNU licenco in je na voljo za Unix, Windows in MacOS platforme.

3.4 Druga orodja

Pri izdelavi diplomske naloge smo uporabili še nekatera druga orodja.

V programskem jeziku C# smo razvili program za obdelavo XML datotek. C# smo izbrali, ker ima zelo dobro podporo za branje XML datotek. Za branje XML datotek smo uporabili razred `XmlReader`, ki omogoča hitro, enosmerno branje. Program smo razvili v razvojnem okolju Microsoft Visual Studio 2013.

Za shranjevanje verzij izvorne kode smo uporabili Git, izvorno kodo pa smo shranjevali na spletnem servisu GitHub.

Poglavje 4

Razširitev leksikona

Poglavje o razširitvi leksikona besednih oblik Sloleks smo razdelili na štiri dele. V prvem bomo obstoječe samostalnike iz Sloleksa z gručenjem z medoidi razvrstili v skupine, ki imajo podobne pregibne vzorce ter oblikoskladenjske lastnosti. Nato bomo z rezultati gručenja zgradili naivni Bayesov klasifikator za napovedovanje skupin novim neznanim samostalnikom. V tretjem delu bomo z napovedanimi skupinami ustrezno dopolnili manjkajoče podatke v korpusu. Poglavje bomo zaključili z analizo rezultatov.

4.1 Razvrščanje v skupine

Postopek širitve leksikona besednih oblik Sloleks smo začeli z razvrstitvijo obstoječih samostalnikov v skupine, ki imajo enake oziroma podobne pregibne vzorce. Pred razvrščanjem v skupine smo morali podatke ustrezno pripraviti za uporabo v programskem okolju R.

4.1.1 Priprava podatkov

Za pripravo podatkov smo napisali konzolno aplikacijo v programskem jeziku C#. Aplikacija pretvori Sloleks iz XML oblike v CSV obliko ter ustrezno kreira značilke, ki jih bomo uporabili za strojno učenje.

Program sprejme kot vhodni parameter pot do XML datoteke s Sloleksom, kot izhod pa kreira CSV datoteko. Program uporablja za branje XML datoteke razred `XmlReader`. Razred omogoča samo enosmerno sekvenčno branje, ki pa je zelo hitro in porabi malo pomnilnika. Z razredom `XmlReader` preberemo celotno geslo, nato pa ga z razredi iz imenskega prostora `System.Xml.Linq` razčlenimo na posamezne dele - osnovno obliko ter pripadajoče besedne oblike.

Vsak zapis oziroma vrstica v izhodni CSV datoteki predstavlja eno osnovno obliko ter vse njene pripadajoče besedne oblike. Če ima samostalnik v nekem sklonu ter številu več variant zapisa (npr. *oproda*, ki ima v rodilniku ednine varianti zapisa *oproda* ter *oprode)*, potem v izhodni datoteki vrstice z osnovno obliko ponovimo tako, da zajamemo vse variante zapisa.

V tabeli 4.1 so opisane značilke, ki jih pripravi program za besedno vrsto samostalnik. Vsak zapis oblike je zapisan s 5 značilkami. Primer značilk za zapis osnovne oblike:

- *lema_zapis_oblike*: zapis oblike
- *lema_kon1*: zadnja črka v zapisu oblike
- *lema_kon2*: zadnji dve črki v zapisu oblike
- *lema_kon3*: zadnji tri črki v zapisu oblike
- *lema_kon4*: zadnje štiri črke v zapisu oblike

Pripravljene podatke nato preberemo v programskem okolju R, kot prikazuje spodnji izsek kode.

- (vrstica 2) Datoteko naložimo z ukazom `read.table`.
- (vrstica 3) V kolikor nam na računalniku primanjkuje pomnilnika, lahko iz nabora vzamemo samo del podatkov. Celotna vhodna datoteka vsebuje približno 55.800 vrstic. Na osebnem računalniku z 12GB pomnilnika smo lahko uspešno razvrstili približno 10.000 primerov.

- (vrstica 4) Značilke z zapisi oblik izločimo iz vhodnih podatkov, saj jih za razvrščanje ne potrebujemo.

```
1 filePath <- c("d:/Data/samostalnik.tab")
2 data <- read.table(filePath, header = TRUE, fileEncoding = "UTF
    -8-BOM", na = "", sep = "\t")
3 data <- data[sample(nrow(data), 10000), ]
4 clData <- data[,-grep(".*zapis_oblike$", colnames(data))]
```

ime značilke	**opis značilke**
lema_{X}	zapis osnovne oblike
ednina_imenovalnik_{X}	zapis v imenovalniku ednine
ednina_rodilnik_{X}	zapis v rodilniku ednine
ednina_dajalnik_{X}	zapis v dajalniku ednine
ednina_tozilnik_{X}	zapis v tožilniku ednine
ednina_mestnik_{X}	zapis v mestniku ednine
ednina_orodnik_{X}	zapis v orodniku ednine
dvojina_imenovalnik_{X}	zapis v imenovalniku dvojine
dvojina_rodilnik_{X}	zapis v rodilniku dvojine
dvojina_dajalnik_{X}	zapis v dajalniku dvojine
dvojina_tozilnik_{X}	zapis v tožilniku dvojine
dvojina_mestnik_{X}	zapis v mestniku dvojine
dvojina_orodnik_{X}	zapis v orodniku dvojine
mnozina_imenovalnik_{X}	zapis v imenovalniku množine
mnozina_rodilnik_{X}	zapis v rodilniku množine
mnozina_dajalnik_{X}	zapis v dajalniku množine
mnozina_tozilnik_{X}	zapis v tožilniku množine
mnozina_mestnik_{X}	zapis v mestniku množine
mnozina_orodnik_{X}	zapis v orodniku množine
samo_edninska_oblika	ali obstaja samo edninska oblika
samo_mnozinska_oblika	ali obstaja samo množinska oblika

Tabela 4.1: Značilke samostalnikov.

4.1.2 Gručenje

Gručenje (ang. *clustering*) sodi med nenadzorovano učenje. Sestavljeno je iz treh povezanih korakov [17]:

1. izračuna razdalje (podobnosti) med dvema primeroma,
2. izbire algoritma za razvrščanje in
3. izbire števila skupin.

Izračun razdalje

Pred izbiro algoritma za razvrščanje moramo ustrezno izbrati funkcijo za izračun razdalje med primeri. V algoritmih za gručenje (npr. k voditeljev, ang. *K-means*) se pogosto uporablja evklidska razdalja, ki pa je primerna zgolj za številske (ang. *numeric*) značilke. Ker v podatkih o samostalnikih uporabljamo tudi kategorične (zapisi končnic) ter logične (podatki o edninski ter množinski obliki) značilke, evklidska razdalja ni primerna.

Ena izmed razdalj, ki upošteva tudi kategorične značilke je **Gowerjeva** razdalja. Razdalja med dvema primeroma je definirana kot utežena sredina prispevkov vseh značilk [18].

$$d_{ij} = d(i,j) = \frac{\sum_{k=1}^{p} \omega_k \delta_{ij}^{(k)} d_{ij}^{(k)}}{\sum_{k=1}^{p} \omega_k \delta_{ij}^{(k)}}$$

d_{ij} je utežena sredina $d_{ij}^{(k)}$ z utežmi $\omega_k \delta_{ij}^{(k)}$, kjer je $\omega_k =$ `weights`[k] (`weights` je vektor uteži, podan s klicom funkcije `daisy`), $\delta_{ij}^{(k)}$ je 0 ali 1, in $d_{ij}^{(k)}$, prispevek `k`-te značilke k skupni razdalji, je razdalja med `x`[i,k] in `x`[j,k] (razdalja `k`-te značilke med primeroma `i` in `j`).

$\delta_{ij}^{(k)}$ je 0, kadar je vsaj ena od značilk `x`[i,k] ali `x`[j,k] brez vrednosti, ali pa je značilka definirana kot asimetrični logični tip in sta obe vrednosti enaki 0. V vseh ostalih primerih je $\delta_{ij}^{(k)}$ enak 1.

Prispevek $d_{ij}^{(k)}$ kategoričnega ali logičnega tipa k skupni razdalji je 0, če sta obe vrednosti `x`[i,k] ter `x`[j,k] enaki, sicer pa je vrednost 1. Prispevek

značilke drugega tipa je enak absolutni razliki vrednosti, deljeni s celotnim obsegom vrednosti značilke.

Ker je vrednost posameznega prispevka $d_{ij}^{(k)}$ med [0,1], je tudi skupna razdalja d_{ij} med [0,1].

V programskem okolju R Gowerjevo razdaljo izračunamo s funkcijo `daisy`, ki se nahaja v paketu `cluster`. Opcijski parameter `weights` je vektor dolžine, ki je enaka številu značilk v učnih primerih. Z njim določimo uteži posameznim značilkam. Če parametra ne določimo, potem ima vsaka značilka privzeto utež, ki je enaka 1. Funkcija `daisy` vrne matriko razdalj (ang. *dissimilarity matrix*), kot prikazuje spodnja koda:

1. (vrstica 1) Ustvarimo vektor dolžine, ki je enaka številu značilk v učni množici (clData).

2. (vrstica 2) Vsem značilkam nastavimo privzeto utež, ki je enaka 1.

3. (vrstica 3-9) Ustrezno utežimo značilke za samostalniško besedno vrsto.

4. (vrstica 11) Pokličemo funkcijo za izračun matrike razdalj, pri čemer uporabimo učno množico, Gowerjevo razdaljo ter definirane uteži značilk.

```
1  weights <- integer(ncol(clData))
2  weights[] <- 1
3  weights[grep("samo_edninska_oblika", colnames(clData))] <- 80
4  weights[grep("samo_mnozinska_oblika", colnames(clData))] <- 80
5  weights[grep("ednina_imenovalnik_kon1", colnames(clData))] <- 60
6  weights[grep("ednina_imenovalnik_kon2", colnames(clData))] <- 50
7  weights[grep("ednina_rodilnik_kon1", colnames(clData))] <- 60
8  weights[grep("ednina_rodilnik_kon2", colnames(clData))] <- 50
9  weights[grep("ednina_rodilnik_kon3", colnames(clData))] <- 40
10
11 diss <- daisy(clData, metric = "gower", weights = weights)
```

Izbira algoritma za gručenje

Ko smo izračunali matriko razdalj, lahko izberemo algoritem za gručenje. R vsebuje kar nekaj algoritmov gručenja, ki jim lahko podamo izračunano matriko razdalj. Odločili smo se, da bomo uporabili algoritem **PAM** (ang. *Partitioning Around Medoids*). PAM smo uporabili, ker smo lahko z izračunom povprečne širine silhuete našli optimalno število skupin, ki ga nismo mogli določiti ročno ter zato, ker PAM ob gručenju vrne tudi medoide, ki smo jih potrebovali kot vzorce za dopolnitev manjkajočih besednih oblik.

V okolju R je PAM algoritem implementiran v funkciji `pam`, v paketu `cluster` [18]. Po načinu delovanja je podoben gručenju z voditelji (paket `stats`), z nekaj dodatnimi lastnostmi:

- omogoča, da mu podamo matriko razdalj, ki smo jo izračunali v prejšnjem koraku,
- je bolj robusten, saj minimizira vsoto razdalj, namesto vsote kvadratnih evklidskih razdalj,
- omogoča izris grafa silhuete,
- omogoča izbor števila skupin z uporabo povprečne širine silhuete.

Delovanje PAM algoritma je mogoče opisati z naslednjimi iterativnimi koraki [17, 18]:

1. Izberemo k tipičnih primerov imenovih tudi **medoidi**. Medoide lahko podamo s parametrom `medoids` ali pa jih algoritem izbere naključno.
2. Vsak primer dodelimo tipičnemu primeru, ki mu je najbližje. Algoritem pri tem uporabi podano matriko razdalj (ali pa razdaljo izračuna iz podatkov, če matrike razdalj nismo podali).
3. V vsaki skupini poiščemo primer, ki minimizira vsoto razdalj v skupini, če bi ta primer postal medoid. Če nam uspe najti tak primer, potem ta primer postane novi medoid.

4. Če smo v koraku 3 uspeli najti vsaj en nov medoid, potem ponovimo korak 2. V nasprotnem primeru je algoritem končal gručenje.

Izbira števila skupin

Za začetek gručenja moramo ob klicu funkcije pam s parametrom `k` določiti število skupin. Pri določanju števila si lahko pomagamo z različnimi metodami. V diplomski nalogi bomo uporabili metodo povprečne širine silhuete.

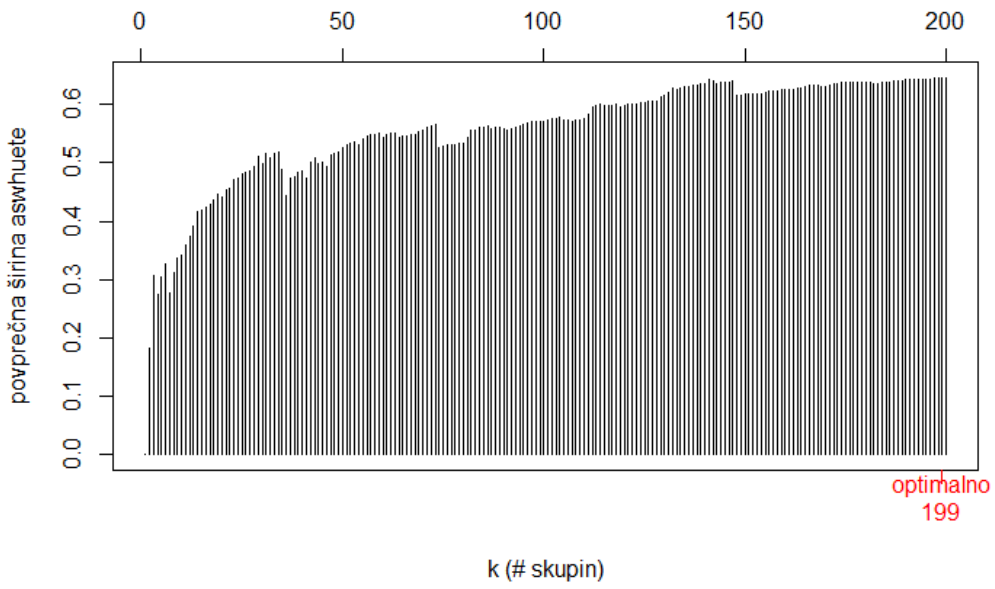
Povprečna širina silhuete nam pove povprečno razdaljo primerov do najbližje skupine, kateri primer ne pripada. Vrednost povprečne širine je med [-1, 1], kjer je večja vrednost boljša. Povprečno širino silhuete lahko preberemo iz rezultata gručenja. Če vrednost maksimiziramo z različnimi števili skupin, potem lahko najdemo optimalno število. Spodnja programska koda prikazuje iskanje optimalnega števila skupin v programskem okolju R:

```
#' Funckija vrne optimalno število skupin glede na povprečno širino silhuete.
#' @param diss Matrika razdalj izračunana s funkcijo daisy.
#' @param minClusters Spodnja meja števila skupin.
#' @param maxClusters Zgornja meja števila skupin.
#' @return Optimalno število skupin med \code{minClusters} in \code{maxClusters}.
findOptimalNumberOfClusters <- function(diss, minClusters = 2, maxClusters = 200) {
  asw <- numeric(maxClusters)
  for (k in minClusters : maxClusters) {
    asw[[k]] <- pam(diss, k, diss = TRUE) $ silinfo $ avg.width
  }

  k.best <- which.max(asw)
  cat("Optimalno število skupin:", k.best, "\n")
  plot(1 : maxClusters, asw, type = "h", xlab = "k (# skupin)",
       ylab = "povprečna širina silhuete", axes = FALSE)
  axis(3)
  axis(2)
  box()
```

```
18    axis(1, k.best, paste("optimalno", k.best, sep = "\n"), col =
          "red", col.axis = "red")
19    return (k.best)
20  }
```

Graf na sliki 4.1 prikazuje povprečno širino silhuete glede na število skupin. V primeru razvrščanja samostalinkov smo preizkusili povprečno širino silhute za število skupin med 2 in 200 ter ugotovili, da je optimalno število skupin 199.



Slika 4.1: Optimalno število skupin.

Ko smo ugotovili optimalno število skupin, lahko pokličemo funkcijo pam z matriko razdalj iz prvega koraka ter optimalnim številom skupin.

```
1  noClusters <- findOptimalNumberOfClusters(diss, 1, 200)
2  clusters <- pam(diss, noClusters, diss = TRUE)
```

Funkcija pam vrne objekt tipa pam, ki predstavlja rezultat razvrščanja. Objekt je seznam, ki v komponenti clustering vsebuje vektor dolžine n, število primerov, ki za vsak primer vsebuje številko skupine, kateri primer pripada. Vektor, ki vsebuje skupine, združimo s podatki o samostalnikih ter rezultate shranimo v datoteko, saj jih bomo potrebovali pri napovedovanju skupin. V datoteko shranimo tudi medoide, ki jih bomo potrebovali pri generiranju oblikoslovnih paradigem.

Spodnji izsek kode prikazuje shranjevanje rezultatov gručenja.

```
# Združimo samostalnike iz Sloleksa, skupaj z vektorjem, ki
    vsebuje skupine.
output <- data.frame(cluster = clusters$clustering, data)

# Združene podatke shranimo v datoteko, kjer so podatki ločeni s
    tabulatorji.
clustersPath <- c("d:/Data/samostalnik199_pam.tab")
write.table(output, file = clustersPath, quote = FALSE, sep = "\
    t", row.names = FALSE, fileEncoding = "UTF-8", na = "")

# V datoteko shranimo vektor z medoidi. Vektor je dolžine, enake
    številu skupin.
# Vsak element vektorja vsebuje indeks v tabeli data, ki kaže na
    medoid skupine.
medoidsPath <- c("d:/Data/samostalnik199_medoids.tab")
write(clusters$id.med, file = medoidsPath)
```

Rezultati razvrščanja v skupine

V tabeli 4.2 podajamo nekaj primerov skupin ter pripadajočih samostalnikov (osnovnih oblik), ki smo jih dobili z opisanim postopkom.

Skupina 1 - samostalniki z edninskim delom paradigme:				
absint	AGRFT	Alples	Arnes	bambus
Bayer	Brdo	burja	CTK	Delhi
FLRJ	GESŠ	hipnoza	internet	jezikoslovje
kWh	NUK	sirtaki	Škuc	Zemlja
Skupina 9:				
bučnica	glivica	hrčica	iglica	kitica
masnica	osemletnica	polovica	pravljica	premica
rutica	šalica	tlačiteljica	tortica	zajčnica
Skupina 11:				
bljuvanje	copotanje	čivkanje	drsanje	hujskanje
jadranje	naslanjanje	obdolževanje	odnašanje	piskanje
renčanje	servisiranje	trajanje	varovanje	zdesetkanje
Skupina 17 - samostalniki z množinskim delom paradigme:				
Alpe	Brioni	Dražgoše	Hoče	Jesenice
Pale	pleča	pljuča	Radeče	saldokonti
Selca	Strelci	Trate	Ture	Žage
Skupina 37:				
Aleksič	bič	Deržič	Djurdjič	Kidrič
Mokrovič	Partljič	pobič	Robič	Spetič
Tomažič	Veselinovič	Uršič	zatič	Žunič
Skupina 53:				
analitik	Blatnik	brusnik	dimnik	epileptik
Hladnik	ledenik	nadžupnik	obračalnik	opomnik
pesnik	Repnik	sedemletnik	štrik	župnik
Skupina 182:				
Avbelj	Gaj	Knafelj	Mertelj	Peternelj
Pregelj	Retelj	Ručigaj	Rugelj	Snoj
Stroj	Šantej	Šinigoj	Škrlj	Tiselj

Tabela 4.2: Primeri predstavnikov v skupinah.

4.2 Napovedovanje oblikoslovne skupine

V drugem delu diplomske naloge smo izdelali model za napovedovanje skupin, pri čemer smo uporabili rezultate (razvrščene skupine) iz prvega dela naloge. Postopku napovedovanja skupin pravimo tudi klasifikacija in spada na področje nadzorovanega strojnega učenja.

Klasifikator je funkcija f, ki preslika vhodne vektorje značilk $x \in X$ v izhodne oznake razredov $y \in \{1, ..., C\}$, kjer je X prostor značilk. Tipično velja, da $X = \mathbb{R}^D$ ali $X = \{0, 1\}^D$, kar z drugimi besedami pomeni, da je vektor značilk vektor D realnih števil ali D binarnih števil. Naš cilj je iz označene učne množice N parov $(x_n, y_n), n = 1 : N$ naučiti se funkcijo f [19].

V diplomski nalogi smo za napovedovanje skupin uporabili **naivni Bayesov klasifikator** (ang. *Naive Bayesian classifier*).

4.2.1 Naivni Bayesov klasifikator

Naivni Bayesov klasifikator deluje na osnovi Bayesovega pravila:

$$p(y|x) = \frac{p(y) \times p(x|y)}{p(x)}$$

Klasifikator izračuna pogojne verjetnosti za napoved razreda glede na podane značilke. Klasifikator se imenuje naivni, saj predvideva neodvisnost med značilkami. V tem primeru velja [12]:

$$p(x|y = c) = \prod_{i=1}^{D} p(x_i|y = c)$$

Kljub temu, da so značilke pogosto vsaj delno odvisne, klasifikator deluje presenetljivo dobro. Naivni Bayesov klasifikator je preprost za uporabo in je zato tudi pogosto uporabljen.

Implementacija naivnega Bayesovega klasifikatorja, ki smo jo uporabili v diplomski nalogi, se nahaja v funkciji `naiveBayes` v paketu `e1071`.

4.2.2 Testiranje modela

Za oceno točnosti modela smo uporabili **k-kratno prečno preverjanje** (ang. *k-fold cross-validation*). V k-kratnem prečnem preverjanju podatke za testiranje modela razdelimo v k enako velikih skupin, kjer eno skupino uporabimo za testiranje modela, preostale k - *1* skupine pa so uporabljene za učenje modela. Prečno preverjanje je ponovljeno k krat, kjer je vsaka izmed k skupin točno enkrat uporabljena kot testna skupina. Točnost modela ocenimo s povprečno točnostjo posameznih prečnih preverjanj [20].

Vrednost k je lahko poljubna, pogosto uporabljeno je desetkratno prečno preverjanje, ki smo ga uporabili tudi pri diplomski nalogi.

Za izdelavo skupin smo uporabili funkcijo createFolds iz paketa caret.

```
library(caret)
library(e1071)
set.seed(2409)

# Branje skupin, ki smo jih pripravili v prvem koraku
clusters <- loadClustersData("d:/Data/samostalnik199_pam.tab")

# Desetkratno precno preverjanje
folds <- createFolds(clusters$cluster, k = 10)
acc <- numeric(10)
for (i in 1 : 10) {
  train <- modelData[-folds[[i]], ]
  test <- modelData[folds[[i]], ]
  test_x <- test[, -which(names(test) == "cluster")]
  test_y <- test[, "cluster"]
  model <- naiveBayes(cluster ~ ., data = train)
  predictions <- predict(model, newdata = test_x)
  confMatrix <- table(test_y, predictions)
  acc[i] <- sum(diag(confMatrix))/sum(confMatrix)
}
summary(acc)
```

Iz desetkratnega prečnega preverjanja smo ugotovili, da je povprečna klasifikacijska točnost 0,8899 oziroma **88,99%**, s standardnim odklonom **0,0041**.

4.3 Kreiranje paradigem

V okviru razširitve leksikona besednih oblik Sloleks, smo izdelali orodje za kreiranje celotnih paradigem na podlagi osnovnih oblik ter pripadajočih besednih oblik, ki se nahajajo v pisnem korpusu ccGigafida [1].

Postopek kreiranja paradigem smo razdelili na tri korake, ki so v nadaljevanju opisani podrobneje:

1. priprava podatkov,
2. napovedovanje skupin,
3. kreiranje paradigem.

4.3.1 Priprava podatkov

Priprava podatkov poteka znotraj iste aplikacije, ki smo jo napisali za pretvorbo Sloleksa iz XML v CSV obliko. V ta namen smo napisali nov razred, ki iz izvornih XML datotek korpusa ccGigafida izlušči osnovne oblike ter pripadajoče besedne oblike samostalnikov, ki ustrezajo kriterijem, ter jih shrani v CSV datoteko. Da je samostalnik zajet, mora ustrezati naslednjim kriterijem:

- osnovna oblika se mora v korpusu pojaviti najmanj dvakrat,
- dolžina osnovne oblike ne sme biti daljša kot 25 znakov,
- osnovna oblika lahko vsebuje samo črke slovenske abecede,
- osnovna oblika mora biti zapisana z malimi črkami.

[1] Zbirka je dostopna na spletni strani http://www.slovenscina.eu/korpusi/proste-zbirke

Struktura CSV datoteke, ki jo aplikacija generira, ustreza strukturi datoteke, ki nastane ob pretvorbi Sloleksa v CSV. Značilke so opisane v tabeli 4.1.

4.3.2 Napovedovanje skupin

Za učenje naivnega Bayesovega klasifikatorja smo uporabili iste parametre, ki smo jih uporabili pri desetkratnem prečnem preverjanju, vendar smo tokrat klasifikator učili na celotni učni množici.

Ker lahko podatki vsebujejo tudi končnice zapisov oblik, ki jih ni v učni množici, klasifikator naučimo tudi z novimi podatki. To pomeni, da moramo diskretnim značilkam pridružiti možne nove vrednosti atributov, ki se nahajajo v korpusu.

```
# Nastavimo poti do datotek
sloleksPath <- c("d:/Data/samostalnik.tab")
clustersPath <- c("d:/Data/samostalnik199_pam.tab")
medoidsPath <- c("d:/Data/samostalnik199_medoids.tab")
corpusPath <- c("d:/Data/samostalnik_gfcc.tab")
reportPath <- c("d:/Data/samostalnik199_report.tab")

# Preberemo podatke iz razvrščanja
clustersData <- read.table(clustersPath, header = TRUE,
    fileEncoding = "UTF-8", na = "", sep = "\t")

# Spremenimo gruče v factor
clustersData$cluster <- as.factor(clustersData$cluster)

# Odstranimo oblike, ki jih ne potrebujemo za učenje modela
clustersTrain <- clustersData[, -grep(".*zapis_oblike$",
    colnames(clustersData))]
clustersTrain <- clustersTrain[, -grep(".*samo_edninska_oblika$"
    , colnames(clustersTrain))]
clustersTrain <- clustersTrain[, -grep(".*samo_mnozinska_oblika$
    ", colnames(clustersTrain))]

# Preberemo podatke iz korpusa
```

```
corpusData <- read.table(corpusPath, header = TRUE, fileEncoding = "UTF-8", na = "", sep = "\t")

# Odstranimo oblike, ki jih ne potrebujemo za napovedovanje skupine
corpusPred <- corpusData[, -grep(".*zapis_oblike$", colnames(corpusData))]
corpusPred <- corpusPred[, -grep(".*samo_edninska_oblika$", colnames(corpusPred))]
corpusPred <- corpusPred[, -grep(".*samo_mnozinska_oblika$", colnames(corpusPred))]

# Združimo kategorije učnih podatkov in korpusa
for (n in names(corpusPred)) {
  mergedLevels <- union(levels(clustersTrain[[n]]), levels(corpusPred[[n]]))
  corpusPred[[n]] <- factor(corpusPred[[n]], levels = mergedLevels)
  clustersTrain[[n]] <- factor(clustersTrain[[n]], levels = mergedLevels)
}

# Treniranje učnega modela
model <- naiveBayes(cluster ~ ., data = clustersTrain, laplace = 0.01)

# Napovedovanje skupin za podatke iz korpusa
predictions <- predict(model, newdata = corpusPred)
```

4.3.3 Kreiranje paradigem

Ko smo delnim paradigmam iz korpusa ccGigafida napovedali skupine, ki jim pripadajo, lahko uporabimo funkcijo za dopolnitev paradigem. Funkcijo lahko opišemo z naslednjo psevdo kodo:

- za vsako paradigmo iz korpusa ccGigafida:
  - pridobi razred, ki mu paradigma pripada,
  - pridobi medoid (tipični primer), ki pripada razredu,
  - pridobi osnovni zapis (lemo) medoida,
  - pridobi osnovni zapis (lemo) paradigme,
  - za vsak zapis oblike paradigme:
    - preveri, ali je zapis oblike prazen; če ni, nadaljuj z naslednjim zapisom oblike,
    - preveri, ali je zapis oblike v medoidu prazen; če je, nadaljuj z naslednjim zapisom oblike,
    - pridobi zapis oblike v medoidu,
    - generiraj zapis oblike glede na osnovni zapis medoida, osnovni zapis paradigme ter zapis oblike v medoidu,
    - shrani generiran zapis oblike.

Denimo, da želimo v paradigmi *banana* dopolniti dajalnik dvojine. Pri tem uporabimo tipični primer (medoid) *revija*, ki ima v dajalniku dvojine obliko *revijama.* Funkcija za generiranje zapisa oblike je definirana z naslednjo psevdo kodo:

1. izračunaj dolžino skupnega dela osnovne oblike ter zapisa oblike medoida. Primer: *revija, revijama, dolžina skupnega dela je 5 znakov (revij).*

2. izračunaj dolžino končnice v zapisu oblike medoida. Primer: *revijama - revij, dolžina končnice je 3 znake (ama).*

3. izračunaj dolžino končnice v osnovni obliki medoida. Primer: *revija - revij, dolžina končnice je 1 znak (a).*

4. iz osnovne oblike paradigme odstrani končnico dolžine, izračunane v 3. koraku. Primer: *banana - 1 (a) = banan*

5. osnovni obliki paradigme, dobljeni v 4. koraku, dodaj končnico, dobljeno v 2. koraku. Primer: *banan + ama = bananama*. To je tudi rezultat klica funkcije za generiranje besedne oblike.

Programska koda obeh funkcij se nahaja v GitHub repozitoriju.

4.3.4 Analiza rezultatov

V okviru diplomske naloge smo iz korpusa ccGigafida zajeli 70.228 samostalniških primerov, ki smo jim napovedali skupine ter dopolnili manjkajoče oblike. Pri tem smo ugotovili, da se med 70.228 primeri 32.227 primerov že nahaja v leksikonu besednih oblik Sloleks, od tega je pravilnih primerov 28.260, oziroma **87.69%**.

Pravilno dopolnjene paradigme

V tabeli 4.3 je navedenih nekaj primerov dopolnjenih paradigem, kjer so besedne oblike, ki so napisane z navadno pisavo, pridobljene iz korpusa ccGigafida, besedne oblike, ki so odebeljene, pa so bile kreirane avtomatsko. Vsi primeri so bili dopolnjeni pravilno, kar pomeni, da so vse oblike enake kot v Sloleksu.

Napačno dopolnjene paradigme

Analizirali smo tudi nekaj primerov, kjer dopolnjene paradigme niso identične paradigmam v Sloleksu in jih kategorizirali v naslednje skupine:

1. V prvi skupini so paradigme, ki jim je bila napačno napovedana skupina. S tem smo za dopolnjevanje paradigme uporabili napačni medoid, zato je bilo tvorjenje novih besednih oblik nepravilno. Nekaj primerov napačno napovedanih skupin je v tabeli 4.4. Pri dopolnjevanju so bili uporabljeni naslednji medoidi: *tablet* za *hrbet*, *pomen* za *teden* in *patologinja* za *proizvodnja*.

	ednina	dvojina	množina
imenovalnik	letovišče	**letovišči**	letovišča
rodilnik	letovišča	**letovišč**	letovišč
dajalnik	**letovišču**	**letoviščema**	**letoviščem**
tožilnik	letovišče	letovišči	letovišča
mestnik	letovišču	**letoviščih**	letoviščih
orodnik	letoviščem	letoviščema	letovišči
imenovalnik	**rezanec**	**rezanca**	rezanci
rodilnik	**rezanca**	**rezancev**	rezancev
dajalnik	**rezancu**	**rezancema**	rezancem
tožilnik	**rezanca**	**rezanca**	rezance
mestnik	**rezancu**	**rezancih**	rezancih
orodnik	rezancem	**rezancema**	rezanci
imenovalnik	**podpisovalec**	**podpisovalca**	podpisovalci
rodilnik	**podpisovalca**	**podpisovalcev**	podpisovalcev
dajalnik	**podpisovalcu**	**podpisovalcema**	**podpisovalcem**
tožilnik	**podpisovalca**	**podpisovalca**	**podpisovalce**
mestnik	**podpisovalcu**	**podpisovalcih**	**podpisovalcih**
orodnik	**podpisovalcem**	**podpisovalcema**	podpisovalci

Tabela 4.3: Primeri pravilno dopolnjenih paradigem.

2. V drugi skupini so paradigme, ki imajo v korpusu ccGigafida vneseno vsaj eno napačno obliko glede na leksikon Sloleks. Ker v postopku dopolnjevanja paradigme obstoječih oblik iz korpusa ne spreminjamo, je posledično celotna paradigma napačna. Primeri samostalnikov z napačnimi besednimi oblikami v korpusu se nahajajo v tabeli 4.5. Napačne oblike so zapisane s poševnimi črkami.

3. V tretji skupini so paradigme, ki smo jim napačno napovedali edninske ali množinske skupine, ali pa obratno: edninske ali množinske skupine nismo napovedali, pa bi jih morali. V tabeli 4.6 je prikazanih nekaj primerov iz te skupine. Samostalniku *epigonstvo* smo glede na obstoječe podatke v Sloleksu napačno napovedali edninsko skupino z medoidom *poljedelstvo*, množinskemu samostalniku *drva* smo napačno napovedali skupino z medoidom *Žemva* ter edninskemu samostalniku *biokemija* skupino z medoidom *penetracija*.

	ednina	dvojina	množina
imenovalnik	hrbet	**hrbeta**	hrbti
rodilnik	hrbta	**hrbetov**	hrbtov
dajalnik	hrbtu	hrbtoma	hrbtom
tožilnik	hrbet	**hrbeta**	hrbte
mestnik	hrbtu	**hrbetih**	hrbtih
orodnik	hrbtom	hrbtoma	hrbti
imenovalnik	teden	tedna	tedni
rodilnik	tedna	tednov	tednov
dajalnik	tednu	**tedenoma**	**tedenom**
tožilnik	teden	tedna	tedne
mestnik	tednu	tednih	tednih
orodnik	tednom	tednoma	tedni
imenovalnik	proizvodnja	**proizvodnji**	proizvodnje
rodilnik	proizvodnje	**proizvodnj**	proizvodenj
dajalnik	proizvodnji	**proizvodnjama**	**proizvodnjam**
tožilnik	proizvodnjo	proizvodnji	proizvodnje
mestnik	proizvodnji	**proizvodnjah**	proizvodnjah
orodnik	proizvodnjo	proizvodnjama	proizvodnjami

Tabela 4.4: Primeri napačno dopolnjenih paradigem iz skupine 1.

	ednina	dvojina	množina
imenovalnik	februar	februarja	**februarji**
rodilnik	*februara*	**februarjev**	**februarjev**
dajalnik	februarju	**februarjema**	februarjem
tožilnik	februar	**februarja**	februarje
mestnik	februarju	**februarjih**	**februarjih**
orodnik	februarjem	**februarjema**	**februarji**
imenovalnik	staž	**staža**	staži
rodilnik	staža	**stažev**	stažev
dajalnik	stažu	**stažema**	stažem
tožilnik	staž	**staža**	staže
mestnik	stažu	**stažih**	**stažih**
orodnik	*stažom*	**stažema**	**staži**
imenovalnik	začetek	začetka	začetki
rodilnik	začetka	**začetkov**	začetkov
dajalnik	začetku	začetkoma	začetkom
tožilnik	začetek	začetka	začetke
mestnik	*začeteku*	**začetkih**	začetkih
orodnik	*začetekom*	**začetkoma**	začetki

Tabela 4.5: Primeri napačno dopolnjenih paradigem iz skupine 2.

	ednina	dvojina	množina
imenovalnik	epigonstvo		
rodilnik	epigonstva		
dajalnik	**epigonstvu**		
tožilnik	epigonstvo		
mestnik	epigonstvu		
orodnik	epigonstvom		
imenovalnik	**drva**	**drva**	drva
rodilnik	**drva**	**drv**	drv
dajalnik	**drvi**	**drvama**	**drvam**
tožilnik	**drva**	**drva**	drva
mestnik	**drvi**	**drvah**	drveh
orodnik	**drvo**	**drvama**	drvmi
imenovalnik	biokemija	**biokemiji**	**biokemije**
rodilnik	biokemije	**biokemij**	**biokemij**
dajalnik	biokemiji	**biokemijama**	**biokemijam**
tožilnik	biokemijo	**biokemiji**	**biokemije**
mestnik	biokemiji	**biokemijah**	**biokemijah**
orodnik	biokemijo	**biokemijama**	**biokemijami**

Tabela 4.6: Primeri napačno dopolnjenih paradigem iz skupine 3.

Poglavje 5

Sklepne ugotovitve

V diplomski nalogi smo izdelali orodje za avtomatsko dopolnjevanje leksikona besednih oblik Sloleks. Problema smo se lotili s strojnim učenjem. V prvem koraku smo samostalnike iz Sloleksa razvrstili v skupine s podobnimi oblikoskladenjskimi lastnostmi, pri čemer smo uporabili gručenje z medoidi. Pri določanju števila skupin smo si pomagali z metodo povprečne širine silhuete.

V nadaljevanju smo z rezultati razvrščanja zgradili naivni Bayesov klasifikator. Z desetkratnim prečnim preverjanjem smo ugotovili, da je povprečna uspešnost klasifikatorja **88,99%**.

V zadnjem delu naloge smo iz pisnega korpusa ccGigafida zajeli samostalnike, ki smo jim s pomočjo klasifikatorja in tipičnih predstavnikov skupin generirali manjkajoče besedne oblike. Pri tem smo ugotovili, da se med 70.228 primeri, ki smo jim dopolnili paradigme, 32.227 primerov že nahaja v leksikonu besednih oblik Sloleks, od tega je pravilnih primerov 28.260, oziroma **87.69%**.

5.1 Težave in mogoče izboljšave

Pri izdelavi diplomske naloge smo naleteli na nekaj težav:

1. V koraku razvrščanja pri računanju matrike razdalj smo imeli težave s porabo pomnilnika. V okolju z 10 GB prostega pomnilnika smo lahko

izračunali matriko za približno 10.000 samostalnikov, v okolju s 100GB prostega pomnilnika pa matriko za približno 50.000 primerov. Da bi uspešno izračunali razdaljo za vse primere iz Sloleksa (približno 55.800) bi potrebovali še nekoliko več pomnilnika.

2. Pri računanju optimalnega števila skupin smo imeli težave s časovno zahtevnostjo. Računanje posamezne povprečne širine silhuete se je s številom skupin povečevalo. Za izračun vseh povprečnih širin silhuete za 10.000 primerov samostalnikov ter v razponu med 2 in 200 skupin smo potrebovali kar 72 ur. Postopek bi lahko pohitrili s paralelizacijo računanja, za kar bi lahko uporabili paket parallel.

3. Pri pripravi podatkov za dopolnjevanje paradigem iz pisnega korpusa ccGigafida smo imeli težave z zajemom samostalnikov. Izkazalo se je, da so nekatere besedne oblike zapisane z različnimi velikostmi črk, kot npr. *miza*, *Miza*, *MIZA* ali *miZA*, zato smo se odločili, da bomo zajeli samo samostalnike, kjer so osnovna oblika ter vse pripadajoče besedne oblike zapisane z malimi črkami, vendar pa s tem nismo zajeli lastnih imen ter okrajšav. Ob naslednji iteraciji bi morali spremeniti pravila za zajem samostalnikov, ki bi zajela tudi manjkajoče samostalnike. Pravilo za zapis oblike bi lahko dopolnili na naslednji način:

   - če je celotna osnovna oblika zapisana z velikimi črkami (okrajšave), potem mora biti tudi pripadajoča oblika zapisana z velikimi črkami,
   - če je celotna oblika zapisana z malimi črkami, potem mora biti tudi oblika zapisana z malimi črkami, oziroma lahko obliko samodejno pretvorimo v male črke,
   - če je prva črka osnovne oblike zapisana z velikimi črkami, ostale črke pa so zapisane z malimi črkami (lastna imena) potem mora biti tudi pripadajoča osnovna oblika zapisana s takimi črkami (pri tem bi morali upoštevati lematizacijska pravila v oblikoslovnem označevalniku Obeliks).

4. Napovedovanje skupin bi lahko preverili tudi z uporabo drugih klasifikatorjev, kot je npr. metoda naključnih gozdov (ang. *Random forests*).

5. Za dopolnjevanje paradigem drugih besednih vrst (tukaj mislimo predvsem na odprti besedni vrsti - glagol ter pridevnik), bi bilo treba orodje dopolniti z ustreznimi utežmi, ki so primerne za izbrano besedno vrsto. Pomožni program za pretvorbo Sloleksa iz XML formata v CSV format že generira datoteke za glagole ter pridevnike z ustreznimi značilkami.

Literatura

[1] Simon Krek, Kaja Dobrovoljc, Tomaž Erjavec, Sara Može, Nina Ledinek, Nanika Holz: Učni korpus ssj500k, *Center za jezikovne vire in tehnologije Univerze v Ljubljani*. Dosegljivo na: https://www.clarin.si/repository/xmlui/handle/11356/1052 [Dostopano 10. 3. 2017].

[2] Špela Arhar Holdt, Vojko Gorjanc: Korpus FidaPLUS: nova generacija slovenskega referenčnega korpusa, *Jezik in slovstvo*, št. 2, str. 97-110, 2007.

[3] Kaja Dobrovoljc, Simon Krek, Tomaž Erjavec: Leksikon besednih oblik Sloleks in smernice njegovega razvoja, v knjigi: Slovar sodobne slovenščine: problemi in rešitve (ur. Vojko Gorjanc, Polona Gantar, Iztok Kosem, Simon Krek), *Znanstvena založba Filozofske fakultete Univerze v Ljubljani*, 2015.

[4] Špela Arhar Holdt: Učni korpus SSJ in leksikon besednih oblik za slovenščino, *Jezik in slovstvo*, št. 3-4, str. 43-56, 2009.

[5] Sloleks. Dosegljivo na: http://www.slovenscina.eu/sloleks/opis/ [Dostopano 9. 1. 2017].

[6] Polona Gantar, Vojko Gorjanc, Nataša Logar, Simon Krek: Leksikografski opis slovenščine v digitalnem okolju, *Znanstvena založba Filozofske fakultete Univerze v Ljubljani*, 2015.

[7] Tomaž Erjavec, Peter Holozan, Simon Krek, Matej Pivec, Simon Rigač, Simon Rozman, Aleš Velušček: Specifikacije za leksikon (besednih oblik) - projekt "Sporazumevanje v slovenskem jeziku". Dosegljivo na: http://projekt.slovenscina.eu/Vsebine/Sl/Kazalniki/K3.aspx [Dostopano 10. 3. 2017].

[8] International Organization for Standardization: ISO 24613:2008 – Language resource management - Lexical markup framework (LMF). Dosegljivo na:
https://www.iso.org/standard/37327.html [Dostopano 10. 3. 2017].

[9] Tomaž Erjavec, Simon Krek: Oblikoskladenjske specifikacije in označeni korpusi JOS, *Šesta konferenca Jezikovne tehnologije*, 2008.

[10] Seznam oblikoskladenjskih oznak. Dosegljivo na:
http://nl.ijs.si/jos/msd/html-sl/msd.index.msds.html [Dostopano 7. 3. 2017].

[11] Machine Learning: What it is and why it matters. Dosegljivo na:
http://www.sas.com/en_us/insights/analytics/machine-learning.html [Dostopano 15. 1. 2017].

[12] Kevin P. Murphy: Machine learning: a probabilistic perspective, *MIT press*, 2012.

[13] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, Blaž Zupan: Orange: Data Mining Toolbox in Python, *Journal of Machine Learning Research*, št. 14, str. 2349-2353, 2013.

[14] R: What is R? Dosegljivo na:
https://www.r-project.org/about.html [Dostopano 16. 1. 2017].

[15] Wikipedia: R (programming language). Dosegljivo na: https://en.wikipedia.org/w/index.php?title=R_(programming_language) [Dostopano 6. 1. 2017].

[16] CRAN Task View: Cluster Analysis & Finite Mixture Models. Dosegljivo na: http://cran.cnr.berkeley.edu/web/views/Cluster.html [Dostopano 18. 1. 2017].

[17] Daniel P. Martin: Clustering Mixed Data Types in R. Dosegljivo na: https://www.r-bloggers.com/clustering-mixed-data-types-in-r/ [Dostopano 22. 1. 2017].

[18] Martin Maechler: Package cluster. Dosegljivo na: https://cran.r-project.org/web/packages/cluster/ [Dostopano 26. 2. 2017].

[19] Kevin P. Murphy: Naive bayes classifiers, *University of British Columbia*. Dosegljivo na: http://www.ic.unicamp.br/~rocha/teaching/2011s1/mc906/aulas/naive-bayes.pdf [Dostopano 10. 3. 2017].

[20] Wikipedia: Cross-validation (statistics). Dosegljivo na: https://en.wikipedia.org/w/index.php?title=Cross-validation_(statistics) [Dostopano 28. 2. 2017].