Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Žiga Simončič

Implementacija sistema za ocenjevanje esejev na podlagi koherence in semantične skladnosti

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: prof. dr. Zoran Bosnić

Ljubljana, 2020

Zahvala

Iskreno se zahvaljujem mentorju prof. dr. Zoranu Bosniću za vso pomoč, koristne nasvete in skrben pregled ter popravke dela.

Prav tako se zahvaljujem dr. Kaji Zupanc za dosedanje delo ter obrazložitev in napotke pri implementaciji.

Velika zahvala gre tudi članom Biolaba: Ajdi Pretnar za nenehno pomoč in testiranje implementacije, dr. Marku Toplaku za pomoč pri teptanju kritičnega hrošča in doc. dr. Tomažu Hočevarju za pregled vmesnika v zgodnji fazi razvoja.

Ogromna zahvala pripada tudi moji družini, sorodnikom in prijateljem, ki so me podpirali ne samo pri zadnjem vzponu na ta mali vrh, ampak na celotni poti, četudi je bila včasih zelo skalnata ali prepredena z razpotji in nevihtami. Hvala!

Žiga Simončič, 2020

Know your history.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
QWK	quadratic weighted kappa	kvadratno utežena kapa
EA	exact agreement	točna ocena/strinjanje
RF	random forest	naključni gozdovi
LR	linear regression	linearna regresija
AVG	average	povprečje

Povzetek

Naslov: Implementacija sistema za ocenjevanje esejev na podlagi koherence in semantične skladnosti

Cilj magistrskega dela je implementirati sistem za ocenjevanje esejev v angleškem jeziku. Zgledujemo se po metodologiji obstoječega sistema, ki poleg ocenjevanja sintakse uporablja tudi mere koherentnosti in semantične skladnosti. Metodologijo implementiramo v grafičnem okolju Orange, s prijaznim vmesnikom, opcijsko uporabo vektorskih vložitev za predstavitev besedila in možnostjo nadaljnjega razvoja sistema. Sistem evalviramo na podatkih, dostopnih na spletnem mestu Kaggle in, kolikor je mogoče, rezultate primerjamo z rezultati dosedanje metodologije in jih podrobno analiziramo. Poglobimo se tudi v izbiranje atributov za izboljšanje rezultatov. Glavni prispevki dela obsegajo (1) implementacijo sistema, (2) preprostost uporabe in (3) izboljšave dosedanjega dela, vključno z dodatnimi računskimi možnostmi in podrobno analizo izbiranja atributov za izboljšanje rezultatov.

Ključne besede

samodejno ocenjevanje esejev, semantična skladnost, Orange

Abstract

Title: Implementation of an automated essay evaluation system based on coherence and semantic consistency

The goal of this thesis is the implementation of an essay grading system. We lean heavily on the methodology of an existing system, which, besides using syntactical measurements, also uses coherence and semantic cosistency measures. We implement the methodology in the Orange data mining tool, with a firendly user interface, optional use of word embeddings for word representation and the possibility for further developments of the system. The system is evaluated on public datasets from the Kaggle website. The results are to the most possible extent compared with the results of the existing methodology and analyzed in detail. We also compare several attribute selection methods, which improve our results. Main contributions of this work are comprised of (1) implementation of the system, (2) ease of use and (3) improvements upon previous work, including additional computing options and detailed attribute selection analysis.

Keywords

automatic essay evaluation, semantic consistency, Orange

Poglavje 1

Uvod

Učitelji v izobraževalnih ustanovah so odgovorni za predajanje znanj velikemu številu učencev. Del učnega procesa je tudi pisanje esejev, ki jih morajo učitelji prebrati in oceniti. Ocenjevanje esejev ni le časovno potratno, ampak potencialno tudi nekoliko pristransko. Naloga učitelja je tudi, da napake označi, popravi in komentira celotno delo.

S pomočjo računalnika lahko ocenjevanje esejev olajšamo. Področje samodejnega ocenjevanja esejev se razvija že od 60. let prejšnjega stoletja. Dandanašnji sistemi za ocenjevanje esejev (tudi komercialni) se osredotočajo predvsem na sintaksno analizo, premalo pozornosti pa posvečajo semantiki [1]. To slabost obstoječih sistemov rešuje sistem SAGE, ki ga Zupanc opisuje v svoji disertaciji [2]. SAGE dosega zavidljivo napovedno točnost v primerjavi z drugimi sodobnimi sistemi, vendar je trenutna implementacija sistema v prototipni fazi in ni zrela za produkcijo.

Cilj magistrskega dela je uporabiti sodobne metode razvoja programske opreme za implementacijo sistema SAGE, ki bo zrel za javno uporabo oz. produkcijo. Delo obsega: implementacijo atributskih funkcij za opis besedila, analizo semantične koherence, implementacijo sistema za samodejno zaznavanje in povratno informacijo o napakah ter implementacijo vsega naštetega v grafično okolje, ki je uporabniku prijazno in preprosto za uporabo. Osnovne atributske funkcije obsegajo preproste statistike, kot so število znakov, besed,

povedi itd. ter različne mere berljivosti in podobnosti. Semantična koherenca je predstavljena z večdimenzionalnimi prostori drsečega okna, ki se premika skozi celotno besedilo. Z različnimi merami in analizo večdimenzionalnih prostorov ocenimo konsistentnost besedila in tok misli. Sistem za zaznavanje semantičnih napak v ozadju uporablja ontologijo, ki ji postopoma dodajamo iz besedila izluščena dejstva. Z logičnim sklepanjem nato ugotovimo, ali so trditve iz besedila logične ali ne. Za dosego teh ciljev smo se odločili, da uporabimo in sistem implementiramo v programskem okolju Orange.

Sistem je v Orangeu implementiran v obliki gradnikov (angl. widgets). Med seboj jih lahko povezujemo in kombiniramo, tako da smo uvoz datotek, gradnjo in testiranje modelov prepustili gradnikom, ki so v Orange-u že implementirani. Skupno smo implementirali tri gradnike — prvi implementira vse atributske funkcije, vključno s koherenco, drugi implementira sistem za analizo semantične skladnosti, tretji pa je namenjen evalvaciji modela po kvadratno uteženi kapi.

Dosedanje delo tudi nadgradimo z uporabo opcijskih vektorskih vložitev za predstavitev besed, podrobno analizo izbora najustreznejših atributov za izboljšanje rezultatov in primerjavo z nevronskimi modeli. Sistem in rezultate evalviramo v smislu pravilnega delovanja (natančnosti), vendar zaradi narave implementacije na ustreznih mestih omenimo tudi določene performančne pomisleke. Implementacija sistema je odprtokodna in na voljo na repozitoriju git.[1]

V 2. poglavju opredelimo teoretično ozadje sistema, skupaj s podrobnostmi implementiranih atributskih funkcij, poglavje 3 opredeljuje cilje in zgled vmesnikov ter predstavi podrobno implementacijo našega sistema, v poglavju 4 pa predstavimo dosežene rezultate, funkcionalnosti implementacije, podrobno analizo izbiranja atributov in primerjavo z nevronskimi modeli. Delo sklenemo z diskusijo in zaključkom v 5. poglavju.

[1] `https://github.com/venom1270/essay-grading`

Poglavje 2

Teoretično ozadje

V svetu obdelave podatkov imamo na voljo veliko orodij in pristopov k reševanju problemov. Pri obdelavi besedilnih podatkov, nizov oz. konkretnejše esejev imamo pred seboj niz znakov, ki v sebi hrani bogato količino informacij.

Objektivno ocenjevanje besedil sega že v sredino prejšnjega stoletja, ko so se začele pojavljati različne formule oz. berljivostne mere/berljivostni indeksi (angl. *readability index*), katerih cilj je bil ocena besedila po različnih merilih. V osnovi so to formule, ki merijo raznolikost besed v besedilu in na osnovi tega ocenijo zahtevnost besedila, ki se določi na podlagi prej definirane lestvice.

Zupanc in Bosnić [1] ter Zupanc [2] v svojem članku in disertaciji vse mere razvrstita v različne skupine. Zaradi preglednosti so tudi v nadaljevanju te mere predstavljene v podobni strukturi, kot sta jih uvrstila Zupanc in Bosnić. Predstavili bomo vse uporabljene mere, ki smo jih implementirali v sklopu našega dela.

Vsi v nadaljevanju opisani atributi tvorijo tri različne sisteme: *AGE*, *AGE+* in *SAGE* [1]. Posamezni sistem definira nabor atributov, ki jih izračunamo iz podanih besedil esejev in jih nato uporabimo za gradnjo napovednega modela (ocenjevanja esejev).

Skupek atributov osnovne sintaktične statistike, berljivostnih, leksikalnih, slovničnih in vsebinskih mer tvorijo sistem *AGE*. Atributom sistema *AGE*

dodamo atribute za merjenje koherence in to označimo kot sistem *AGE+*. Če vsem zgornjim atributom dodamo še nabor treh atributov, ki jih pridobimo s preverjanjem semantične skladnosti, govorimo o sistemu *SAGE*.

Razdelki v nadaljevanju podrobneje opišejo posamezne uporabljene atribute, na koncu poglavja, v razdelku 2.6, pa so za boljši pregled vsi atributi zbrani v tabelah.

2.1 Leksikalne mere

Osnovne mere predstavljajo skupine mer, ki temeljijo na plitvi analizi besedila, velikokrat so predstavljene v obliki ene formule. Merijo razgibanost besedila, raznolikost besed in razmerje med različnimi kategorijami besed.

Osnovo za izračun teh mer predstavljajo preproste statistične mere, kot so: število znakov, besed, povedi, povprečna dolžina besed in povedi, število kratkih/dolgih besed in povedi, število različnih besed ipd.

2.1.1 Mere berljivosti

Gunning Fog index, predstavljen v knjigi *The Technique of Clear Writing*, [3] je eden izmed prvih poskusov računske izmerljivosti berljivosti angleških besedil [4]. Izračuna se po spodnji formuli:

$$Gunning\ Fog = 0,4\left(\frac{št.\ besed}{št.\ povedi} + 100\ \frac{št.\ kompleksnih\ besed}{št.\ besed}\right) \quad (2.1)$$

Med kompleksne besede štejemo tiste besede, ki imajo vsaj tri zloge in niso (a) lastno ime, (b) sestavljene iz preprostejših besed (z vezajem), in (c) dvozložne besede z dodano končnico (*-ing*, *-ed* itd.). Formula nam vrne število, ki določa stopnjo berljivosti, ki je korelirana z ravnijo zahtevnosti besedila in se izrazi glede na stopnjo v izobraževalnem sistemu (npr. besedilo, primerno za osmi razred OŠ).

Flesch reading ease je populariziral Rudolph Flesch. Formula nam izračuna rezultat med 0 in 100, kjer večje število predstavlja lažje berljivo bese-

dilo. Za večino standardnih tehničnih dokumentov se vrednost giblje med 60 in 70 [4].

$$Flesch\ Reading\ Ease = 206,835 - \left(1,015\ \frac{št.\ besed}{št.\ povedi}\right) - \left(84,6\ \frac{št.\ zlogov}{št.\ besed}\right) \quad (2.2)$$

Flesch Kincaid readability index predstavlja poenostavitev Flesch-ove formule, ki jo je zasnoval J. Peter Kincaid. Vrednost, ki jo vrne ta poenostavljena formula, je v podobnem razponu kot pri indeksu *Gunning Fog*.

$$Flesch\ Kincaid\ Readability = 0,39\ \frac{št.\ besed}{št.\ povedi} + 11,8\ \frac{št.\ zlogov}{št.\ besed} - 15,59 \quad (2.3)$$

Dale-Chall readability formula se ponovno zgleduje po indeksu *Flesch reading ease* in ga nadgradi s štetjem *težkih besed*. Za težke besede dojemamo vse besede, ki se ne pojavijo na seznamu najpogostejših besed (teh je okrog 3.000) [5].

$$Dale\ Chall\ Readability\ = 15,79\ \frac{št.\ težkih\ besed}{št.\ besed} + 0,0496\ \frac{št.\ besed}{št.\ povedi} \quad (2.4)$$

Če je razmerje težkih besed večje od 5 %, h končnemu rezultatu prištejemo še 3,6365.

Automated readability index je (bil) namenjen samodejnemu izračunu berljivosti besedila pri pisanju na pisalne stroje. Podobno kot zgornje formule uporablja preproste konstrukte za izračun berljivosti. Rezultati so podobni drugim indeksom [6].

$$ARI = 4,71\ \frac{št.znakov}{št.\ besed} + 0,5\ \frac{št.\ besed}{št.\ povedi} - 21,43 \quad (2.5)$$

Simple measure of Gobbledygook se namesto na težavnost posameznih besed in zlogov osredotoči na besede, ki imajo več kot tri zloge (*večzložnice*). Za izračun ni treba več preštevati posameznih besed in zlogov, ampak samo prešteti število večzložnic in povedi [7].

$$SMOG = 1,0430 \sqrt{\check{s}t.\ ve\check{c}zlo\check{z}nic \frac{30}{\check{s}t.\ povedi}} + 3,1291 \tag{2.6}$$

LIX oz. *Lasbarhetsindex* je formula za izračun berljivosti besedila, ki je bila prvotno razvita na Švedskem, za švedska besedila, vendar se dobro obnese tudi na besedilih v drugih jezikih, vključno z angleškimi [8].

$$LIX = \frac{\check{s}t.\ besed}{\check{s}t.\ pik,\ dvopi\check{c}ij,\ v.\ za\check{c}etnic} + \frac{\check{s}t.\ dolgih\ besed \cdot 100}{\check{s}t.\ besed} \tag{2.7}$$

Namesto štetja pik, dvopičij in velikih začetnic lahko uporabimo število povedi.

Word variation index oz. *OVIX* je, podobno kot LIX, namenjen besedilom v švedščini. Od vseh omenjenih formul je ta najkompleksnejša. Meri leksikalno raznolikost (angl. *lexical variation*) oz. raznolikost besed (angl. *word variation*). Skupaj z mero *Nominal ratio* nakazujeta na vsebinsko/idejno gostoto besedila [9].

$$OVIX = \frac{\log(\check{s}t.\ besed)}{\log\left(2 - \frac{\log(\check{s}t.\ unikatnih\ besed)}{\log(\check{s}t.\ besed)}\right)} \tag{2.8}$$

Nominal ratio se izračuna tako, da število samostalnikov, predlogov in deležnikov delimo s številom zaimkov, prislovov in glagolov (angl. *nouns, prepositions, participles* in *pronouns, adverbs, verbs*) [9].

$$NR = \frac{\check{s}t.\ samostalnikov + \check{s}t.\ predlogov + \check{s}t.\ dele\check{z}nikov}{\check{s}t.\ zaimkov + \check{s}t.\ prislovov + \check{s}t.\ glagolov} \tag{2.9}$$

2.1.2 Mere leksikalne raznolikosti/bogatosti

Osnova leksikalne raznolikosti je razmerje različnih besed s številom vseh besed. To je osnova, ki ima veliko pomanjkljivost, in sicer odvisnost od dolžine

besedila. Pri daljših besedilih je neizogibno dejstvo, da se besede začnejo ponavljati, zato bo delež unikatnih besed manjši, s tem pa bo izračunana leksikalna raznolikost tudi manjša. S tem se implicira manjši besedni zaklad besedila. Osnovna formula se je čez čas nadgrajevala z novimi formulami, ki so pri ocenjevanju leksikalne različnosti upoštevale tudi dolžino besedila [10].

Type-Token ratio je prva in najosnovnejša mera za ocenjevanje leksikalne raznolikosti. Predstavljena je le s količnikom med številom različnih besed in številom vseh besed.

$$TTR = \frac{št.\ različnih\ besed}{št.\ besed} \tag{2.10}$$

Guiraud's index oz. *root type-token ratio* nadgradi prejšnjo formulo s korenom v imenovalcu.

$$Guiraud's\ index = \frac{št.\ različnih\ besed}{\sqrt{št.\ besed}} \tag{2.11}$$

Podobnih nadgradenj indeksov je še kar nekaj: *corrected TTR*, *Herdan index*, *Summer index*, *Mass index* itd. [11].

Yule's K ocenjuje konstantnost besedila v kontekstu ponavljanja besedišča. Namenjen je merjenju besednega zaklada besedila: večja vrednost pomeni manj bogato besedišče [12]. V splošnem je rezultat neodvisen od dolžine besedila, če je dolžina več kot tisoč besed, vendar je robusten tudi za krajša besedila [13].

$$Yule's\ K = 10^4 \frac{\sum_{r=1}^{N} r^2 V_r - N}{N^2} \tag{2.12}$$

N predstavlja število različnih besed, V_r pa število besed, ki se pojavijo r-krat.

D estimate je osnovan na temelju parametra D, ki predstavlja krivuljo na grafu TRR in števila različnih besed. Večji D pomeni večjo raznolikost besedila [14]. Originalno je formula zapisana v spodnji obliki:

$$TRR = \frac{D}{N}\left[\left(1 + 2\frac{N}{D}\right)^{\frac{1}{2}} - 1\right] \tag{2.13}$$

kjer izpostavimo D in dobimo končno obliko:

$$D = -\frac{N \cdot TTR^2}{2(TTR - 1)} \tag{2.14}$$

N predstavlja število različnih besed, TTR pa mero *type-token ratio* (2.10).

Hapax legomena predstavlja število besed, ki se v besedilu pojavijo samo enkrat [13]. To število normaliziramo s številom besed v besedilu.

$$HL = \frac{\check{s}t.\ besed\ z\ eno\ pojavitvijo}{\check{s}t.\ vseh\ besed} \tag{2.15}$$

Advanced Guiraud's index je izboljšava mere *Guiraud's index* (2.11). V osnovni formuli se v števcu število različnih besed zamenja s številom različnih *naprednejših besed*, z argumentom, da z upoštevanjem vseh besed ne dobimo prave informacije o bogatosti besedila, temveč je pomembnejša uporaba kompleksnejših, naprednih besed [15]. Za identifikacijo naprednih besed lahko uporabimo seznam osnovnih besed, podobno kot pri formuli *Dale-Chall* (2.4).

$$Advanced\ Guiraud's\ index = \frac{\check{s}t.\ različnih\ naprednih\ besed}{\sqrt{\check{s}t.\ besed}} \tag{2.16}$$

2.2 Slovnica

V sklopu ocenjevanja slovnice preverjamo skladenjsko strukturo povedi in slovnične napake. Za ugotavljanje skladenjske strukture povedi uporabimo oblikoskladenjsko označevanje (angl. *part-of-speech (PoS) tagging*), ki nam za vsako besedo v besedilu določi njeno vlogo. Bodisi je to samostalnik, glagol, pridevnik, predlog ipd. Skladenjsko razčlenjevanje (angl. *dependency parsing*) v povedi odkrije skladenjska razmerja, npr. odvisnosti, prilastke itd. Z uporabo tovrstnega označevanja lahko odkrijemo strukturo posameznih povedi in stavkov. Na osnovi teh tehnik izpeljemo naslednje mere za naš sistem.

Število različnih oblikoskladenjskih oznak — preštejmo število različnih oblikoskladenjskih vlog, ki nastopajo v celotnem besedilu.

Višina skladenjskega drevesa povedi — za izračun končne vrednosti se višina drevesa povedi povpreči čez celotno besedilo.

Število glagolskih besed.

Število posameznih oblikoskladenjskih oznak — število oblikoskladenjskih oznak v besedilu za vsako kategorijo.

Število slovničnih in pravopisnih napak:

- splošnih slovničnih napak (angl. *grammar*),
- napačnih črkovanj (angl. *spellcheck*),
- rabe velike in male začetnice (angl. *capitalization*) ter
- uporabe ločil (angl. *punctuation*).

2.3 Vsebinske mere

Vsebinske mere primerjajo podobnost dokumentov glede na njihovo vektorsko predstavitev. Ta sklop mer je osnovan na primerjavi neocenjenih esejev z že ocenjenimi eseji, kar pomeni, da za uporabo teh mer potrebujemo neki vzorec ročno ocenjenih esejev. V našem delu smo uporabili dve tehniki vektorskih predstavitev.

TF-IDF (angl. *term frequency—inverse document frequency*) je numerična statistika, ki predstavlja pomembnost besede v dokumentu glede na korpus. Izračuna se s številom pojavitev posamezne besede t v dokumentu d (*TF(t, d)*) in pomembnostjo besede t skozi celoten korpus D (*IDF(t, D)*).

$$\begin{aligned} TF\text{-}IDF(t,d,D) &= TF(t,d) \cdot IDF(t,D) \\ &= \frac{|\{t \in d\}|}{|\{w \in d\}|} \cdot \log \frac{|\{d \in D\}|}{||\{d \in D : t \in d\}||} \end{aligned} \tag{2.17}$$

GloVe (angl. *Global Vectors*) [16] vektorske vložitve poleg pomembnosti besed skušajo zajeti tudi pomen in relacije med besedami. Besedi *king* in *queen* bi morali imeti v novem vektorskem prostoru podobno relacijo kot besedi *man* in *woman*. Z vektorskimi vložitvami GloVe poskušamo besedila predstaviti z vektorji, ki poleg številske predstavitve nosijo tudi neki pomen.

Dokumente, predstavljene s številkami, vektorji, lahko med seboj primerjamo. V našem delu za ugotavljanje podobnosti besedil uporabimo kosinusno podobnost. Z njo merimo kot med dvema vektorjema. Bolj kot vektorja sovpadata, večja je kosinusna podobnost, ki ima razpon med vključno nič in ena.

$$podobnost(a, b) = \frac{a \cdot b}{||a|| \times ||b||} = \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}} \tag{2.18}$$

Kosinusna podobnost je ključna za izračun naslednjih mer za primerjavo esejev:

Kosinusna podobnost z izvornim besedilom pri tistih zbirkah, ki izvorno besedilo imajo.

Ocena eseja, ki mu je esej (d) glede na kosinusno podobnost najbolj podoben.

Podobnost z eseji, ki imajo najboljšo oceno. Tukaj vzamemo največjo podobnost z eseji, ki imajo najboljšo oceno. Eseji z oceno i so predstavljeni z oznako D_i, esej, ki ga obravnavamo, ima oznako d.

$$podobnost\_najboljsi(d, D_i) = max(podobnost(d, D_i)) \tag{2.19}$$

Pattern cosine [17] je nadgradnja iskanja ocene eseja z najvišjo podobnostjo. Pri ugotavljanju ocene esejev, ki jim je trenutni esej (tisti, ki ga primerjamo) podoben, predpostavljamo, da je podobnost pozitivno korelirana s pravimi ocenami. Vendar, če je najvišja podobnost korelirana

s pravimi ocenami, potem je tudi druga najvišja podobnost pozitivno korelirana z ocenami. Nasprotno je ocena esejev z najnižjo podobnostjo negativno korelirana z ocenami. Iz tega sklepamo, da uporaba podobnosti pri vseh ocenah pripelje do celovitejše informacije. Na osnovi teh sklepanj sestavimo spodnjo formulo.

$$Pat.Cos = \sum_{i}^{k} S_i R_i \tag{2.20}$$

S S_i označimo ocene, z R_i pa zaporedje podobnosti po ocenah i. Vseh ocen je k.

Utežena vsota vseh kosinusnih korelacijskih vrednosti (angl. *weighted sum of all cosine correlation values*) [17] je podobna izpeljanka, kot *Pattern cosine*. Za osnovo vzamemo primerjanje trenutnega eseja z najboljšimi eseji, dodamo pa še podobnosti z drugo najboljšo oceno, nato tretjo in tako naprej do najslabše ocene. Vse podobnosti ustrezno utežimo, tako da podobnost z bolje ocenjenimi eseji prispeva k boljšemu končnemu rezultatu, podobnost s slabšimi pa rezultat poslabša. Če imamo npr. šest različnih ocen, zgornjo polovico členov utežimo s +1, spodnjo pa z −1.

$$Val.Cos = 1\ Cos6 + 1\ Cos5 + 1\ Cos4 - 1\ Cos3 - 1\ Cos2 - 1\ Cos1 \tag{2.21}$$

2.4 Mere koherentnosti

Merjenje koherentnosti besedila snujemo na predpostavki, da se semantična vsebina eseja skozi različne odseke besedila postopoma spreminja [1]. Začnemo z delitvijo eseja na več zaporednih prekrivajočih delov, ki jih dobimo s premikanjem okna skozi celoten esej. Za korak smo vzeli 10 besed in velikost okna kot 25 % povprečnega števila besed v korpusu. Primer: če je povprečno število besed v korpusu 280, dolžina trenutnega eseja pa 320, smo s premikanjem okna dobili 26 delov ($\frac{320-280*25\%}{10} + 1$). Če je esej prekratek oz. okno

za ta esej preveliko, se velikost okna in korak razpolovita (tudi večkrat, če je potrebno).

Vsak del eseja nato s pomočjo TF-IDF (2.17) ali GloVe (2.3) pretvorimo v vektorski prostor. Glede na zgornjo predpostavko bi morali posamezni zaporedni deli koherentnih esejev v večdimenzionalnem vektorskem prostoru biti relativno blizu.

2.4.1 Osnovne mere koherentnosti

Osnovne mere koherence merijo razdaljo med posameznimi deli eseja v vektorskem prostoru. Za vsako mero izmerimo razdaljo na dva načina: enkrat z uporabo evklidske razdalje in enkrat z uporabno kosinusne podobnosti (2.18).

Povprečna razdalja med sosednjimi točkami v vektorskem prostoru (deli eseja so predstavljeni s točkami).

Minimalna in maksimalna razdalja med sosednjimi točkami ter njihov **kvocient**.

Povprečna razdalja med dvema točkama ocenjuje, kako dobro se ideja obdrži in prenaša skozi besedilo.

Maksimalna razdalja med dvema točkama meri premer kroga, ki obdaja vse točke v vektorskem prostoru in tako meri širino koncepta o katerem govori esej.

Razdalja Clark in Evans' do najbližjega soseda vsake točke, kar meri prostorske relacije med točkami.

$$R = \frac{\frac{\sum_{i=1}^{N} r_i}{N}}{\frac{1}{2\sqrt{N}}} = \frac{2\sqrt{N}\sum_{i=1}^{N} r_i}{N} \tag{2.22}$$

r predstavlja razdaljo od dane točke i do najbližjega soseda, N pa število točk.

Povprečna razdalja do najbližjega soseda meri hitrost razvoja ideje skozi esej.

Kumulativna frekvenčna porazdelitev G za razdalje najbližjih sosedov, kjer r predstavlja razdaljo od dane točke do najbližjega soseda, $\overline{r}$ povprečna razdalja do najbližjega soseda, N pa število točk. Mera izraža delež vsebine, ki odstopa od glavne ideje.

$$G(r) = \frac{|r \leq \overline{r}|}{N} \tag{2.23}$$

2.4.2 Prostorska analiza

Drugi sklop atributov se osredotoča na prostorske značilnosti vektorjev, kjer poskušamo pridobiti informacije o prostorski statistiki in vzorcih. Naslednje mere v splošnem na osnovi središčnih točk (centroidov) merijo prostorsko razpršenost.

Povprečna evklidska razdalja do centroida za vsako točko v eseju. Tu se meri razpršenost v točkastem vzorcu.

Minimalna in maksimalna evklidska razdalja do centroida za vsako točko in njihov **kvocient**. Meri največje odstopanje od glavne ideje.

Standardna razdalja predstavlja prostorski ekvivalent standardnemu odklonu (angl. *standard deviation*), ki meri absolutni odklon točk v prostoru.

$$S_D = \sqrt{\frac{\sum_{k=1}^{n}\sum_{i=1}^{N}(D_i^k - \overline{D_c^k})^2}{N}} \tag{2.24}$$

$D_i^k, k = 1...n, i = 1...N$ predstavlja k-to komponento točke i, $\overline{D_c^k}$ k-to komponento centra (središčne točke, centroida), n število dimenzij, N število točk. Podobno kot pri standardnemu odklonu, zaradi kvadriranja odstopajoče vrednosti močno vplivajo na rezultat. To nam tudi omogoča, da ugotovimo kateri odseki v eseju odstopajo od povprečja.

Relativna razdalja je mera za relativno prostorsko razpršenost. Izračunamo jo z deljenjem standardne razdalje z vrednostjo, ki opisuje območje,

ki ga v prostoru zavzemajo točke, torej maksimalna razdalja katere koli točke do središčne točke. Označimo jo z d_{max}.

$$R_D = \frac{S_D}{d_{max}} \tag{2.25}$$

Determinanto matrike razdalj uporabimo kot mero prostorske razpršenosti, ki nam omogoča merjenje razpršenosti in širino vsebine.

2.4.3 Prostorska korelacija

Mere prostorske korelacije izražajo ali so točke v prostoru blizu skupaj (pozitivna korelacija) ali pa so narazen, razpršene (negativna korelacija). Omogočajo oceno lokalne in globalne koherence eseja. Če se esej izkaže za izrazito pozitivno koreliranega, ga dojemamo kot dober esej, ki je zgledno strukturiran in je njegova vsebina povezana od začetka do konca.

Tipične mere prostorske korelacije so *Moran's I*, *Geary's C* in *Gettis's G*. Te mere so za naš vektorski prostor zaradi dodatne dimenzije povprečene preko vseh komponent.

Moran's I oceni splošen vzorec gručenja in razpršenosti točk. Razpon mere je od -1 do 1, kjer večja pozitivna vrednost nakazuje pozitivno prostorsko korelacijo, kar pomeni, da so sosednje točke blizu. Nasprotno velja za negativne vrednosti. Vrednosti okoli 0 nakazujejo na naključno prostorsko porazdelitev točk.

$$I = \frac{N}{S} \cdot \frac{1}{n} \sum_{k=1}^{n} \left[\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}(D_i^k - \overline{D_c^k})(D_j^k - \overline{D_c^k})}{\sum_{i=1}^{N}(D_i^k - \overline{D_c^k})^2} \right] \tag{2.26}$$

$D_i^k, k = 1...n, i = 1...N$ predstavlja k-to komponento točke i, D_c^k k-to komponento središčne točke, n število dimenzij, N število točk in S vsoto vseh uteži w_{ij}. Uteži w_{ij} se nastavijo za vsak par točk, in sicer z vrednostjo $w_{ij} = 1$, če sta točki i in j sosednji ter z vrednostjo $w_{ij} = 0$ sicer.

Geary's C je inverzno sorodna mera *Moran's I*. V tem primeru na končni rezultat ne vpliva toliko razdalja od središčne točke, ampak se meri odstopanje posameznih sosednjih točk.

$$C = \frac{N-1}{2S} \cdot \frac{1}{n} \sum_{k=1}^{n} \left[\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}(D_i^k - D_j^k)^2}{\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}(D_i^k - \overline{D_c^k})^2} \right] \tag{2.27}$$

$D_i^k, k = 1...n, i = 1...N$ predstavlja k-to komponento točke i, D_c^k k-to komponento središčne točke, n število dimenzij, S vsoto vseh uteži w_{ij}, N število točk in w_{ij} so enake uteži, kot so opisane zgoraj.

Getis's G omogoča ugotavljanje vzorcev točk v lokalnem območju. Meri koncentracijo oz. razpršenost vseh parov točk, ki so v določenem območju.

$$G(d) = \frac{1}{n} \sum_{k=1}^{n} \left[\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}(d) D_i^k D_j^k}{\sum_{i=1}^{N} \sum_{j=1}^{N} D_i^k D_j^k} \right] \tag{2.28}$$

$D_i^k, k = 1...n, i = 1...N$ predstavlja k-to komponento točke i, D_c^k k-to komponento središčne točke, n število dimenzij, N število točk in d predstavlja povprečno razdaljo med dvema točkama v tem prostoru. Uteži $w_{ij}(d)$ imajo vrednost $w_{ij}(d) = 1$, če sta točki i in j oddaljeni največ za d in $w_{ij}(d) = 0$ sicer.

2.5 Semantična analiza z ontologijami

Eden izmed glavnih prispevkov dela Zupanc in Bosnić [1] je uporaba ontologij za ugotavljanje semantične skladnosti. Ta postopek je uporaben na dva načina: z njim pridobimo nekaj dodatnih atributov, ki jih lahko uporabimo pri napovedovanju ocen esejev, dodatno pa nam ta postopek tudi sporoči, kje so semantične napake oz. neskladja. Slednja funkcionalnost je zelo pomembna, saj tako učenec prejeme neposredno informacijo o napakah v eseju.

Postopek temelji na uporabi ontologije, v katero postopoma dodajamo v relacije strukturirane stavke in sproti preverjamo skladnost ontologije.

2.5.1 Ontologija

Ontologije predstavljajo bazo znanja, ki je predstavljena v obliki grafa. Za osnovo smo vzeli ontologijo COSMO (angl. *Common Semantic Model*), ki združuje elemente različnih ontologij, kot so OpenCyc, SUMO, BFO in DOLCE. Ontologija COSMO je predstavljena v semantičnem jeziku OWL[1] (*Web Ontology Language*), ki omogoča gradnjo kompleksnih shem različnih konceptov, dejstev in medsebojnih relacij. OWL je logični jezik, kar nam glede na dodane relacije omogoča sklepanje in odkrivanje napak.

Osnovna struktura ontologije je predstavljena s "trojicami" v obliki *(osebek (angl. subject), relacija/predikat, predmet (angl. object))*. Relacija lahko predstavlja omejitev, konceptualno povezavo (npr. *(Alice, isMotherOf, Bob)*) ali definira tip. V implementaciji smo za predstavitev trojic uporabili jezik RDF, ki je podoben jeziku OWL, vendar ni logični jezik. To pomeni, da RDF ne podpira logičnih omejitev in sklepanja ter tako omogoča veliko bolj prosto strukturo in lažje upravljanje trojic.

2.5.2 Semantična analiza

Ontologija COSMO že definira večino konceptov, ki jih uporabljamo v vsakdanjem življenju. V primeru, da bi hoteli ontologiji dodati dodatna posebna znanja, to lahko storimo. V našem primeru je poleg nekaterih korpusov tudi izvorno besedilo, na osnovi katerega so bili eseji spisani. Če želimo znanje tega izvornega besedila upoštevati pri ocenjevanju esejev, ga dodamo v osnovno ontologijo, preden se lotimo pregledovanja esejev. Izvorno besedilo dodamo v ontologijo po enakem postopku kot eseje in je razložen spodaj.

Najprej za vsak esej naredimo kopijo besedila. Potem za posamezni esej poiščemo koreference v besedilu (angl. *coreference resolution*). Določena entiteta — oseba, organizacija, lokacija itd. — se lahko v besedilu pojavi večkrat na posreden način, npr. z zaimki. Ugotavljanje referenc nam omogoča odkrivanje teh posrednih referenc na določene entitete in zamenjavo z neposredno

[1] `https://www.w3.org/OWL/`

entiteto. Primer: "*Bob likes pizza. He eats it all the time.*" nadomestimo z "*Bob likes pizza. Bob eats pizza all the time*". Uporabili smo SpaCyjev rezreševalnik koreferenc[2].

Naslednji korak sta razčlenitev besedila na posamezne povedi in ekstrakcija informacij s pomočjo sistema OpenIE (angl. *Open Information Extraction*). V tem koraku posamezne povedi pretvorimo v eno ali več trojic, ki opišejo relacije, izražene v povedi in so primerne za logično obdelavo. V ta korak je vključena lematizacija, postopek pretvorbe posameznih besed v njihovo osnovno obliko, torej brez končnic in drugih skladenj. Za zgornji primer bi tako dobili dve trojici: *(Bob, like, pizza)* in *(Bob, eat, pizza)*. Nekateri sistemi za ekstrakcijo informacij upoštevajo tudi časovno informacijo, tako da bi v primeru druge povedi kot četrti element "trojice" dobili tudi "*all the time*", vendar te informacije ne upoštevamo. Uporabili smo sistem za ekstrakcijo ClausIE [18], podpiramo pa tudi možnost uporabe sistema OpenIE5, saj je po študiji [19] njegov predhodnik (OpenIE4) dosegal najboljše rezultate.

Vse pridobljene trojice nato postopoma dodajamo v ontologijo, obenem pa preverjamo njeno skladnost. Skladnost preverjamo z logičnim sklepalnikom HermiT, ki vrača dva tipa napak. Prvi tip napak se zgodi, ko ima neki razred (*owl:class*) prirejene entitete, za katere sklepalnik implicitno ugotovi, da jih ne sme imeti (*unsatisfiable case*). Ontologijo s tovrstno napako lahko še vedno uporabljamo in sklepamo naprej. Drugi tip napak pa se sproži, ko se s sklepanjem ugotovi logična napaka — nekonsistentna ontologija (angl. *inconsistent ontology*). Do takšnih napak pride po navadi zaradi neposrednih nasprotij (npr. *owl:disjointWith* med dvema relacijama, ki pravi, da entiteta ne more imeti obeh relacij hkrati).

Trojice, pridobljene z metodo OpenIE, obravnavamo zaporedno. Vsak element posamezne trojice dodatno prečistimo — odstranimo morebitna ločila in znake. Tu pazimo predvsem na poševnice, saj so vse entitete predstavljene v obliki URI (*Universal Resource Identifier*) in lahko uporaba doda-

[2] `https://spacy.io/`

tnih poševnic v nazivu povzroči težave. Za osebek (prvi element trojice) nato pogledamo, ali v ontologiji že obstaja. Če obstaja, nadaljnja akcija ni potrebna, v nasprotnem primeru pa z uporabo taksonomije WordNet[3] najdemo vse sopomenke osebka in pogledamo, ali so te že v ontologiji. V primeru, da še vedno ne najdemo ujemanja, preverimo še nadpomenke in osebek v ontologijo dodamo kot podrazred najdene nadpomenke (s pomočjo relacije *subClassOf*). V najslabšem primeru, ko ne najdemo ničesar od naštetega, v ontologijo dodamo osebek kot novo samostojno vozlišče (tipa *owl:class*). V vsakem primeru nato v ontologijo dodamo povezave na nadpomenke in protipomenke (če teh v ontologiji ni, jih dodamo). Enako storimo za predmet in predikat, z razliko, da pri predikatu poskušamo najti tudi nasprotne relacije in jih označiti kot disjunktne (*disjointWith*). Po kakršnem koli dodajanju v ontologijo zaženemo logični sklepalnik HermiT in beležimo rezultat. Na koncu v ontologijo dodamo še celotno trojico in ponovno zaženemo logični sklepalnik. To ponavljamo, dokler ne obdelamo vseh trojic.

Na podlagi povzročenih tipov napak osnujemo tri dodatne atribute, ki jih lahko uporabimo pri napovedovanju ocen esejev:

1. Število neizpolnjenih primerov — beleži število prvih tipov napak. Po navadi se prožijo po dodajanju novih entitet v ontologijo.

2. Število napak nekonsistentne ontologije — drugi tip napak. Prožijo se ob dodajanju celotne trojice v ontologijo.

3. Skupno število napak — vsota zgornjih dveh.

Dodatno lahko s tem postopkom ugotovimo, kje so semantična/logična neskladja. Eden izmed načinov je ročno spremljanje napak. Ob zaznani napaki po dodajanju trojice pogledamo vse morebitne disjunktne relacije predikata in njegovih nadpomenk (nadrazredov). Ko najdemo ustrezno nasprotno relacijo, si jo zapomnimo in jo na koncu procesiranja izpišemo. Drugi način nam omogoča logični sklepalnik HermiT. Omogočimo lahko, da beleži

[3] `https://wordnet.princeton.edu/`

pot sklepanja in jo ob morebitnih napakah tudi izpiše. Takšno sklepanje je po izkušnjah nekoliko dolgotrajnejše kot sicer.

2.6 Povzetek

Glede na uporabljene atribute govorimo o treh različnih sistemih: *AGE*, *AGE+* in *SAGE*. Sistem *AGE* predstavlja skupek atributov osnovne sintaktične statistike, berljivostnih, leksikalnih, slovničnih in vsebinskih mer. Tabela 2.1 povzema vse atribute, ki pripadajo tem skupinam. Ker iz literature ni bilo razvidno, ali se pri številu znakov upošteva presledke, smo ta atribut razbili na dva — število znakov s presledki in število znakov brez presledkov. Štiri atribute smo pridobili tudi z uporabo celotnega nabora oblikoskladenjskih oznak.

Atributom sistema *AGE* dodamo atribute za merjenje koherence in s tem dobimo sistem *AGE+*. Atributi za merjenje koherence so zbrani v tabeli 2.2. Simbol zvezdica (*) označuje, kateri atributi se izračunajo v dveh različicah — enkrat z evklidsko razdaljo in drugič s kosinusno razdaljo.

Če vsem zgornjim atributom dodamo še nabor treh atributov, ki jih pridobimo s preverjanjem semantične skladnosti, govorimo o sistemu *SAGE*. Atributi so zbrani v tabeli 2.3.

Tabela 2.1: Skupek atributov, ki tvorijo osnovni sistem AGE.

Osnovne statistike	Mere berljivosti	Leksikalna raznolikost
1. Število znakov (s presledki)	15. Gunning fog index (2.1)	24. Type-token ratio (2.10)
2. Število znakov (brez presledkov)	16. Flesch reading ease (2.2)	25. Guiraud's index (2.11)
3. Število besed	17. Flesch-Kincaid grade level (2.3)	26. Yule's K (2.12)
4. Število dolgih besed	18. Dale-Chall readability (2.4)	27. D estimate (2.13)
5. Število kratkih besed	19. Automated readability index (2.5)	28. Število unikatnih besed
6. Najpogostejša dolžina besed	20. SMOG (2.6)	29. Advanced Guiraud (2.16)
7. Povprečna dolžina besed	21. LIX (2.7)	
8. Število povedi	22. Word variation index (2.8)	
9. Število dolgih povedi	23. Nominal ratio (2.9)	
10. Število kratkih povedi		
11. Najpogostejša dolžina povedi		
12. Povprečna dolžina povedi		
13. Število različnih besed		
14. Število mašil (angl. *stopwords*)		

Slovnica	Vsebina
30. Število različnih PoS oznak	73. Podobnost z izvornim besedilom
31. Povp. višina stavčnega drevesa	74. Ocena najbolj podobnih esejev
32. Število glagolskih besed	75. Podobnost esejev z najvišjo oceno
33. Število slovničnih napak	76. Pattern Cosine (2.20)
34. Število črkovalnih napak	77. Korelacijske vrednosti (2.21)
35. Število napak z začetnico	
36. Število napačnih uporab ločil	
37.–72. Število posameznih PoS-oznak	

Tabela 2.2: Dodatne mere koherence, ki skupaj s prejšnjimi tvorijo sistem AGE+.

Osnovne mere	Prostorska analiza
78.–79. Povprečna razdalja med sosednjimi točkami*	93.–94. Povprečna razdalja do središčne točke*
80.–81. Najmanjša razdalja med sosednjimi točkami*	95.–96. Najmanjša razdalja do središčne točke*
82.–83. Največja razdalja med sosednjimi točkami*	97.–98. Največja razdalja do središčne točke*
84.–85. Kvocient najmanjše in največje razdalje*	99.–100. Kvocient najmanjše in največje razdalje*
86.–87. Povprečna razdalja med točkami*	101. Standardna razdalja (2.24)
88.–89. Največja razdalja med točkami*	102. Relativna razdalja (2.25)
90. Clark and Evans (2.22)	103. Determinanta matrike razdalj
91. Povprečna razdalja do najbližjega soseda	
92. Kumulativna frekvenca porazdelitve (2.23)	

Prostorska avtokorelacija
104. Moran's I (2.26)
105. Geary's C (2.27)
106. Gettis's G (2.28)

Tabela 2.3: Dodatni atributi za sistem SAGE

Semantična skladnost
107. Število napak pri dodajanju razredov in entitet v ontologijo
108. Število napak pri dodajanju trojic v ontologijo
109. Število vseh napak

Poglavje 3

Zasnova in implementacija

3.1 Zasnova

Cilj dela je implementacija sistema za ocenjevanje esejev na način, ki bo preprost za uporabo in s tem primeren za širšo uporabo. Podrobneje si cilje razdelamo kot:

1. algoritmično implementacijo sistema za ocenjevanje esejev, kot je opisan v članku avtorjev Zupanc in Bosnić [1] in disertaciji Kaje Zupanc [2],

2. sistem izboljšati z vpeljavo izbirnih vektorskih vložitev za predstavitev besed in preprosto uporabo različnih napovednih modelov za ocenjevanje esejev ter

3. vse skupaj umestiti v grafično okolje, ki bo uporabniku prijazno.

Za dosego teh ciljev smo se odločili sistem implementirati v grafičnem okolju Orange,[1] ki se uporablja za podatkovno rudarjenje. S tem smo pokrili prvi in tretji cilj, saj je namen Orangea hitra, preprosta in uporabniku prijazna grafična uporaba metod strojnega učenja ter gradnja različnih modelov.

[1] `https://orange.biolab.si/`

Drugi cilj poskušamo izpolniti z vpeljavo izbire uporabniku in splošne svobode, ki jo Orange omogoča (združljivost sistema z drugimi funkcionalnostmi Orange). Podrobnosti o Orangeu so opisane v poglavju o implementaciji (poglavje 3).

V sklopu svojega dela se je Zupanc osredotočila na (v času njenega raziskovanja že zaključeno) tekmovanje samodejnega ocenjevanje esejev, ki ga je gostil Kaggle.[2] Na tem tekmovanju so se pomerili različni sistemi, s katerimi je Zupanc primerjala svoj sistem. Najboljša mesta na končni lestvici so večinoma zasedali komercialni sistemi za ocenjevanje esejev, nekaj pa je bilo tudi po meri narejenih uporabniških modelov. Komercialni sistemi, kot so PEG,[3] e-rater[4] in IntelliMetric,[5] imajo že dolgo zgodovino in s tem velik tržni delež ter izpopolnjen finančni model. V času raziskovanja noben od naštetih ni ponujal brezplačne različice sistema.

Sistem Lexile,[6] ki se je na tekmovanju sicer odrezal nekoliko slabše od prej naštetih, ponuja prijazen spletni vmesnik z možnostjo testiranja okrnjenega delovanja sistema. Slika 3.1 prikazuje analizo kratkega eseja. Na desni strani opazimo, da se sistem osredotoča na sintaktično analizo — dolžino povedi, pogostost besed itd.

Zanimivo je tudi odprtokodno orodje LightSIDE[7], ki ni namenjeno samo ocenjevanju esejev, ampak obdelavi besedil nasploh. Slika 3.2 prikazuje videz orodja. Vidimo, da orodje obsega celoten postopek od ekstrakcije atributov do gradnje in evalvacije modelov. Z vidika splošne celovite rešitve, ki je relativno preprosta za uporabo (in zanjo ne potrebujemo znanja programiranja), je to orodje zelo podobno Orangeu, vendar je slednji vseeno bolj splošen in prilagodljiv. Orange že sam po sebi implementira veliko funkcionalnosti, ki jih vključuje LightSIDE, zato pri naši implementaciji tako natrpan vmesnik

[2] `https://www.kaggle.com/`

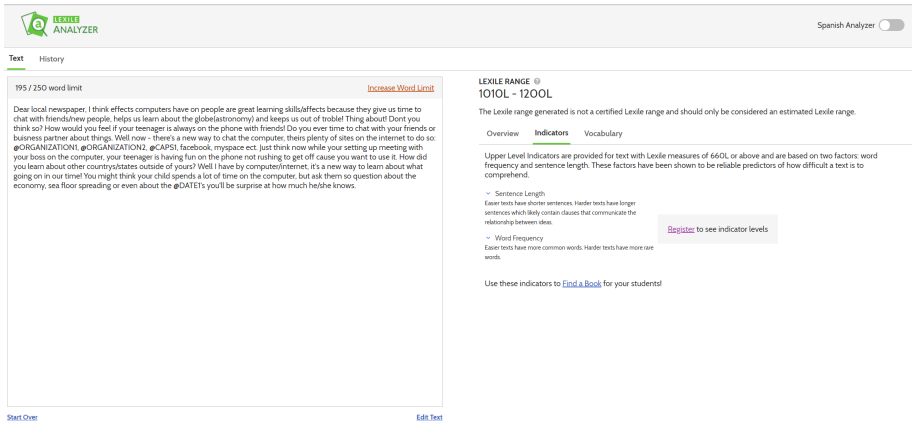
[3] `https://www.measurementinc.com/products-services/automated-essay-scoring`

[4] `https://www.ets.org/`

[5] `http://www.intellimetric.com/direct/`

[6] `https://hub.lexile.com/`

[7] `https://github.com/LightSideWorkbench/LightSide`

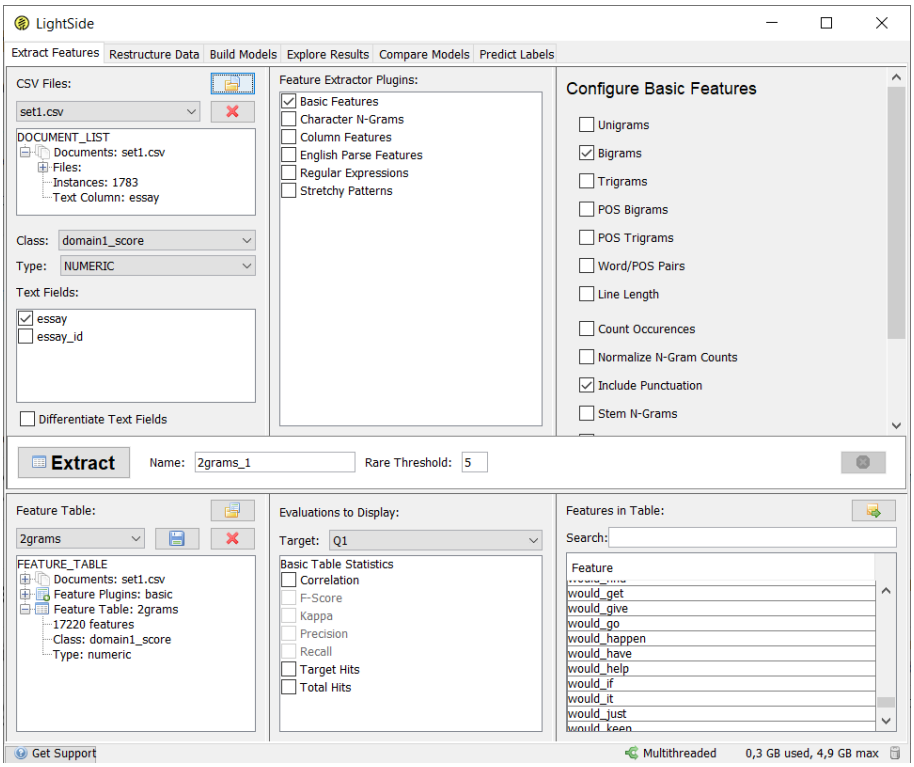


Slika 3.1: Zaslonska slika spletnega vmesnika sistema Lexile

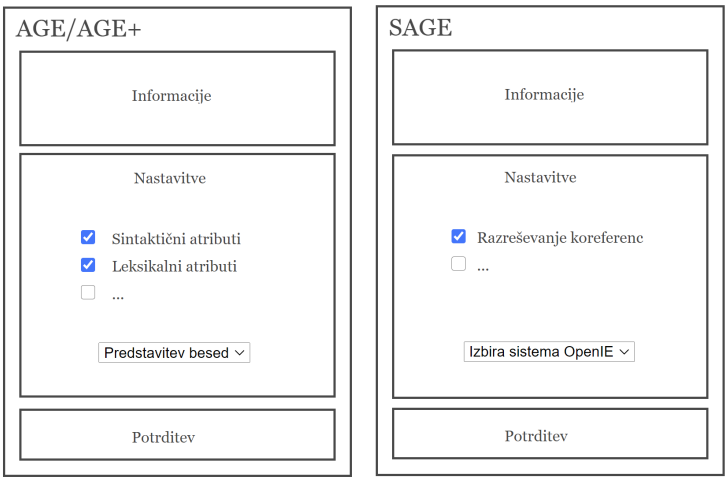
ni bil potreben. V skladu z vizijo Orange-a smo se poskušali nagibati k preprostejšemu vmesniku v slogu sistema Lexile, vendar tako, da uporabniku še vedno prepustimo del nadzora nad delovanjem.

V sklopu implementacije je bilo treba implementirati tri sisteme: AGE, AGE+ in SAGE. Vsak sistem nadgrajuje prejšnjega z dodatnimi atributi. Z vidika implementacije sta si AGE in AGE+ relativno podobna, SAGE pa zahteva zelo drugačen pristop. Zato smo se odločili implementacijo razbiti na dva dela: AGE/AGE+ in SAGE. Do atributov sistema AGE in AGE+ bomo dostopali preko enakega vmesnika. Kateri sistem bomo želeli uporabiti, bomo izbrali preko nastavitev. Sistem SAGE pa bo ločen in namenjen samo pridobivanju dodatnih atributov (tabela 2.3), ki jih potem lahko združimo z AGE/AGE+. Osnutek vmesnikov je prikazan na sliki 3.3.

Podrobnosti in končni videz ter uporaba so opisani v nadaljevanju (razdelka 3.3 in 3.5). Čeprav navzven gradniki delujejo preprosto, je implementacija terjala kar nekaj časa, predvsem pri gradniku za ugotavljanje semantične skladnosti. Pri tem gradniku je bilo treba v ozadju povezati različne tehnike in sisteme na način, da je navzven vse dostopno prek enega ali nekaj gumbov. V poglavju 4 je tudi podrobna analiza rezultatov, ki je razdeljena na dva dela: analizo sistema AGE/AGE+ (razdelek 4.3) in analizo sistema SAGE



Slika 3.2: Zaslonska slika odprtokodnega orodja LightSIDE



Slika 3.3: Zasnova vmesnika za sistema AGE, AGE+ (levo) in SAGE (desno)

(razdelek 4.4). Obe analizi vključujeta tudi uporabo nekaterih metod izbiranja atributov, s katerimi izboljšamo rezultat. Ker sistem temelji na klasični ekstrakciji značilnosti/atributov iz besedila, nevronski modeli pa postajajo gonilna sila marsikaterih aplikacij, smo razdelek 4.5 posvetili primerjavi sistema z nekaterimi nevronski modeli.

3.2 Uporabljena orodja

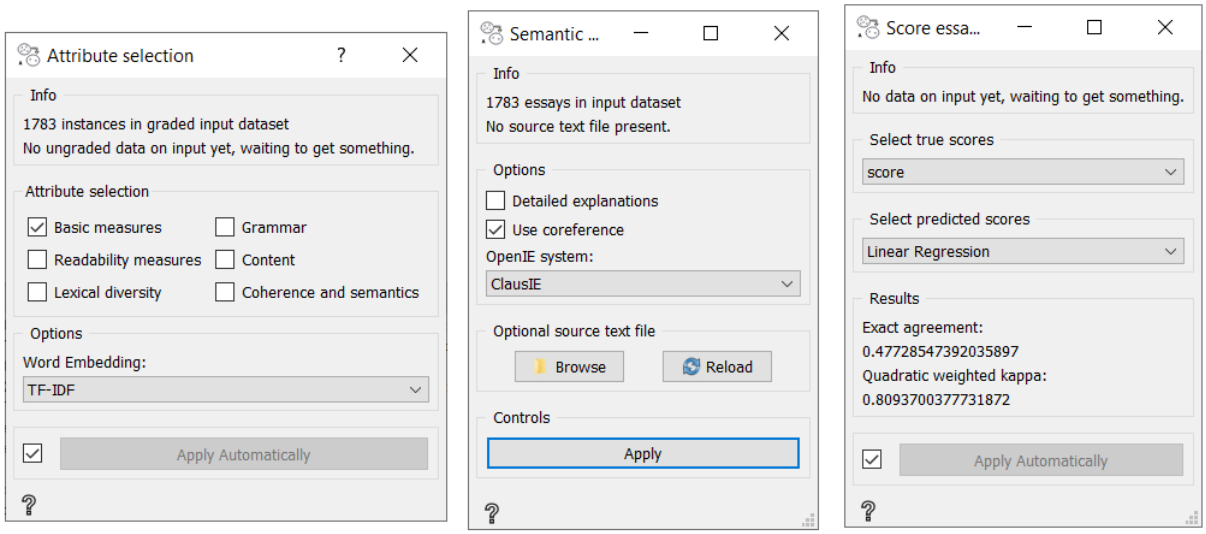
Celoten sistem smo implementirali znotraj orodja za podatkovno rudarjenje Orange[8] v programskem jeziku Python. Glavne uporabljene knjižnice za razčlenitev besedila in izračun atributov so NLTK,[9] SpaCy,[10] scikit-learn[11] in language-check[12] za zaznavanje pravopisnih napak.

Za delo z ontologijami smo uporabili knjižnico rdflib[13] in zunanja sistema (v smislu samostojna lokalna programa) ClausIE (na voljo tudi OpenIE5.0) in HermiT. Za lažji ogled ontologij smo si pomagali z orodjem Protege.[14]

Orange Data Mining je programsko okolje za vizualno in preprosto uporabo metod strojnega učenja. Omogoča nam uporabo različnih metod strojenega učenja brez znanja programiranja ali globokega poznavanja posameznih metod. Namenjen je tako začetnikom kot izkušenim programerjem in poznavalcem s področja, saj omogoča tudi neposreden dostop do kode in tako večje prilagodljivosti modelov.

Orange je osnovan na gradnikih (angl. *widgets*), ki jih vlečemo na risalno površino (angl. *canvas*) in jih med seboj vizualno povezujemo. Vsak gradnik ima definirane določene vhode in izhode, ki so bodisi obvezni ali neobvezni. S klikom na posamezni gradnik se nam odpre okno, kjer nastavimo obvezne nastavitve za delovanje gradnika, obenem pa imamo možnost nasta-

[8] `https://orange.biolab.si/`
[9] `https://www.nltk.org/`
[10] `https://spacy.io/`
[11] `https://scikit-learn.org/stable/`
[12] `https://pypi.org/project/language-check/`
[13] `https://rdflib.readthedocs.io/en/stable/`
[14] `https://protege.stanford.edu/`



Slika 3.4: Prikaz vseh treh gradnikov.

vljanja dodatnih parametrov (katerih vrednosti lahko pustimo tudi privzete). Praviloma so nastavitve v gradnikih jasne in preproste. Ne prikazujejo vseh podrobnosti, ki jih gradnik sicer podpira, ampak uporabniku omogočijo spreminjanje le najnujnejših, najvplivnejših nastavitev.

V tem duhu smo zasnovali tudi naše gradnike. Gradniki ne razkrivajo vseh podrobnosti, ki bi jih lahko, ampak smo jih zasnovali tako, da omogočajo le najnujnejše.

3.3 Gradniki v Orange

Skupno smo razvili tri gradnike, ki zajemajo celoten opisani sistem. Slika 3.4 prikazuje vse tri gradnike, ki so opisani v nadaljevanju.

Prvi gradnik je namenjen izračunu vseh različnih opisanih mer, ki so osnovne (plitke) statistične mere, mere berljivosti, leksikalne mere, slovnične mere, vsebinske mere in mere koherentnosti. Ker je računanje nekaterih naprednih mer zahtevnejše, se lahko uporabnik odloči za izračun kakršne koli kombinacije naštetih šestih skupin mer. Za vsebinske mere in mere koheren-

tnosti je na voljo dodatna izbira metode pretvorbe besedila v večdimenzionalni vektorski prostor. Tu podpiramo dve metodi: statistično pretvorbo TF-IDF in vektorske vložitve GloVe (v dveh izvedbah: SpaCy in Flair).

Gradnik ima tri vhode:

1. vhod za ocenjene eseje,
2. vhod za neocenjene eseje in
3. vhod za izvorno besedilo.

Vhoda za ocenjene in neocenjene eseje sta ločena zaradi nekaterih atributskih funkcij — za izračun nekaterih atributov potrebujemo že ocenjene eseje, npr. za računanje podobnosti z najboljšimi (najbolje ocenjenimi) eseji. V primeru, da že ocenjenih esejev nimamo (imamo recimo samo naučen model in neocenjene eseje), se izračun atributov, ki potrebujejo ocenjene eseje, preskoči.

Vhod za izvorno besedilo je neobvezen. Izvorno besedilo predstavlja osnovno zgodbo, poziv ali idejno zasnovo, na osnovi katere so pisani eseji. V primeru šol je to odlomek iz neke zgodbe (knjige). Če so eseji osnovani na nekem izvornem besedilu, ga povežemo na ustrezen vhod in s tem izračunamo dodatni atribut (podobnost eseja z izvornim besedilom).

Gradnik ima dva izhoda, in sicer izhod za izračunane atribute ocenjenih esejev in izhod za izračunane atribute neocenjenih esejev. Ločitev je pomembna, saj nam to omogoča, da podatke ustrezno nastavimo kot vhod v druge Orangeove gradnike (npr. gradnik za prečno preverjanje *Test and Score* ali gradnik za napovedovanje *Predictions*).

Drugi gradnik obsega delo in iskanje semantičnih neskladnosti z ontologijo. Gradnik je samostojen zaradi velike računske in časovne zahtevnosti. Ima dve nastavitvi: ali želimo uporabiti razreševalnik koreferenc in ali želimo, da se nam za semantične napake vrne podrobna razlaga. Uporaba koreferenc je priporočljiva, saj je v primerih posrednega navezovanja na različne pojme v besedilu to edini način zajetja celotne semantične informacije. Podrobna razlaga ni nujna, saj se nam v primeru napake konsistence besedila vrne že

osnovna razlaga, podrobna razlaga pa služi kot dodatek tej razlagi, vendar je časovno potratna. Izberemo lahko tudi osnovno besedilo, s katerim se razširi ontologija, tako da ta vključuje tudi vsebino osnovnega besedila. To besedilo bo obdelano pred vsem drugim, izluščene trojice pa bodo dodane v ontologijo. Razširjena ontologija bo uporabljena za preverjanje skladnosti esejev. Če osnovnega besedila ne dodamo, za preverjanje skladnosti normalno uporabi osnovna ontologija (v našem primeru ontologija COSMO).

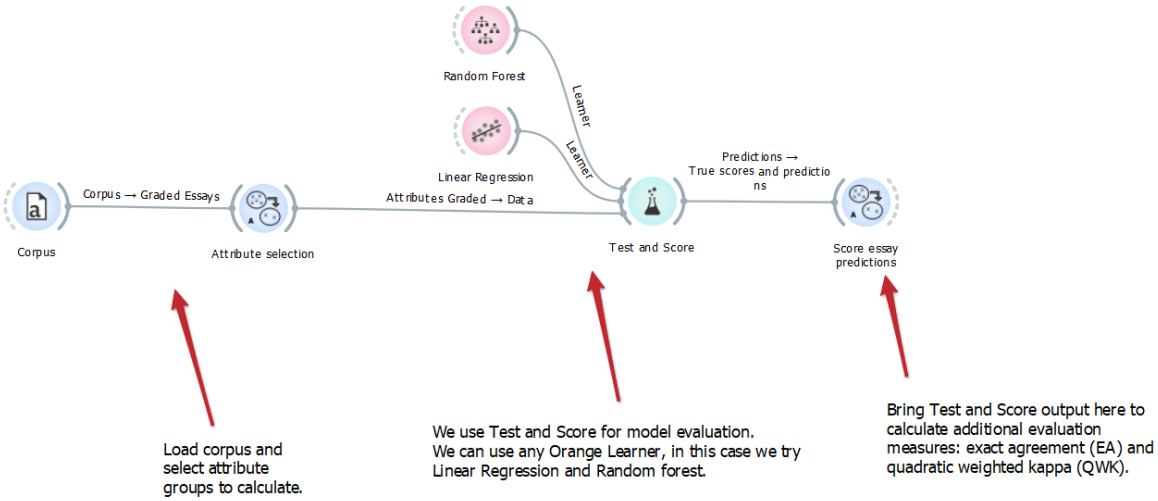
Gradnik ima samo en vhod — vhod za eseje in en izhod — tabela treh atributov o številu posameznih napak in niz z osnovno razlago. V primeru izbranih podrobnih razlag se tabeli doda tudi stolpec s podrobnimi razlagami.

Tretji gradnik je namenjen evalvaciji napovedanih ocen in pravih ocen esejev. Ker Orange ne podpira mer za izračun natančnega strinjanja (angl. *exact agreement*) in kvadratne uteženе kape (angl. *quadratic weighted kappa – QWK*), smo naredili gradnik, ki prejme tabelo z napovedanimi ocenami (lahko več različnih modelov) in pravimi ocenami. Zgledovali smo se po izhodu gradnika *Test and Score* — za zagotavljanje interoperabilnosti lahko ta izhod vežemo neposredno na vhod našega gradnika, kjer se izračunata prej omenjeni meri.

Uporaba gradnika za izbor atributov in evalvacije modela s kvadratno uteženo kapo je prikazana na sliki 3.5. Enak primer najdemo tudi med primeri v Orange-u (s klikom na *Help* in *Example workflows*). Vsak gradnik ima tudi stran za pomoč, kjer so opisani vhodi/izhodi, uporaba, primeri in splošni opis. En tak primer izseka iz pomoči prikazuje slika 3.6.

3.4 Podrobnosti implementacije gradnika za preverjanje semantične skladnosti

Izračun atributov in odkrivanje semantičnih neskladij zahteva kar nekaj operacij. Najprej potrebujemo razreševalnik koreferenc, nato sistem za ekstrakcijo trojic (OpenIE) ter nato še orodje za delo in preverjanje ontologij. Za razreševalnik koreferenc ni bilo težav, saj že programski paket *SpaCy* po-



Slika 3.5: Primer uporabe sistema AGE/AGE+

nuja dobro implementacijo te metode. Prav tako je za delo in upravljanje z ontologijo zadostovala tudi prej omenjena knjižnica *rdflib*. Knjižnica za delovanje sicer uporablja jezik *RDF* in ne *OWL*, v katerem je zapisana ontologija COSMO. Kljub temu knjižnica omogoča branje in zapis ontologij različnih formatov, vključno z *OWL*, tako da tu ni bilo težav.

Za funkcionalnosti ekstrakcije trojic in preverjanja ontologije smo morali uporabiti zunanje sisteme (v smislu drugih samostojnih lokalnih programov). Za ekstrakcijo trojic smo kot glavni sistem uporabili *ClausIE*. V osnovi gre za program, ki ga poganjamo prek ukazne vrstice, zato smo uporabili (nekoliko modificirano) ovojnico za jezik Python[15] in vse skupaj vključili v projekt.

Za preverjanje skladnosti ontologije uporabljamo logični sklepalnik HermiT, ki je tudi na voljo v obliki konzolne aplikacije. Za Python smo spisali ovojnico, ki ontologijo, naloženo s knjižnico *rdflib*, serializira v obliko *OWL*, shrani na disk in zažene HermiT, kjer je vhod ta ontologija. Serializacija je postopek pretvorbe in zapis programskih objektov, ki jih imamo naložene

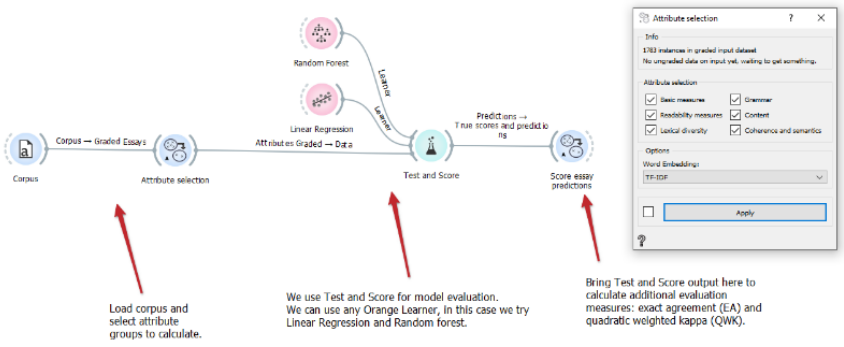
[15] `https://github.com/AnthonyMRios/pyclausie`

- Clark Evans nearest neighbour
- Average distance of nearest neighbour
- Cumulative frequency
- Avg/min/max distance to centroid (2x, euc. and cos. distance)
- Standard distance
- Relative distance
- Determinant of distance matrix
- Moran's I
- Geary's C
- Gettis' G

2. Select word embeddings:
   - Word embeddings are used during calculations of 'Content' and 'Coherence' attributes
   - Choose 'TF-IDF' or 'GloVe' (SpaCy and Flair implementations available) word embeddings
3. The calculation may take a few minutes, depending on attribute categories selected. 'Grammar', 'Content' and 'Coherence' are most demanding.
4. Due to speed, selection changes are NOT communicated automatically. You can change this by ticking the checkbox next to 'Apply' button.

Examples

In the below example we put our training set corpus on the "graded essays" input. We calculate all attributes (using TF-IDF) and do 10-fold cross-validation in "Test and Score" widget. Quadratic weighted kappa is calculated in "Score essay predictions" widget.



If we have a defined train and test set beforehand, we would put the training set on "graded essays" input and the test set on "ungraded essays" input. The difference is in calculations of some attributes, as some require knowledge of best graded essays ('training set') for comparison.

Slika 3.6: Izsek iz pomoči (*help*)

v programu (pomnilniku), v obliko, primerno za zapis in shranjevanje. Po zaključenem pregledovanju HermiT vrne stanje ontologije in morebitne napake. Z nekaj modifikacijami smo vključili tudi možnosti vračanja podrobne razlage napak. Ovojnica morebitne napake ustrezno obravnava, podrobne razlage tudi razčleni in pretvori v bolj naravno berljivo obliko.

V ozadju sta torej oba sistema integrirana prek komunikacije z datotekami (vhod) in sistemskih ukazov za zagon posameznega sistema.

3.5 Algoritem preverjanja semantične skladnosti

Pri postopku preverjanja semantične skladnosti vstopamo v še pretežno neraziskan svet možnosti in tudi pasti. Veliko je parametrov, možnosti in načinov, kako se preverjanja lotiti, kjer pa je treba paziti na časovno in prostorsko zahtevnost problema, če želimo, da bo sistem uporaben na povprečnih domačih računalnikih. V tem sklopu podajamo nekatere podrobnosti algoritma za lažjo predstavo in razmišljanje o idejah za morebitno nadgradnjo.

V grobem je začetek postopka opisan že v prejšnjem poglavju, tako da tukaj omenimo samo glavne stvari: ločitev besedil na povedi, razreševanje koreferenc, ekstrakcijo in predprocesiranje trojic *(osebek, predikat, predmet)*.

Za sledenje dodanim entitetam v ontologijo uporabljamo podatkovno strukturo, ki omogoča hitro dodajanje, iskanje in osnovno filtriranje entitet. Pred začetkom dodajanja trojic v ontologijo iz ontologije preberemo vse trojice in njihove elemente shranimo (v originalni in korenjeni obliki) ter pri tem ločimo dve skupini: osebke/predmete in predikate. Ti dve skupini lahko enačimo s samostalniškimi izrazi (angl. *NP - Noun Phrase*) in glagolskimi izrazi (angl. *VP - Verb Phrase)*, zato iz besedila (eseja) pridobimo vse samostalniške in glagolske izraze ter jih poskusimo združiti z elementi v ontologiji. Ker so posamezne besede kjub korenjenju lahko zelo razgibane in imajo različne predpone, uporabimo nestrogo ujemanje po znakih s pragom

70 %. Ob ujemanju si oba elementa shranimo v podatkovno strukturo. S tem si ustvarimo hitri dostop oz. preslikavo iz entitet v besedilu v elemente, ki so že v ontologiji.

Nato se lotimo dodajanja trojic v ontologijo, kjer je cilj:

1. dodati vsak posamezni element v ontologijo (če še ne obstaja),
2. vsakemu elementu dodati povezave na obstoječe elemente ontologije — sopomenke, protipomenke, nadpomenke — in
3. v ontologijo dodati trojico kot celoto.

Za vsak element trojice izvedemo postopek v nadaljevanju, najprej za osebek in predmet, nato za predikat.

1. Preverimo, ali je element v naši podatkovni strukturi: originalna in korenjena oblika, strogo ujemanje, nato pa še prej omenjeno pragovno ujemanje po znakih.
2. Če je točka 1 neuspešna, za vse sopomenke elementa preverimo, ali so v ontologiji. Če so, v nadaljevanju element nadomestimo s to sopomenko, v nasprotnem primeru pa postopek ponovimo za nadpomenke.
3. Če sta točki 1 in 2 neuspešni, v ontologijo dodamo novi samostojen element (vozlišče). Temu se skušamo izogniti, saj tako vozlišče v grafu z globalnega vidika tvori otok — torej ni povezano z nobenim elementom/konceptom v ontologiji.
4. Na tem mestu imamo element dodan v ontologiji in ga poskušamo čim bolj zvezati z drugimi elementi v ontologiji. Pregledamo njegove nadpomenke in protipomenke ter jih, če tak element v ontologiji obstaja, s tem elementom povežemo.
5. V primeru, da gre za predikat, pogledamo, ali gre za nikalni izraz (vsebuje *not*). Takemu izrazu po najboljših močeh negacijo odstranimo in ga v ontologijo dodamo kot nasprotje originalnemu nikalnemu izrazu.

6. Identifikator (URI) dodanega elementa v ontologiji si zapomnimo.

Ko ta postopek opravimo za vse tri elemente trojice, v ontologijo poskušamo dodati celotno trojico (s pridobljenimi identifikatorji). Za zagotovitev tranzitivnosti relacijo, ki jo izraža trojica, dodamo tudi vsem otrokom/podrejenim vozliščem predmeta, torej dodamo *(osebek, predikat, x)*, kjer so x vsa podrejena vozlišča *predmeta*. Pri preverjanju skladnosti vsa dodajanja za zagotovitev tranzitivnosti obravnavamo skupaj — atomarno. Postopek ponavljamo, dokler ne obravnavamo in ne dodamo vseh trojic.

Po kakršnem koli dodajanju elementov v ontologijo je smiselno preveriti, ali je z dodajanjem nastala kakšna napaka oz. neskladje. V primeru neskladja pri dodajanju posameznih elementov v ontologijo (elementov trojic, povezav, sopomenk itd.) to obravnavamo kot napako prvega tipa (primer, ki mu ni mogoče zadostiti), v primeru napake pri dodajanju celotne trojice pa napako drugega tipa (nekonsistentna ontologija — semantično neskladje). Po dodani zadnji trojici oba tipa napak seštejemo in s tem dobimo zadnje tri atribute, ki se osredotočajo na semantično skladnost besedila.

Ker preverjanje ontologije potrebuje nekaj časa, poskušamo število preverjanj čim bolj zmanjšati. Namesto, da ontologijo preverjamo po vsakem dodajanju elementa, jo pregledamo samo po dodajanju celotne trojice. V večini primerov ontologija ne bo imela napak in lahko normalno nadaljujemo, s čimer smo se izognili vsaj nekaj preverjanjem. Če pride pri preverjanju po dodani trojici do napake, moramo stanje povrniti na trenutek, ko je bila (uspešno) dodana prejšnja trojica. Podatkovna struktura za ta namen omogoča shranjevanje in povrnitev stanja. Ko stanje povrnemo na ustrezen trenutek, ponovimo postopek dodajanja elementov trojice, le da tokrat preverjamo ontologijo pri vsakem dogajanju. Tak varen, a počasen način dodajanja obdržimo, dokler ponovno ne dodamo trojice, pri kateri je bilo prvotno preverjanje neuspešno. Rezultat te optimizacije je, da pri stavkih in povedih, kjer ne bi prišlo do napak, po nepotrebnem ne preverjamo vsakega dodajanja v ontologijo in tako prihranimo čas. V primeru, da bi pri vseh trojicah prihajalo do napak, je ta način sicer počasnejši, vendar je to

ekstrem, ki ne predstavlja realnih primerov. Paziti moramo tudi na pravilno štetje napak, da slučajno kakšne napake zaradi vrnitve na starejše stanje ne štejemo večkrat.

Za dodatno pohitritev se uporabimo nitenje (oz. v programskem jeziku Python večprocesiranje) s knjižnico *multiprocessing*.

Poleg preverjanja ontologij moramo tudi poskrbeti za razlago neskladij. Za osnovno razlago zadostuje beleženje trojic, ki po dodajanju v ontologijo povzročijo neskladnost. Za boljšo berljivost trojice pred izpisom uporabniku preslikamo v poved/stavek, iz katerega ta trojica izhaja. To skušamo nadgraditi tako, da v ontologiji pregledamo vse nasprotne relacije (predikate) trojice in v primeru, da kakšnega najdemo, tudi to trojico (oz. ustrezno poved) izpišemo. Če ustreznih nasprotnih relacij ne najdemo, izpišemo samo poved, ki je povzročila neskladje.

Logični sklepalnik HermiT omogoča tudi razlago oz. izpis poti, po kateri je odkril neskladje. To zmožnost lahko preprosto testiramo v orodju za pregledovanje in urejanje ontologij Protege,[16] kjer je HermiT vključen v obliki vtičnika. HermiT, ki ga uporabljamo v tem delu, je zasnovan kot aplikacija za uporabo prek ukazne vrstice in funkcionalnosti za izpis razlage privzeto ne podpira oz. nam je neposredno prek zastavic/stikal ne izpostavlja. Napisan je v programskem jeziku Java, na voljo pa je tudi izvorna koda, ki smo jo dopolnili s funkcionalnostjo vračanja razlag. Uporabniku smo zato ponudili možnost, da pri pregledovanju ontologij HermiT vrača tudi pot morebitnih neskladij. To nato izpišemo in imamo s tem zagotovljeno razlago neskladij. Edina slabost tega načina je nekoliko počasnejše izvajanje sklepalnika HermiT. Počasnost sčasoma postaja izrazitejša, saj se ontologija nenehno povečuje (z dodajanjem različnih elementov) in postaja iskanje po grafu časovno zahtevnejše in časovno potratnejše.

[16] `https://protege.stanford.edu/`

3.6 Težave

V tem sklopu opišemo nekatere vidnejše težave, ki so bile med razvojem zelo izrazite.

3.6.1 Komunikacija med sistemi

V našem delu uporabljamo kar nekaj različnih tehnik, metod in sistemov za obdelovanje besedila. Večina metod je že vključenih v Python knjižnice, ki so *de-facto* standardne pri obdelavi besedil. Nekateri sistemi pa v te knjižnice niso vključeni in za njih tudi ni na voljo posebnih namenskih knjižnic, ki bi pokrivale te funkcionalnosti, redko so kvečjemu na voljo ovojnice za različne programske jezike (angl. *wrapper*).

Konkretno v primerih ekstrakcije trojic (sistem ClausIE) in iskanju napak v ontologiji z logičnim sklepalnikom (HermiT) je bila potrebna izdelava lastne ovojnice v Python-u oz. popravek ene od obstoječih. Oba sistema sta na voljo v obliki *konzolne aplikacije* oz. programa, ki ga uporabljamo preko ukazne vrstice, kot vhod zahtevata datoteko, na enak način zapišeta tudi svoj izhod.

Zagon in uporaba sistemov potekata torej s pomočjo sistemskih ukazov za zagon programov preko ukazne vrstice, vhodi in izhodi v sisteme pa so realizirani v obliki datotek. Tej komunikaciji prek datotek bi se raje poskušali izogniti, saj lahko pri takšnem načinu komunikacije hitro pride do napak (napačna absolutna pot, napaka pri razčlenjevanju datotek, napačen format vhodnih datotek itd.).

Omenili bi še storjen popravek, ki orisuje, kako hitro lahko pri datotekah nastane težava. Za sistem ClausIE smo poskusili uporabiti obstoječo ovojnico. Ta ovojnica nam ni delovala — pri uporabi je takoj prišlo do napak v zvezi z datotekami. Ugotovili smo, da avtor pri branju in pisanju datotek teh ni ročno zaprl. Na nekaterih operacijskih sistemih je tak način kjub tej napaki očitno deloval. Ko smo to napako odpravili (in sproti posodobili uporabo nekaterih funkcij na novejšo različico Pythona), je ovojnica delovala brezhibno. Podobni spregledi in napake se lahko zgodijo kjer koli, opazimo

pa jih šele, ko testiramo na drugem operacijskem sistemu ali na drugi konfiguraciji sistema.

3.6.2 Povratna informacija o napakah

Pridobivanje informacije o semantičnih neskladjih je sprva predstavljalo veliko težavo, saj izvedba sklepalnika HermiT, ki smo ga uporabljali, ni podpirala vračanja podrobnosti o neskladjih oz. vračanja poti, ki pripeljejo do logičnih nasprotij. Tako smo v začetni fazi razvoja podpirali samo možnost izpisa, pri dodajanju katere trojice se zgodi napaka. To smo kasneje nadgradili na izpisovanje originalnih povedi, pri katerih pride do napake, nato pa še z ročnim iskanjem nasprotnih relacij po vseh podpomenkah in nadpomenkah predmeta trojice. V primeru zadetka se povratna informacija dopolni z nasprotujočo povedjo. Tukaj naletimo na manjšo težavo beleženja nasprotujočih trojic/povedi. Vemo namreč, pri dodajanju katere trojice je prišlo do napake in za to trojico ni težko najti originalne povedi. Ko (in če) z ročnim preverjanjem nasprotnih relacij najdemo trojico v ontologiji, ki predstavlja nasprotje tej povedi, se lahko opremo le na močno obdelano trojico v ontologiji iz česar težko pridemo do originalne povedi.

Situacija se še nekoliko bolj zaplete, če uporabljamo izvorno besedilo, s katerim smo ontologijo napolnili pred preverjanjem esejev. Kako bomo vedeli, da nasprotna trojica v ontologiji, ki smo jo našli ročno, izhaja iz izvornega besedila ali eseja, ki ga trenutno preverjamo? Kako bomo prišli do originalne povedi?

Težavo smo rešili z morda nekonvencionalno uporabo lastnosti *rdfs:comment*. Ta nam omogoča, da vsaki entiteti (elementu) v ontologiji dodamo enega ali več komentarjev, ki lahko vsebujejo kakršno koli besedilo. Namen te lastnosti je sicer opis entitete, kaj predstavlja in čemu služi, vendar nam pri reševanju te težave pride zelo prav.

Pri vsakem dodajanju kakršne koli entitete v ontologijo dodamo tej entiteti še komentar (*rdfs:comment*), ki vsebuje indeks (zaporedno številko) povedi, iz katere ta entiteta izhaja. Posamezni entiteti se sčasoma dodajo

različni indeksi (nastopa v več različnih povedih). Pri entitetah, ki izhajajo iz morebitnega izvornega besedila, indekse predznačimo z minusom. Paziti je treba tudi na morebitno brisanje indeksa iz entitete, če je treba neko relacijo s to entiteto izbrisati, entitete pa ne. Ko pri ročnem iskanju najdemo nasprotno trojico, pogledamo presek indeksov vseh treh elementov trojice. Praviloma bi moral v preseku biti samo en indeks, ki nam pove, iz katere povedi je ta relacija prišla. S tem smo rešili težavo, opisano zgoraj. Negativni indeksi, kot omenjeno, predstavljajo relacijo, ki smo jo dobili iz izvornega besedila. Absolutna vrednost negativnega indeksa nam pove dejanski indeks povedi.

Zgodi se lahko tudi primer, ko nasprotje ni tako neposredno in lahko ugotovljivo. V programu Protege je že vgrajen vtičnik za sklepalnik HermiT, ki omogoča sklepanje in ugotavljanje logičnih neskladij ter vračanje poti sklepanja ob zaznani napaki. To je osrednja funkcionalnost sklepalnika HermiT, zato je še posebej zanimivo, da namizna različica programa tega ne omogoča. HermiT je odprtokodni program, kar nam je omogočilo vpogled v kodo, ki je razkril, da HermiT to pričakovano sicer omogoča, vendar ta funkcionalnost uporabnikom (torej nam) ni izpostavljena. Da smo to funkcionalnost omogočili, smo v kodi dodali uporabo teh metod in funkcionalnost, skladno z ostalimi, izpostavili prek stikala. To nam je potem omogočilo beleženje poti, ki pri sklepanju pripelje do logične napake.

Vseeno pa ta metoda močno upočasni delovanje sistema pri daljših besedilih, ko se ontologija zelo razširi. Tu je tudi ena od težav bolj estetske narave — HermiT nam pot sklepanja ob napaki vrne v svojem formatu, kjer uporablja nazive svojih entitet, ki so v našem primeru močno sprocesirani deli besedila in morda ne odražajo več smiselne povedi ali stavka (dobili bi npr. zelo preprosto nasprotje *Lisa likes Tennis* in *Lisa doesNotLike Tennis*).

Zavedati se je tudi treba, da tak način ne bo vedno odkril vseh logičnih oz. semantičnih napak, saj so povedi lahko zelo sintaktično (in logično) kompleksne. Tukaj je veliko odvisno od natančnosti ostalih uporabljenih orodij (razreševanja koreferenc, ekstrakcije trojic itd.) in neoporečnosti besedila v

smislu slovničnih napak.

3.6.3 Počasnost odkrivanja semantičnih napak

Odkrivanje semantičnih napak po opisanem postopku je dolgotrajno zaradi pogostega preverjanja ontologije z logičnim sklepalnikom HermiT. Posamezno sklepanje na začetku traja okoli deset sekund, ob polnejši ontologiji lahko bistveno dlje, odvisno tudi od kompleksnostih dodanih relacij in morebitne izbrane možnosti za podrobno povratno informacijo.

Na prejšnjih straneh smo že opisali implementirane optimizacije, kot so hitro dodajanje in povrnitev stanja ob napaki ter večnitnost. Kljub tem pohitritvam ostaja ozko grlo sistema v časovno potratnem sklepanju sklepalnika.

V zvezi s tem nam je nekaj težav povzročala tudi knjižnica za večnitnost oz. večprocesiranje. Iz te knjižnice uporabljamo priročen konstrukt *Pool*, ki omogoča samodejno ustvarjanje potrebnega števila procesov in sprotno uravnoteženo delitev nalog mednje. Pri velikih podatkovnih zbirkah (med tisoč in dva tisoč primeri oz. eseji), kjer vsakemu procesu sproti dodeljujemo po en esej na enkrat, se delovanje vseh procesov po nekaj desetih opravljenih nalogah močno upočasni. Razlogov za to je lahko kar nekaj, vendar nismo odkrili nobenega, ki bi ustrezal našemu primeru. Sklepamo, da gre za nekakšno optimizacijo operacijskega sistema, ki čez čas delovanje teh procesov upočasni. Predpostavljamo, da zaradi ohranjanja energije, saj gre v tem primeru za prenosni računalnik. Težavo smo rešili z nekoliko grobim (*batch*) pristopom razporeditve esejev (nalog) v manjše skupine (nekaj deset) oz. pakete in ponovnim zagonom (prekinitvijo in ustvarjanjem novih) procesov ob opravljenem paketu nalog. Ko se paket nalog dokončno izvede in konča, se vsi procesi, ki so ta paket obdelovali, prekinejo in ustvarijo novi. Novi procesi prejmejo naslednji paket. Postopek ponavljamo, dokler ne obdelamo vseh paketov. S tem pristopom pri vsaki obdelavi novega procesa naredimo "reset" in obdelavo nadaljujemo s prvotno hitrostjo.

Dotaknili bi se tudi hitrosti predprocesiranja esejev. Pri tem mislimo

predvsem na razreševanje koreferenc, iskanje trojic in pripravo ontologije. Tudi ti postopki ob podatkovnih zbirkah takšne velikosti trajajo nekaj časa, vendar je ta čas proti časovni zahtevnosti preverjanja ontologij zanemarljiv.

3.6.4 Dolgotrajno testiranje sistema SAGE

Testiranje delovanja in pridobivanje rezultatov sistemov AGE in AGE+ sta bili relativno preprosti. Nasprotno velja za sistem SAGE, ki poleg bolj ali manj hitro izračunljivih atributov sistema AGE+ potrebuje tudi tri atribute, ki jih pridobimo s pomočjo sistema za odkrivanje semantičnih napak. Kot že nekajkrat omenjeno, je ta sistem zelo počasen, kar nam je otežilo obsežno testiranje in pridobivanje rezultatov. Pred in v času pisanja tega dela je računalnik nenehno obdeloval podatke in preverjal semantično skladnost esejev, bodisi za namen testiranja bodisi za pridobivanje rezultatov. Čas testiranja je zelo odvisen od zmogljivosti računalnika in dolžine esejev v podatkovni zbirki. V povprečju je testiranje ene zbirke trajalo nekoliko dlje kot dva tedna, testirali pa smo na štirih podatkovnih zbirkah. Največjo (časovno) težavo so tu predstavljale nepričakovane tehnične in vsebinske napake ali pomanjkljivosti, ki smo jih morali naknadno popraviti in začeti testirati od začetka. Govorimo o splošnih napakah v kodi, nepredvidenih napakah pri delovanju zunanjih sistemov (npr. ClausIE), pomanjkljivem ali preveč agresivnem predprocesiranju elementov trojic (izguba informacije) itd. Najbolj pa je na celoten postopek vplival predhodno opisana težava z večprocesiranjem, kjer hitrost posameznih procesov sčasoma pada.

Tudi ob manjših podatkovnih zbirkah, namenjenih hitremu testiranju sistema, je trajalo kar nekaj časa, preden je sistem končal in vrnil rezultat. Manjše zbirke tudi niso predvidele (zajele) vseh potencialnih napak, do katerih lahko in je prišlo med obdelavo celotne podatkovne zbirke.

Poglavje 4

Rezultati

V tem poglavju predstavimo uporabljene podatke, metodologijo in rezultate našega sistema. Poglavje je v celoti namenjeno analizi rezultatov in sistema v smislu pravilnosti delovanja (natančnost napovedovanja ocen esejev), saj smo glavne performančne pomisleke že poudarili v poglavju o implementaciji oz. razdelku o težavah (npr. razdelek 3.6.3). Rezultate sistema podrobno analiziramo, kolikor je mogoče, primerjamo s sistemom Zupanc in jih poskušamo izboljšati z ustreznim izbiranjem atributov, kjer preizkusimo nekaj različnih metod izbiranja. Na koncu poglavja naš sistem in pridobljene rezultate primerjamo še z nevronskimi modeli.

4.1 Podatki

Eseje smo pridobili s spletne strani Kaggle,[1] ki je namenjena vsem v zvezi s strojnim učenjem in podatkovno znanostjo. Na njej lahko delimo različne podatkovne zbirke, se podučimo o novostih in metodah podatkovne znanosti ter sodelujemo v tekmovanjih in diskusijah.

Pred osmimi leti je Kaggle gostil tekmovanje o samodejnem ocenjevanju esejev (angl. *AES – Automated Essay Scoring*)[2]. Avtorji tekmovanja so

[1] `https://www.kaggle.com/`

[2] `https://www.kaggle.com/c/asap-aes`

pripravili ustrezno označene podatkovne zbirke (učne, validacijske in testne množice), naloga tekmovalcev pa je bila razvoj modela za čim bolj natančno oceno neocenjenih esejev glede na (učno) množico ocenjenih esejev. Za testiranje in pridobivanje rezultatov smo uporabili podatkovne zbirke, ki so na voljo v sklopu tega tekmovanja. Enake podatkovne zbirke so uporabljene tudi v članku Zupanc in Bosnić [1] ter v disertaciji Zupanc [2].

Testiranje modelov je potekalo tako, da so tekmovalci naložili svoje rezultate na spletno stran, kjer je nato strežnik ocenil natančnost rezultatov, ne da bi razkril ocene esejev v testni množici. Tako rezultati testne množice podatkov niso na voljo (ocene esejev v testni množici niso na voljo). Ugotavljanje natančnosti na validacijski množici je sledilo istemu postopku, zato tudi ocene esejev v validacijski množici niso na voljo. Tekmovanje je že nekaj let končano in nove oddaje niso omogočene.

Tekmovanje je obsegalo osem različnih podatkovnih zbirk, kjer ima vsaka različno temo oz. področje pisanja. Vsaka zbirka definira tudi svoj nabor mogočih ocen esejev. Nekatere imajo razpon ocen relativno nizek (npr. od ena do šest), nekatere pa kar velik (npr. od nič do trideset). V splošnem sta za vsak esej oceno podala dva ocenjevalca. Ta ocena se je nato povprečila v končno oceno (dostop imamo do vseh teh podatkov). Pri nekaterih zbirkah se je poleg celotne ocene eseja ocenjevalo tudi nekatere druge, bolj določene stvari. Vsaka zbirka vsebuje okoli dva tisoč esejev.

Druga zbirka je ocenjena dvakrat (po dveh različnih merilih), zato je ta zbirka razdeljena na dve: *2A* in *2B*. Zbirke tri, štiri, pet in šest vsebujejo eseje, ki so pisani na osnovi izvorne zgodbe oz. izvornega besedila. Ker je pomembna semantična skladnost teh esejev z dejstvi, ki so zapisana v izvornem besedilu, smo te eseje testirali s sistemom za preverjanje semantične skladnosti.

Podatki oz. posamezni eseji so bili ročno prepisani iz resničnih esejev na papirju v digitalno obliko, zato lahko pri pregledovanju podatkov zasledimo kar nekaj slovničnih ali tipkarskih napak zaradi nečitljivega besedila, kar otežuje natančno gradnjo in učenje modelov.

4.2 Metodologija

Za vse eseje smo izračunali opisane atribute s sistemoma AGE in AGE+. To je zajemalo osnovne statistične mere, leksikalno raznolikost, mere berljivosti, slovnico, vsebinsko podobnost in v primeru AGE+ tudi koherenco besedil.

Za vse zbirke so bili izračunani atributi obeh sistemov v vseh kombinacijah vektorskih predstavitev besed: TF-IDF, GloVe (implementacija SpaCy) in GloVe (implementacija Flair). Za zbirke tri, štiri, pet in šest smo izračunali tudi dodatne atribute, potrebne za sistem SAGE (semantična skladnost).

Rezultate smo merili na osnovi dveh metrik: natančnega ujemanja (angl. *exact agreement*) in kvadratno utežene kape (angl. *quadratic weighted kappa – QWK*).

Popolno ujemanje je definirano kot odstotek esejev, ki jih je napovedni model ocenil enako kot ocenjevalci. Ta mera je zelo stroga, saj je pri nekaterih zbirkah razpon ocen zelo velik, kar pomeni, da še tako majhno odstopanje napovedi od pravilne ocene ta mera označi za napačno napoved in je primerjava med posameznimi zbirkami z različnimi razponi ocen neuravnotežena.

Kvadratno utežena kapa [1] oz. kvadratno utežena *Cohenova kapa* je metrika, ki meri stopnjo strinjanja/soglasja med dvema ocenjevalcema. V našem primeru med seboj primerjamo ocenjevalca in napovedane ocene (računalnik). Tipično ima mera razpon od 0 do 1, kar pomeni od naključnega strinjanja do povsem enakega strinjanja. Redko je lahko tudi manjša od 0, kar predstavlja manjše strinjanje, kot je pričakovano po naključju.

Mera se izračuna kot:

$$\kappa = 1 - \frac{\sum_{i,j} w_{i,j} O_{i,j}}{\sum_{i,j} w_{i,j} E_{i,j}} \tag{4.1}$$

kjer ima nabor esejev E različne mogoče ocene $1, 2, 3, ..., S$, w so uteži, O je posebna matrika napovedanih ocen in E je posebna matrika pričakovanih, pravilnih ocen. Matrika uteži w_{ij} velikosti SxS se izračuna

glede na razliko med ocenami ocenjevalcev:

$$w_{i,j} = \frac{(i-j)^2}{(S-1)^2} \tag{4.2}$$

Matrika napovedanih ocen O je histogram velikosti SxS, ki ga zgradimo tako, da element $O_{i,j}$ predstavlja število esejev, ki so prejeli oceno i od prvega ocenjevalca (A) in oceno j od drugega ocenjevalca (B). Enako velja za matriko E, ki jo zgradimo na enak način:

$$E_{i,j} = \frac{H_{Ai} \cdot H_{Bj}}{N} \tag{4.3}$$

kjer $H_{Ai}, i = 1, ..., S$ predstavlja število esejev, ki jih je prvi ocenjevalec (A) ocenil z oceno i, podobno za drugega ocenjevalca B z oceno j, N je število ocen oz. esejev. Matrika E je normalizirana z N, tako da imata matriki E in O enako vsoto.

Pri pridobivanju rezultatov smo natančnost modelov izračunali z obema metrikama, vendar smo pri primerjavi modelov upoštevali zgolj kvadratno uteženo kapo. Da bi lahko modele ob manjkajočih ocenah esejev v testni množici primerjali med seboj in se hkrati izognili pretiranemu prilagajanju podatkom (angl. *overfitting*), smo uporabili 10-kratno prečno preverjanje (angl. *10-fold cross validation*).

Za napovedovanje ocen smo na začetku poskušali s tremi različnimi modeli: linearno regresijo, naključnimi gozdovi in nevronskimi mrežami. Uporabili smo že implementirane modele oz. gradnike v Orange-u.

Najboljši rezultat so konsistentno dosegali naključni gozdovi, nekoliko slabša je bila linearna regresija. Nevronske mreže so dajale precej slabše rezultate. Sklepamo, da je to posledica velikega števila parametrov, ki jih lahko (in velikokrat moramo) pri nevronskih mrežah ustrezno nastaviti, v kombinaciji s tem, da so nevronske mreže v Orangeu povsem navadne, osnovne. V tem kontekstu bi bilo bolje poskusiti z naprednejšimi konstrukti, kot so mreže RNN (angl. *Recurrent Neural Networks*) ali LSTM (angl. *Long short-term*

memory). Kasneje v razdelku 4.5 primerjamo nekatere nevronske modele z našim sistemom.

Za nadaljnje poglobljeno testiranje smo zato uporabili naključne gozdove in linearno regresijo. Linearno regresijo bi se nam zdelo smiselno testirati v vsakem primeru, saj je računsko nezahtevna in na splošno preprosta za uporabo. S spreminjanjem regularizacijskih parametrov pri linearni regresiji nam je uspelo rezultat nekoliko izboljšati od naključnih gozdov.

Končne vrednosti parametrov obeh modelov so bile:

- naključni gozdovi: 100 gozdov, najmanjše število elementov v listu 5, ponovljivo učenje (naključno stanje vedno enako), druge nastavitve privzete;
- linearna regresija: regularizacija L2 (oz. *ridge regression*) z vrednostjo $\alpha = 0,02$.

Vrednosti parametrov smo pridobili z ročnim testiranjem različnih parametrov na pretežno prvi podatkovni zbirki (*DS1*). Tako kot Zupanc, smo pri naključnih gozdovih uporabili 100 dreves. Dodatno smo dodali mejo najmanjšega števila elementov v listu in uporabili ponovljivo učenje, da smo si v sklopu analize zagotovili konsistentne rezultate. Linearni regresiji smo tudi dodali regularizacijo, ki v dosedanjem delu [1, 2] ni bila uporabljena.

4.3 Analiza sistemov AGE in AGE+

Začeli bomo z analizo in primerjavo sistemov AGE in AGE+.

V tabelah 4.1 in 4.2 so prikazani rezultati za oba sistema v vseh kombinacijah predstavitev besed TF-IDF in GloVe (SpaCy) ter primerjava modela linearne regresije in naključnih gozdov. Za vsak primer sta izračunani metriki popolnega ujemanja in kvadratno utežene kape.

Tako v primeru sistema AGE in sistema AGE+ vidimo, da je najboljši rezultat dosegala linearna regresija s predstavitvijo besed TF-IDF. Pri sistemu AGE nato sledita linearna regresija z vložitvami GloVe (Flair) in naključni

Tabela 4.1: Rezultati sistema AGE

AGE								
	TF-IDF				GloVe			
	RF		LR		RF		LR	
	Exact	QWK	Exact	QWK	Exact	QWK	Exact	QWK
DS1	0,5020	0,8401	0,4919	0,8358	0,5076	**0,8437**	0,4896	0,8347
DS2a	0,7089	**0,7148**	0,6900	0,7001	0,6989	0,7014	0,6894	0,6879
DS2b	0,6756	0,6453	0,7011	**0,6789**	0,6728	0,6359	0,6917	0,6677
DS3	0,6477	0,6441	0,6420	**0,6578**	0,6530	0,6527	0,6547	0,6398
DS4	0,6416	0,7438	0,6445	**0,7536**	0,6253	0,7106	0,6236	0,7158
DS5	0,6670	**0,7994**	0,6726	0,7964	0,6615	0,7919	0,6587	0,7875
DS6	0,6267	0,7531	0,6450	**0,7734**	0,6233	0,7455	0,6289	0,7439
DS7	0,1421	0,7898	0,1542	**0,8071**	0,1485	0,7937	0,1651	0,7998
DS8	0,1037	0,7009	0,1093	**0,7479**	0,1010	0,7050	0,1231	0,7474
AVG	0,5239	0,7368	0,5278	**0,7501**	0,5213	0,7312	0,5250	0,7360

Tabela 4.2: Rezultati sistema AGE+

AGE+								
	TF-IDF				GloVe			
	RF		LR		RF		LR	
	Exact	QWK	Exact	QWK	Exact	QWK	Exact	QWK
DS1	0,5048	0,8346	0,4902	0,8343	0,5154	**0,8413**	0,4874	0,8343
DS2a	0,6983	0,6989	0,6939	0,7073	0,7094	**0,7093**	0,6839	0,6823
DS2b	0,6694	0,6382	0,6939	0,6676	0,6633	0,6271	0,6994	**0,6779**
DS3	0,6547	0,6529	0,6651	**0,6622**	0,6518	0,6480	0,6547	0,6450
DS4	0,6439	0,7448	0,6507	**0,7547**	0,6196	0,7080	0,6185	0,7148
DS5	0,6587	0,7943	0,6665	**0,7955**	0,6504	0,7843	0,6576	0,7887
DS6	0,6300	0,7479	0,6400	**0,7675**	0,6156	0,7400	0,6339	0,7405
DS7	0,1587	0,7867	0,1466	**0,8034**	0,1600	0,7906	0,1600	0,8013
DS8	0,1093	0,6976	0,1120	**0,7428**	0,1024	0,6952	0,1148	0,6491
AVG	0,5253	0,7329	0,5288	**0,7484**	0,5209	0,7271	0,5234	0,7260

gozdovi s TF-IDF, pri AGE+ pa sledijo naključni gozdovi s TF-IDF in nato naključni gozdovi z vložitvami GloVe (SpaCy).

Če primerjamo najboljša rezultata pri obeh sistemih, opazimo, da ima sistem AGE nekoliko boljši rezultat od sistema AGE+. To lahko pripišemo velikemu številu atributov (preko sto) in načinu evalvacije modelov, saj smo uporabili 10-kratno prečno preverjanje in si s tem zmanjšali učno množico, obenem pa vzeli majhen vzorec za testno množico. Izkazalo se je, da pri nekaterih iteracijah določeni atributi zelo pripomorejo k rezultatu, medtem ko pri drugih iteracijah isti atributi k rezultatu ne pripomorejo oz. ga celo kvarijo.

Zupanc je v svojem delu [2] najprej z 10-kratnim prečnim preverjanjem sistema AGE z različnimi modeli ocenila najboljši model in s tem modelom nadaljevala analizo. V njenem primeru so se najbolje odrezali ekstremno naključni gozdovi z rezultatom 0,7365, vendar je testiranje nadaljevala z navadnimi naključnimi gozdovi z rezultatom 0,7356, saj sta si rezultata blizu. S pogledom na obe tabeli (2.1, 2.2) opazimo, da smo v obeh primeri (AGE in AGE+) oba rezultata nekoliko presegli. Sistema AGE+ in SAGE je Zupanc uporabila na dejanskih testnih podatkih tekmovanja, kjer so rezultati (v članku in po njenih besedah) veliko boljši kot s prečnim preverjanjem. Pri večini podatkovnih zbirk v njenih delih je viden boljši rezultat sistema AGE+ in SAGE, kar je tudi posledica uporabe vnaprejšnjega izbiranja atributov na takrat dostopni validacijski in testni množici podatkov.

4.3.1 Izbiranje atributov

Ker je atributov veliko, smo poskusili rezultat izboljšati z nekaj tehnikami izbora najustreznejših atributov.

Prvi poskus je izbor atributov neposredno v Orangeu, saj ta že implementira gradnik, ki atribute razvršča po pomembnosti glede na količino informacije, ki je nosijo. Nad našim naborom atributov smo lahko uporabili dve metodi: enociljno regresijo (angl. *univariate regression*) in *RRefiefF*. Poskušali smo z različnim metrikami (najboljših 50 atributov, atributi z re-

zultatom, višjim od vnaprej izbrane vrednosti itd.), vendar nam s tem ni uspelo izboljšati rezultatov.

Na osnovi našega modela linearne regresije (regularizacija L2 s parametrom 0,02) smo sestavili lestvico z rangi najpomembnejših atributov. Lestvica je bila sestavljena z razvrščanjem (rangiranjem) atributov pri vsaki iteraciji prečnega preverjanja (angl. *fold*), povprečeno čez vse iteracije in nato še čez vse podatkovne zbirke. Rangi najboljših 50-ih atributov so prikazani v tabeli 4.3.

Izbor atributov z omenjenimi metodami ni dajal boljših rezultatov, zato smo poskusili z uporabo tehnike vnaprejšnjega izbiranja atributov (angl. *forward attribute selection*). Ideja te tehnike je, da začnemo s praznim naborom atributov in skozi iteracije v nabor dodamo atribute, ki najbolj povečajo rezultat. V vsaki iteraciji za vse atribute preverimo, za koliko se rezultat modela izboljša in v naboru atributov obdržimo tistega, ki rezultat poveča za največ. Ponavljamo, dokler se rezultat ne izboljša več, ali pa smo dodali vse atribute.

Ker korist dodatnih atributov, ki jih predvideva sistem AGE+, iz tabele ni očitna, smo poskusili narediti test. Za vsako iteracijo prečnega preverjanja (angl. *fold*) smo po zgoraj opisanem postopku izbrali atribute, s katerimi smo dobili najboljši rezultat in rezultat na koncu povprečili. S tem smo sicer izbor atributov preveč prilagodili podatkom oz. podatkom v posamezni iteraciji. Pričakovali pa bi, da je rezultat sistema AGE+ v tem primeru vsekakor boljši, saj imamo na voljo več atributov, torej večjo količino informacij.

Iz tabele 4.4 vidimo, da je rezultat sistema AGE+ s takšnim izborom atributov res boljši. Zanimivo je tudi, da pri uporabi linearne regresije predstavitev besedil s tehniko TF-IDF deluje v obeh primerih veliko bolje kot z vložitvami GloVe. Linearna regresija dosega nekoliko boljše rezultate kot naključna drevesa. Pri obeh modelih je opazna razlika med sistemoma AGE in AGE+, predvsem pri naključnih gozdovih. S tem smo ugotovili, da sistem AGE+ nosi več (uporabnih) informacij kot AGE, vendar je potreben pravilen izbor atributov. V nadaljevanju smo poskusili s podobnimi pristopi.

Tabela 4.3: Najboljših 50 atributov in njihovi rangi na osnovi našega modela linearne regresije, povprečene čez vse iteracije prečnega preverjanja.

Atribut	Povprečen rang
1. Povprečna razdalja do centroida (euc)	6,93
2. Povprečna razdalja med vsemi točkami (cos)	10,56
3. Število znakov	12,13
4. Število besed	13,31
5. Indeks razdalje med sosednjimi točkami (min/max) (euc)	13,42
6. Povprečna razdalja med vsemi točkami (euc)	14,34
7. Standardna razdalja	14,57
8. Maksimalna razdalja do centroida (euc)	15,86
9. minimalna razdalja med sosednjimi točkami (euc)	15,97
10. Število znakov (brez presledkov)	17,08
11. LIX	17,71
12. Maksimalna razdalja med sosednjimi točkami (euc)	18,03
13. Minimalna razdalja do centroida (euc)	18,17
14. Povprečna dolžina povedi	20,43
15. Guiraud's Index	21,53
16. Povprečna razdalja med sosednjimi točkami (cos)	22,5
17. Povprečna razdalja do centroida (cos)	22,61
18. Maksimalna radzalja med vsemi točkami (cos)	22,68
19. Maksimalna razdalja med sosednjimi točkami (cos)	22,76
20. Maksimalna razdalja med vsemi točkami (euc)	22,87
21. Število različnih besed	24,0
22. Povprečna razdalja med sosednjimi točkami (euc)	24,03
23. Maksimalna razdalja do centroida (cos)	25,0
24. Indeks razdalje sosednih točk (min/max) (cos)	25,07
25. Indeks razdalje centroidov (min/max) (euc)	25,81
26. Minimalna razdalja do centroida (cos)	26,59
27. Moran's I	26,76
28. Minimalna razdalja med sosednjimi točkami (cos)	28,1
29. Type-token ratio	28,64
30. Indeks razdalje do centroidov (min/max) (cos)	30,03
31. Relative distance	30,13
32. Automated Readability Index	30,73
33. Flesch-Kincaid Grade Level	32,5
34. Število dolgih besed	34,93
35. Korelacijske vrednosti kosinusne podobnosti	37,87
36. Geary's C	38,58
37. Število samostalnikov, ednina (pos_NN)	39,41
38. Gunning Fog index	40,31
39. Clark Evans-ov najbližji sosed	41,2
40. Word Variation index	42,08
41. Flesch Reading Ease	42,32
42. Advanced Guiraud's Index	46,58
43. Število določnikov (pos_DT)	47,13
44. Število osebnih zaimkov (pos_PRP)	47,32
45. Število pravopisnih napak	47,69
46. D estimate	48,09
47. Število predlogov (pos_IN)	49,69
48. Povprečna razdalja med najbližjimi sosedi	51,42
49. Povprečna dolžina besed	54,63
50. Število stavkov	56,73

Tabela 4.4: Primerjava AGE in AGE+ z najboljšim izborom atributov glede na podatke.

	AGE				AGE+			
	TF-IDF		GloVe		TF-IDF		GloVe	
	LR	RF	LR	RF	LR	RF	LR	RF
DS1	0,8513	0,8513	**0,8555**	0,8542	0,8551	0,8587	0,8516	**0,8602**
DS2a	**0,7531**	0,7428	0,7508	0,7223	0,7602	0,7446	**0,7688**	0,7335
DS2b	**0,725**	0,6649	0,6983	0,6894	**0,7234**	0,7021	0,722	0,6965
DS3	**0,715**	0,7105	0,7121	0,7095	0,722	**0,7225**	0,7214	0,707
DS4	**0,795**	0,7811	0,7631	0,7533	**0,7926**	0,7839	0,7618	0,7452
DS5	**0,8352**	0,8210	0,8243	0,8163	**0,8373**	0,8266	0,8335	0,8267
DS6	**0,7996**	0,7732	0,7687	0,7502	**0,8083**	0,7917	0,7796	0,7622
DS7	**0,8166**	0,8137	0,8164	0,813	0,8196	0,8123	**0,8219**	0,8163
DS8	0,7897	0,7696	**0,7941**	0,7736	0,7939	0,7746	**0,7968**	0,7635
AVG	**0,7867**	0,7698	0,7759	0,7646	**0,7903**	0,7797	0,7842	0,7679

Ker zgornji pristop ni primeren za posplošitev na celotno zbirko podatkov (saj je preveč prilagojen posameznim iteracijam prečnega preverjanja), poskusimo z drugačnim pristopom izbiranja atributov. Namesto, da prilagajamo izbor atributov posameznim iteracijam prečnega preverjanja, kot merilo za izbor atributov upoštevamo povprečje vseh iteracij, torej klasični izbor atributov (angl. *forward attribute selection*). S tem pričakujemo slabši rezultat, saj izbor atributov ne bo več prilagojen posameznim iteracijam, temveč njihovemu povprečju — posplošen na celotno zbirko.

Tabela 4.5 prikazuje rezultate omenjenega postopka. Tabela sledi isti strukturi kot prejšnja (4.4), le da smo tukaj izpustili rezultate za predstavitev besed z GloVe, saj TF-IDF nenehno dosega boljše rezultate. Opazimo, da so rezultati v skladu s pričakovanji slabši. Najboljši rezultat še vedno dosega linearna regresija z uporabo atributov sistema AGE, atributi AGE+ pa ta rezultat nekoliko poslabšajo. Pri naključnih drevesih je rezultat z AGE+ malenkost boljši od AGE.

Tabela 4.5: Primerjava AGE in AGE+ s posplošenim naborom atributov.

	AGE		AGE+	
	LR	RF	LR	RF
DS1	0,8401	0,8375	0,8368	0,8375
DS2a	0,6927	0,7093	0,7144	0,7222
DS2b	0,6732	0,5882	0,6294	0,5882
DS3	0,6623	0,6799	0,663	0,6657
DS4	0,754	0,7708	0,7569	0,7613
DS5	0,8024	0,8082	0,8044	0,8163
DS6	0,7727	0,7692	0,7722	0,7788
DS7	0,7945	0,7971	0,8067	0,7991
DS8	0,7677	0,7236	0,7668	0,7261
AVG	0,7511	0,7426	0,7501	0,7439

Smiselno je pričakovati, da bi moral vsaj v večini primerov, z ustreznim izborom atributov, rezultat sistema AGE+ biti vsaj nekoliko boljši od sistema AGE, saj poleg novih atributov vključuje tudi vse atribute, ki so del sistema AGE. Ker gre pri izboru atributov z uporabo prečnega preverjanja za povprečenje in posplošitev najboljših atributov čez vse iteracije, ugotavljamo, da pri izboru atributov slej kot prej prispemo v lokalni optimum nabora atributov.

Pri naslednjem testu skušamo te lokalne optimume dokazati in izboljšati rezultat splošnega izbora atributov (glej tabelo 4.5). Postopek izbora atributov ostane enak kot pri prejšnjem testu s razliko, da ko pridemo do ustavitvenega pogoja — noben dodatni atribut ne izboljša rezultata —, v nabor atributov kljub temu dodamo najboljši atribut. To končni rezultat sicer poslabša (smo vendarle v lokalnem optimumu), vendar upamo, da po nekaj iteracijah slabšanja začnemo napredovati proti naslednjemu (lokalnemu ali globalnemu) optimumu in tako pridemo do boljšega končnega rezultata. Pri tem dovolimo največ dvajset zaporednih dodajanj atributov, kjer se rezultat poslabša. Ko se ta meja doseže, postopek ustavimo in uporabimo nabor

Tabela 4.6: Primerjava AGE in AGE+ s posplošenim naborom atributov in poskusom izogibanja lokalnim optimumom.

	AGE		AGE+	
	LR	RF	LR	RF
DS1	0,84	0,843	0,8369	0,8462
DS2a	0,7200	0,7291	0,7158	0,7222
DS2b	0,7023	0,6777	0,6941	0,6767
DS3	0,6648	0,6898	0,6656	0,6874
DS4	0,7698	0,7716	0,7619	0,7745
DS5	0,8079	0,8148	0,8028	0,8194
DS6	0,7808	0,7791	0,7771	0,7859
DS7	0,8081	0,8031	0,8083	0,8031
DS8	0,7673	0,7287	0,7681	0,7296
AVG	0,7623	0,7597	0,759	0,7606

atributov, ki je dajal najboljši rezultat.

Tabela 4.6 prikazuje rezultate pri izboru atributov z opisanim postopkom izogibanja lokalnim optimumom. V vseh primerih so rezultati veliko boljši, kar dokazuje prisotnost lokalnih optimumov in pomen pravilnega izbora atributov za ustrezno posplošitev modelov. Še vedno najboljši rezultat dosega linearna regresija z uporabo atributov sistema AGE, obenem pa je podobno kot prej (tabela 4.5) rezultat z atributi AGE+ nekoliko slabši. Naključna drevesa z atributi AGE+ ponovno dosegajo malenkost boljši rezultat kot z atributi AGE.

Poskusili smo tudi z druge strani — postopek izločanja atributov (angl. *backward attribute elimination*). Postopek je ravno obraten od izbiranja atributov. Začnemo z naborom vseh atributov in z vsako iteracijo odstranimo atribut, katerega izločitev pripomore k boljšemu rezultatu. Ponavljamo, dokler ni več atributa, ki bi z izključitvijo izboljšal rezultat. Tudi tu je nevarnost pristanka v lokalni optimum, zato smo uporabili enako tehniko izogibanja lokalnim optimumom kot v prejšnjem testu z izbiranjem atributov.

Tabela 4.7: Primerjava rezultatov z izbiranjem in izločanjem atributov za sistema AGE in AGE+ (TF-IDF)

	Izbiranje		Izločanje	
	AGE	AGE+	AGE	AGE+
DS1	0,84	0,8369	0,8449	0,8439
DS2a	0,72	0,7158	0,7237	0,7324
DS2b	0,7023	0,6941	0,7068	0,7028
DS3	0,6648	0,6656	0,6798	0,6958
DS4	0,7698	0,7619	0,7725	0,7769
DS5	0,8079	0,8028	0,8078	0,8122
DS6	0,7808	0,7771	0,7814	0,7871
DS7	0,8081	0,8083	0,8156	0,8183
DS8	0,7673	0,7681	0,7742	0,7717
AVG	0,7623	0,759	0,7674	**0,7712**

Tabela 4.7 na levi polovici prikazuje rezultate prejšnjega testa (izbiranja atributov z izogibanjem lokalnih optimumov), na desni pa rezultate izločevanja atributov z enako metodo izogibanja optimumom. Prikazani so rezultati za oba sistema (AGE in AGE+) z modelom linearne regresije in TF-IDF predstavitvijo besed. Naključne gozdove smo pri tem testu izpustili, saj je večkratno testiranje (skoraj) celega nabora atributov preveč časovno zahtevno, poleg tega pa se je do zdaj linearna regresija izkazala precej bolje.

Rezultati so dobri. Če primerjamo sistem AGE z izbiranjem in izločanjem atributov, vidimo, da se je izločanje obneslo nekoliko bolje. Zanimivejši pa je rezultat sistema AGE+ z izločanjem atributov, saj je ta rezultat najboljši od vseh doslej (brez prilagajanja) in kar je najpomembneje — ni slabši od rezultata sistema AGE kot pri izbiranju atributov. Sklepamo, da je tako zaradi narave obeh metod. Pri izbiranju atributov, kjer začnemo s praznim naborom, je ključno, da ne pridemo prehitro v lokalni optimum. Če pridemo v optimum relativno na začetku postopka, bodo v naboru atributov prevladovali osnovni atributi (npr. število znakov, besed itd.), saj ti najbolj vplivajo

na končni rezultat. Dodatni atributi sistema AGE+ pridejo do izraza šele, ko izčrpamo informacije drugih atributov. Tej težavi se izogne metoda izločanja atributov, saj začnemo s polnim naborom, kjer obstaja velika verjetnost, da bomo obdržali večino ali vsaj nekaj dodatnih atributov sistema AGE+. Če primerjamo velikost naborov atributov, vidimo, da so praviloma končni nabori pri izločanju atributov povprečno vsaj dvakrat večji.

S tem zaključimo, da je za naš primer najprimernejša metoda izločanja atributov, saj tako v končnem naboru obdržimo večjo količino informacij in se izognemo prehitrim lokalnim optimumom.

4.4 Analiza sistema SAGE

Podrobno smo analizirali tudi sistem SAGE. Najprej poglejmo primerjavo vseh treh sistemov z različnimi modeli in predstavitvami besed (podobno kot na začetku prejšnjega poglavja, kjer smo analizirali sistem AGE+, tabeli 2.1 in 2.2), kjer smo vse atribute neposredno pripeljali v model. Rezultati so prikazani v tabeli 4.8.

Ker je pri sistemu SAGE težava števila atributov še nekoliko večja kot pri AGE in AGE+, smo tudi tukaj opravili podobno analizo kot v prejšnjem poglavju. Testirali smo različne načine izbire ustreznih atributov — izbiranje in izločevanje atributov. Ker se je v prejšnjem poglavju izkazalo, da obstaja nevarnost lokalnih optimumov, smo za ta test uporabili način izogibanja optimumom (toleriramo do dvajset slabših zaporednih izborov atributov). Analizo zaradi preglednosti naredimo na enem od štirih podatkovnih zbirk.

V tabeli 4.9 primerjamo uspešnost sistema SAGE na podatkovni zbirki štiri (*DS4*) z uporabo različnih pristopov izbire atributov. Rezultate primerjamo tudi z rezultati sistema AGE in AGE+. Poleg vsakega rezultata je podano število atributov, ki sestavljajo končni nabor.

Tabela je razdeljena na štiri teste. Pri vseh testih smo se poskušali izogniti lokalnim optimumom.

Tabela 4.8: Primerjava sistemov AGE, AGE+ in SAGE na podatkovnih zbirkah, ki imajo izvorno besedilo (3, 4, 5 in 6), z modeloma linearne regresije in naključnih dreves, neposredno, brez izbiranja atributov.

DS3	Model	AGE	AGE+	SAGE
TF-IDF	RF	0,64413	**0,65286**	0,64828
	LR	0,65776	0,66217	**0,66728**
GloVe	RF	**0,6527**	0,64797	0,64827
	LR	0,63978	**0,645**	0,64049
DS4	Model	AGE	AGE+	SAGE
TF-IDF	RF	0,74383	0,74483	**0,75047**
	LR	0,75357	**0,75471**	0,75379
GloVe	RF	**0,71056**	0,70798	0,70839
	LR	0,71582	0,71482	**0,71948**
DS5	Model	AGE	AGE+	SAGE
TF-IDF	RF	**0,79942**	0,79429	0,79051
	LR	**0,79641**	0,79553	0,79113
GloVe	RF	**0,79187**	0,78433	0,78603
	LR	0,78745	**0,78867**	0,78753
DS6	Model	AGE	AGE+	SAGE
TF-IDF	RF	**0,75311**	0,74793	0,74945
	LR	**0,77343**	0,76749	0,76862
GloVe	RF	**0,74555**	0,74005	0,7428
	LR	**0,74387**	0,74052	0,74119

Tabela 4.9: Primerjava različnih testov za sisteme AGE, AGE+ in SAGE na podatkovni zbirki 4

	AGE		AGE+		SAGE	
Test	Rezultat	Št. atr.	Rezultat	Št. atr.	Rezultat	Št. atr.
1	**0,76978**	49	0,76187	25	0,76187	25
2	0,77255	41	0,77688	42	**0,77876**	49
3	0,76745	34	**0,77097**	59	0,76453	79
4	**0,76978**	49	0,76187	25	0,76555	26

- Prvi test predstavlja navadni izbor atributov z modelom linearne regresije.
- Drugi test predstavlja uporabo izločanja atributov z uporabo modela linearne regresije.
- Tretji test je enak prvemu, le da smo uporabili naključne gozdove.
- Četrti test je tudi enak prvemu, s tem, da smo na začetku pri izbiri atributov v začetni nabor ročno dodali dodatne atribute sistema SAGE.

Vidimo, da glede na ugotovitve iz prejšnjega poglavja najboljše rezultate dosega izločanje atributov. Nazorno se tudi vidi, da pri takem postopku pri večjem številu atributov (AGE+ in SAGE) obdržimo veliko več atributov kot pri navadnem izbiranju (npr. prvi test). Opazimo tudi, da je pri prvem testu izbor atributov sistemov AGE+ in SAGE povsem enak. To je tudi povod za četrti test (prisilimo uporabo SAGE atributov), kjer se je SAGE odrezal nekoliko bolje kot AGE+, vendar je v končni izbor še vedno prišlo zelo malo atributov, končni rezultat pa je tudi slabši od osnovnega sistema AGE. Tretji test z uporabo naključnih dreves kaže, da se je sistem SAGE odrezal slabše kot oba AGE in AGE+, kar lahko pripišemo lokalnemu optimumu. Na splošno bi na tem mestu opozorili, da so razlike med posameznimi sistemi minimalne, pridobivanje atributov (predvsem pri SAGE) pa v nekaterih primerih zelo časovno in procesorsko zahtevno.

Tabela 4.10: Primerjava sistemov AGE, AGE+ in SAGE z modelom linearne regresije in izločanjem atributov

	AGE	AGE+	SAGE
DS1	0,8449	0,8439	0,8439
DS2a	0,7237	0,7324	0,7324
DS2b	0,7068	0,7028	0,7028
DS3	0,6798	0,6958	0,6964
DS4	0,7725	0,7769	0,7788
DS5	0,8078	0,8122	0,8136
DS6	0,7814	0,7871	0,7864
DS7	0,8156	0,8183	0,8183
DS8	0,7742	0,7717	0,7717
AVG	0,7674	0,7712	0,7716

V slogu drugega testa, kjer se je sistem SAGE na podatkovni zbirki 4 najbolje odrezal, smo s sistemom SAGE testirali tudi ostale zbirke (3, 5 in 6). Rezultati so prikazani v tabeli 4.10, kjer so dodani tudi rezultati sistemov AGE in AGE+. Stolpec SAGE ima pri zbirkah 1, 2a, 2b, 7 in 8 zapisane enake rezultate, kot so v stolpcu AGE+. Pri sistemih AGE in AGE+ smo upoštevali najboljše rezultate — model linearne regresije z izločanjem atributov.

4.4.1 Primeri povratne informacije

V tem razdelku opisujemo dva primera povratne informacije semantične skladnosti sistema SAGE.

Prvi primer predstavlja osnovno nasprotje dveh povedi. Če na vhod sistema vstavimo besedilo: “Lisa is a girl. Lisa is a boy.” sistem javi semantično napako. Osnovna povratna informacija se glasi: “Relation ’Lisa is a boy.’ is inconsistent with a relation in ontology: ’Lisa is a girl.’”, ob omogočeni podrobni razlagi pa nam sistem dodatno vrne: “Lisa is Boy. Concepts Boy and

Girl are opposite/disjoint. Lisa is Girl." Osnovna razlaga poskuša kar se da najbolje odkriti, katere povedi so med seboj semantično neskladne, podrobne informacije pa vrnejo podrobno pot v grafu ontologije, ki pripelje do napake. Iz primera vidimo, da sta v ontologiji koncepta "boy" in "girl" nasprotna. To pomeni, da noben razred ali posameznik ne more pripadati obema konceptoma, kar sproži napako. Nekoliko kompleksnejše je besedilo: "Lisa is a girl. Lisa is a man." Sistem na osnovi sopomenk ugotovi, da sta si "man" in "woman" nasprotna, "girl" pa je podrazred/podpomenka "woman".

Drugi primer prikazuje še delovanje razreševalnika koreferenc na nekoliko kompleksnejši povedi. Vhod "George likes basketball and doesn't like sports." sproži napako z osnovno razlago: "Relation 'George likes basketball and George doesn't like sports.' is inconsistent with a relation in ontology: 'George likes basketball and George doesn't like sports.'" in podrobno razlago: "Relation not consistent: Georg likes Basketball. Relations doesNotLike and likes are opposite/disjoint. Relation not consistent: Georg doesNotLike Basketball." Osnovna razlaga deluje na ravni povedi in nam v tem primeru pove, da je poved v nasprotju sama s sabo. Podrobna razlaga pravi, da George ima in nima rad košarke. Beseda "sports" se v podrobni razlagi ne pojavi, ker je košarka podrazred športa in tam najprej pride do nasprotja. Zaradi lematizacije in korenjenja lahko pri nekaterih elementih trojice manjkajo zadnje črke besed.

4.5 Primerjava z nevronskimi modeli

Do sedaj smo opisovali samo delo s klasičnimi metodami izračuna in izbora atributov. Poskušali smo uporabiti tudi nevronske modele, ki so se v splošnem na številnih področjih dokazali za zelo uporabne.

Martinc in sod. [20] opisujejo uspešnost treh različnih nevronskih modelov pri ocenjevanju besedil. Med temi tremi modeli ni bilo absolutnega zmagovalca, zato smo za naš test vzeli dva modela in ju preizkusili na naših podatkih. Uporabili smo modela HAN (*Hierarchichal attention networks*) in

Transfer learning z modelom BERT. Delovanje obeh modelov smo najprej preverili s testnimi podatki, ki so na voljo poleg modelov in pri obeh modelih dosegli ustrezen pričakovan rezultat. Kot hitri kazalnik uspešnosti obeh modelov na naših podatkih smo uporabili prvo podatkovno zbirko (*DS1*), ki smo jo razdelili na učno in testno množico. Rezultati niso bili preveč navdušujoči, saj se je kvadratno utežena kapa pri obeh modelih gibala okoli 0,4. Za primerjavo je rezultat, pridobljen s prečnim preverjanjem, z našim sistemom na tej podatkovni zbirki približno 0,83 (razvidno iz tabel 4.1 in 4.2 z našimi rezultati).

Alikaniotis in drugi so objavili članek [21], v katerem so testirali uspešnost različnih nevronskih modelov na enaki podatkovni zbirki esejev, kot smo jo uporabljali mi. Najboljši model naj bi dosegal rezultat 0,96 (kvadratno utežena kapa). Zaradi odličnega rezultata smo preverili njihovo metodologijo in (javno objavljeno) implementacijo modela. Objavljen je samo model kot samostojna splošna enota, tako da iz tega ni mogoče sklepati, kako točno (parametri, podatki) so model testirali, torej natančna reprodukcija ni mogoča. Idejo za potencialno napako smo dobili iz bloga,[3] na katerem je avtor citiral ta članek in implementiral podoben, vendar preprostejši model, ki dosega enak rezultat — 0,96. Rezultat je sicer odličen, vendar trdimo, da je pri tej implementaciji bila storjena ključna napaka. Avtor je model učil na vseh podatkovnih zbirkah skupaj, čeprav ima vsaka zbirka drugačen obseg možnih ocen. Ker za testiranje in ocenjevanje modelov uporabljamo kvadratno uteženo kapo, je to na neki način "goljufanje", saj je rezultat te mere zelo odvisen od minimalne in maksimalne možne napovedne vrednosti (ocene eseja), ki sta v našem primeru 0 in 60. Tako bo odstopanje za npr. 2 na lestvici od 1 do 6 upoštevano kot odstopanje za 2 na lestvici od 0 do 60 (oz. minimalne in maksimalne vrednosti), kar bo privedlo do zelo dobrega rezultata. Za utemeljitev teh trditev smo na enak (napačen) način testirali tudi naš sistem. Z 10-kratnim prečnim preverjanjem smo z naključnimi goz-

[3] `https://medium.com/analytics-vidhya/automated-essay-scoring-kaggle-competition-end-to-end-project-implementation-part-1-b75a043903c4`

dovi dosegli rezultat 0,975 (z linearno regresijo nekoliko manj — 0,94). S tem potrjujemo skepso, da je bilo učenje zelo uspešnega nevronskega modela napačno, kar je privedlo do (pre)visokih rezultatov.

Sumimo, da so Alikaniotis in drugi v prej omenjenem članku storili podobno napako, saj objavljeni rezultati niso razdelanih po podatkovnih zbirkah, ampak je zabeležen le končni rezultat. Tega sicer ni mogoče preveriti, saj v javno objavljeni kodi ni postopka uporabe, ampak samo implementacija (najboljšega) modela, ki so ga uporabili. V članku so poleg utežene kape zabeležili tudi nekaj drugih relevantnih mer, zato smo poskušali nekaj sklepati na osnovi mere RMSE (*root mean square error*). Njihov najboljši model dosega vrednost kape 0,96 in RMSE 2,4. Vsi slabši modeli imajo vrednost RMSE večjo, kar je pričakovano. Njihovo vrednost RMSE smo primerjali s svojo in sicer v dveh scenarijih: vrednost RMSE, ko učimo na vseh podatkovnih zbirkah skupaj (torej napačen način) in povprečna vrednost RMSE vseh podatkovnih zbirk posebej. V prvem scenariju smo dobili vrednosti 2,8 za linearno regresijo in 1,8 za naključne gozdove — tu se naša kvadratno utežena kapa giblje okoli 0,97. Gibali smo se nekje v rangu njihove vrednosti RMSE. V drugem scenariju (zaradi hitrosti smo testirali samo linearno regresijo) smo dosegli RMSE približno 1,2 (ob kapi 0,75), kar je za polovico manj od njihovega rezultata 2,4 pri kapi 0,96. Po pričakovanjih bi sledilo, da večja kapa korelira z manjšo absolutno napako, vendar se pri napačni rabi kape (različni razponi ocen) ta korelacija nekoliko podre. Ravno to opazimo v prej podanih rezultatih — z manjšo kapo ob pravilni rabi imamo za polovico manjšo absolutno napako kot avtorji članka s skoraj popolnim strinjanjem (kapa skoraj 1). S tem je sum napačnega ocenjevanja modelov bolj utemeljen. Primerjava različnih modelov v članku je kljub temu morda še vedno veljavna.

Tudi Taghipour in Tou Ng [22] sta primerjala različne nevronske modele za ocenjevanje esejev (na istih podatkih kot mi). Številke so podobne našim, koda pa je tudi javna. Implementacijo smo testirali na vzorcu esejev in potrdili, da se na prvi pogled številke približno ujemajo. Glede na članek

se je kot najboljši model izkazal ansambel konvolucijske nevronske mreže in mreže LSTM z rezultatom 0,761, kar je malenkost višje kot naš najboljši rezultat z uporabo vseh atributov (glej tabeli 4.1 in 4.2), vendar nam je z ustreznim izborom atributov ta rezultat uspelo izboljšati in preseči tudi njihovega (glej tabelo 4.7).

Nevronski modeli so se na številnih področjih izkazali kot zelo uspešni, tudi v razumevanju naravnega jezika ter ocenjevanju besedil in esejev. Na tem področju so odprte možnosti za nadaljnje delo, saj imamo pri nevronskih modelih veliko različnih parametrov in navsezadnje različnih struktur. Za bolj poglobljeno analizo bi si bilo treba vzeti več časa, da bi lahko razvili učinkovit model, ki bo dajal dobre rezultate.

Poglavje 5

Zaključek in diskusija

V sklopu tega dela smo implementirali sistem za ocenjevanje esejev po zgledu dela Kaje Zupanc. Sistem temelji na ekstrakciji velikega števila atributov iz besedil in nato izboru najboljšega nabora za določeno podatkovno zbirko. Inovativni del je dodaten sistem za preverjanje semantične skladnosti, s pomočjo katerega nabor atributov dodatno obogatimo, obenem pa imamo možnost, da nam sistem izpiše vse zaznane semantične napake oz. neskladja.

Sistem smo implementirali v programskem okolju Orange, ki omogoča preprosto uporabo in kombiniranje različnih metod strojnega učenja. Osnovna komponenta v Orangeu je gradnik. Skupno smo implementirali tri gradnike, ki zajemajo sisteme AGE, AGE+, SAGE in evalvacijo modelov z metrikama kvadratno uteženе kape in natančnega ujemanja. Nalaganje esejev in ustvarjanje ter učenje modelov smo prepustili že obstoječim gradnikom. Koda je javna in na voljo na repozitoriju git.[1]

Pridobljeni rezultati niso povsem primerljivi, saj ni več na voljo originalne testne množice, na katero se je osredotočila Zupanc. Za konkretno primerjavo imamo na razpolago samo rezultate osnovnega sistema AGE z 10-kratnim prečnim preverjanjem, kjer pa so naši rezultati zelo podobni tistim, ki jih je dobila Zupanc. Njeni končni rezultati so sicer na testni množici boljši od naših, vendar je po njenih besedah testna množica že v osnovi bila takšna,

[1] `https://github.com/venom1270/essay-grading`

da so bili na njej precej boljši rezultati.

Poglobili smo se tudi v izbor atributov in preizkusili nekaj različnih metod. V splošnem se je v našem primeru najbolje izkazalo izločanje atributov, paziti pa je bilo treba tudi na lokalne ekstreme. Z uporabo teh metod nam je uspelo za nekaj decimalk izboljšati končni rezultat.

Dodatni atributi sistema SAGE (preverjanja semantične skladnosti) neposredno niso uspeli izboljšati rezultata. Boljši rezultat nam je na na eni od podatkovnih zbirk uspelo doseči z metodo izločanja atributov. Glede na zapisano bi rekli, da njihov priložnostni minimalni prispevek k rezultatu ne upraviči dolgega in zahtevnega procesiranja. Tudi Zupanc je statistično pomembno boljši rezultat uspelo doseči le na dveh od štirih podatkovnih zbirkah v primerjavi s sistemom AGE+. Kljub temu pa ima SAGE možnost izpisa semantičnih napak, ki je v določenih primerih lahko koristen in zanimiv za nadaljnje izboljšave. Sistem zagotovo ni popoln, saj je zanesljiv le toliko, kot je zanesljiv najšibkejši člen v verigi (predprocesiranje, rezreševanje koreferenc, ekstrakcija trojic itd.). Slabost je tudi že večkrat omenjena časovna in procesorska zahtevnost.

Pogledali smo si tudi nekaj nevronskih modelov, saj na številnih področjih dosegajo zavidljive rezultate. Z uporabo splošnih javnih modelov nam ni uspelo doseči dobrih rezultatov, našli pa smo nekaj člankov, v katerih so avtorji (na enaki podatkovni zbirki esejev kot mi) testirali različne nevronske modele. V enem od člankov je njihov model presegal naš neposredni rezultat z uporabo vseh atributov, vendar nam je uspelo z ustreznim izborom atributov ta rezultat preseči. Verjamemo, da bo z novimi odkritji in preizkusi z nevronskimi mrežami mogoče dosegati zelo dobre in boljše rezultate. Težava pri tematiki ocenjevanja esejev so zagotovo tudi podatkovne zbirke, saj je le-teh relativno malo (javno dostopnih). Pri nevronskih mrežah, ki bi potrebovale veliko podatkov, smo v tem primeru omejeni na zapise na socialnih omrežjih, blogih itd. To je tudi dobra iztočnica za nadaljnje raziskovalno delo.

Področje samodejnega ocenjevanja esejev predstavlja zanimiv problem,

saj bi pričakovali, da za relativno natančno oceno zahteva vsaj osnovno semantično razumevanje besedila. Klub temu smo ugotovili, da k rezultatu najbolj prispevajo relativno preproste značilnosti besedila – število različnih besed, število znakov itd. Že s tovrstnimi osnovnimi statistikami lahko zgradimo kompetenten model za ocenjevanje esejev in besedil, za bolj poglobljeno razumevanje pa potrebujemo kompleksnejše metode in procese. Z našim prispevkom implementacije sistema za ocenjevanje esejev v programskem okolju Orange uporabnikom na preprost način omogočamo uporabo in prilagajanje sistema ter uporabo vseh drugih metod, ki jih Orange že omogoča. Naš sistem omogoča tako uporabo samo osnovnih značilk kot tudi kompleksnejše preverjanje koherentnosti in semantične skladnosti.

Priložnost za konkretno izboljšanje rezultatov morda predstavljajo nevronski modeli, ki so se na številnih področjih že dokazali kot zelo uspešni. Sistem bi bilo smiselno preizkusiti tudi z drugimi napovednimi modeli, saj smo se v našem delu najbolj osredotočili le na linearno regresijo in naključne gozdove. Zanimiv izziv bi bil tudi prilagoditev sistema za slovenski jezik, ker je jezik sintaktično kompleksnejši, orodja za obdelavo besedil pa še niso tako zrela kot za angleški jezik.

Literatura

[1] K. Zupanc, Z. Bosnić, Automated essay evaluation with semantic analysis, Knowledge-Based Systems 120 (2017) 118–132.

[2] K. Zupanc, Semantics-based automated essay evaluation, Doktorska disertacija, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani (2018).

[3] R. Gunning, The Technique of Clear Writing, McGraw-Hill, 1968, pp. 37–39.
URL https://archive.org/details/techniqueofclear00gunn

[4] C. Brucker, Arkansas Tech Writing, Arkansas Tech University, 2009, pp. 109–110.
URL https://www.atu.edu/worldlanguages/texts/atw12th.pdf

[5] Readability Formulas, The New Dale-Chall Readability Formula, (datum dostopa: 3. 7. 2019).
URL https://readabilityformulas.com/new-dale-chall-readability-formula.php

[6] R. Senter, E. A. Smith, Automated readability index, Tech. rep., CINCINNATI UNIV OH (1967).
URL https://apps.dtic.mil/dtic/tr/fulltext/u2/667273.pdf

[7] G. H. Mc Laughlin, Smog grading-a new readability formula, Journal of reading 12 (8) (1969) 639–646.

[8] J. Anderson, Analysing the Readability of English and Non-English Texts in the Classroom with Lix (1981).

[9] C. Smith, A. Jönsson, Automatic summarization as means of simplifying texts, an evaluation for Swedish, 2011.

[10] Z. Siskova, Lexical Richness in EFL Students' Narratives, University of Reading Language Studies Working Papers 4 (2012) 26–36.

[11] J. Torruella, R. Capsada, Lexical Statistics and Tipological Structures: A Measure of Lexical Richness, Procedia - Social and Behavioral Sciences 95 (2013) 447–454. doi:10.1016/j.sbspro.2013.10.668.

[12] K. Tanaka-Ishii, S. Aihara, Computational constancy measures of Texts—Yule's k and rényi's entropy, Computational Linguistics 41 (3) (2015) 481–502. doi:10.1162/COLI_a_00228.
URL https://www.aclweb.org/anthology/J15-3004

[13] A. Mellor, Essay Length, Lexical Diversity and Automatic Essay Scoring, Memoirs of the Osaka Institute of Technology, Series B 55 (01 2011).

[14] D. Malvern, B. Richards, N. Chipere, P. Durán, Lexical Diversity and Language Development: Quantification and Assessment, Palgrave Macmillan UK, 2004, pp. 51–52, 189.

[15] M. Daller, R. Hout, J. Treffers-Daller, Lexical Richness in the Spontaneous Speech of Bilinguals, Applied Linguistics - APPL LINGUIST 24 (2003) 197–222. doi:10.1093/applin/24.2.197.

[16] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[17] Y. Attali, A differential word use measure for content analysis in automated essay scoring, ETS Research Report Series 2011 (2) (2011) i–19.

[18] L. Del Corro, R. Gemulla, ClausIE: Clause-Based Open Information Extraction, in: Proceedings of the 22nd international conference on World Wide Web, 2013, pp. 355–366.

[19] G. Stanovsky, I. Dagan, Creating a Large Benchmark for Open Information Extraction, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 2300–2305.

[20] M. Martinc, S. Pollak, M. Robnik-Šikonja, Supervised and unsupervised neural approaches to text readability, arXiv preprint arXiv:1907.11779 (2019).

[21] D. Alikaniotis, H. Yannakoudakis, M. Rei, Automatic text scoring using neural networks, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (2016). doi:10.18653/v1/p16-1068.
URL http://dx.doi.org/10.18653/v1/P16-1068

[22] K. Taghipour, H. T. Ng, A neural approach to automated essay scoring, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Austin, Texas, 2016, pp. 1882–1891. doi:10.18653/v1/D16-1193.
URL https://www.aclweb.org/anthology/D16-1193