Učinkovit izračun frekvenčnih statistik za slovenske jezikovne korpuse

Aleksander Ključevšek*, Simon Krek†°, Marko Robnik-Šikonja*

*Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana
† Univerza v Ljubljani, Filozofska fakulteta, Aškerčeva 3, 1000 Ljubljana
° Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana
aleksander.kljucevsek@gmail.com, simon.krek@guest.arnes.si, marko.robnik@fri.uni-lj.si

Povzetek

Veliki besedilni korpusi vsebujejo številne informacije o jeziku in njegovi rabi, nekatere lahko iz njih izluščimo s statistično analizo. Večina obstoječih orodij je razvitih in prilagojenih za obdelavo angleških besedil. V prispevku predstavljamo razvoj orodja za statistično analizo velikih slovenskih jezikovnih korpusov, ki upošteva značilnosti slovenščine kot morfološko bogatega jezika. Današnji besedilni korpusi lahko vsebujejo tudi več milijard besed, zato je bil velik del pozornosti namenjen razvoju učinkovitih paralelnih algoritmov, s katerimi bo moč tako obsežne zbirke v razmeroma kratkem času obdelati tudi na običajnih računalnikih. Orodje omogoča analizo na več nivojih: na nivoju besednih nizov, nivoju besed, n-gramov, predpon in končnic ter tudi oblikoskladenjskih oznak v slovenščini. Trenutno so podprti korpusi Gigafida, ccGigafida, Kres, ccKres, GOS in Šolar, vendar je dodajanje novih korpusov enostavno zaradi abstrakcije vhodnih podatkov.

Efficient calculation of frequency statistics for Slovene language corpora

Large text corpora hold a vast amount of information about language and its use, some of which can be extracted with statistical analysis. Most of existing tools are prepared for English texts. We present an application for statistical analysis of large Slovene text corpora, which takes into account rich morphology of Slovene. Since modern text corpora contain billions of words, we developed efficient parallel algorithms capable of processing these collections effectively using desktop computers. Our tool can analyze corpora on multiple levels: as strings, words, n-grams, through prefixes, suffixes, and POS tags. Currently the tools supports Gigafida, Kres, GOS, and Šolar, but adding support for new corpora is simple due to abstraction of input data.

1. Uvod

Čeprav obstajajo številni priročniki, ki formalizirajo pravila uporabe jezika, se jezik spreminja s časom in kontekstom, kar s časom privede do prilagoditve pravil. V zadnjem času se z vse večjimi možnostmi komunikacije razvijajo tudi nove specifične vrste komunikacije: jezik, uporabljen v kratkih sporočilih, se razlikuje od tistega, ki ga isti najstnik uporablja med pisanjem eseja, tako kot se jezik profesionalnega novinarskega članka razlikuje od jezika romanov. S statistično analizo besedil lahko dobimo vpogled v same temelje jezika: kako so sestavljeni stavki, kako pogosto se uporabljajo posamezne besede in besedne vrste, v kakšnih kombinacijah, v besedotvorne procese v različnih tipih besedil oz. v različnih tipih komunikacije itd. S tem spremljamo razvoj in spremembe v jeziku, odkrijemo pa lahko tudi nove jezikovne pojave. V članku opišemo učinkovito prosto dostopno orodje za statistično analizo besedil, ki podaja odgovore na ta in podobna vprašanja.

Velike množice besedil danes hranimo v zbirkah, imenovanih korpusi, večinoma zapisanih v formatu XML, ki omogoča enostavno dodajanje metapodatkov. Statistična analiza jezika je še posebej zanimiva na dovolj velikih korpusih, ki neredko dosežejo več milijard besed. Obdelava tolikšne količine podatkov lahko postane računsko zahtevna in dolgotrajna, še posebej če se ne uporabijo dovolj optimizirani algoritmi in primerna strojna oprema.

Da bi bilo naše orodje uporabno smo si zadali nekaj zahtev:

1. orodje mora biti zmožno učinkovito obdelati korpuse velikosti več milijard besed,
2. delovati mora tudi na enem samem, povprečnem računalniku,
3. sposobno mora biti izkoristiti razpoložljive pomnilniške in procesorske vire računalnika.

Prispevek je razdeljen na šest razdelkov. V 2. razdelku pripravimo kratek pregled obstoječih del. V 3. razdelku na kratko predstavimo zapis korpusov in podprte korpuse. V 4. razdelku predstavimo podprte funkcionalnosti, zgradbo našega orodja, težave, s katerimi smo se soočili, in analizo časovne ter prostorske zahtevnosti nekaterih uporabljenih algoritmov. V 5. razdelku predstavimo manjši vzorec zanimivih analiz. Prispevek zaključimo s 6. razdelkom, v katerem podamo glavne sklepe in navedemo možne izboljšave.

2. Sorodna dela

Eno prvih večjih statističnih analiz sta v 60-ih letih prejšnjega stoletja opravila Mayzner in Tresselt (1965), ki sta iz 100 različnih angleških virov (časopisi, revije, knjige itd.) ročno izločila po 200 zaporednih besed za skupno $20\,000$ besedni korpus. Te besede sta ročno prenesla na luknjane kartice in jih s pomočjo naprave za procesiranje kartic analizirala glede na dolžino in pozicijo črk znotraj besed. Norvig (2013) je ponovil analizo na angleških unigramih iz zbirke Google books Ngrams (Michel et al., 2011; Google, 2012). Uporabili so $97\,565$ različnih besed, ki so se v korpusu pojavile vsaj $100\,000-$krat in so bile skupaj omenjenje $743\,842\,922\,321-$krat, kar pomeni, da je bil korpus 37 milijonkrat obsežnejši kot originalni iz dela (Mayzner in Tresselt, 1965). Za izračun tako obsežnega

korpusa bi računalnik, ki ga je uporabljal Mayzner, potreboval 700 let. Med drugim je analiza razkrila, da število omemb besed glede na dolžino besede sledi Poissonovi distribuciji, pri čemer je $55.95\,\%$ najpogosteje uporabljenih besed dolžine $2-4$ črke. Povprečna dolžina besede je $4,79$ črk na besedo. Besede dolžine več kot 15 črk so uporabljene $834\,000\,000-$krat.

Lauer (1995) je pokazal, da lahko tudi preposte korpusne statistike dodajo informacije o sintaksi samostalniških fraz. Predstavil je štiri algoritme, ki za sintaktično analizo samostalniških fraz uporabljajo korpuse in delovanje algoritmov preizkusil na zbirki 244 izrazov. Trije algoritmi temeljijo na modelu sosednjosti (angl. adjacency model), medtem ko zadnji temelji na modelu odvisnosti (angl. dependency model). V vseh primerih se je za statistično značilno boljšega izkazal slednji.

Statistična analiza korpusov se je za pomembno izkazala tudi pri analizi podobnosti stavkov v kombinaciji s semantičnimi mrežami, kjer dobro odraža uporabo besed in izrazov v vsakdanji rabi (Li et al., 2006). Schiffman et al. (2001) so korpusne statistike uporabili pri avtomatskem generiranju povzetkov biografij. Njihov cilj so bili povzetki za enostavne biografije ljudi iz podatkov, ki so bili o njih objavljeni v medijih. Razvito metodo so uporabili na t. i. "Clintonovem korpusu" - korpusu sodnih zapisnikov predsednika Clintona na obravnavi po razkritju njegove afere. V zapisniku je bilo omenjeno veliko število ljudi in njihovih aktivnosti. Yang in Wilbur (1996) sta s pomočjo korpusnih statistik izračunala pomembnost posameznih besed in s tem skrčila seznam besed uporabljenih pri opisih namenjenih tekstovni kategorizaciji za $87\,\%$, kar je pripomoglo k $63\,\%$ krajšemu času obdelave takšnih seznamov in $74\,\%$ zmanjšanju porabe pomnilnika. Sočasno se je točnost napovedi v povprečju povečala za $10\,\%$ v primerjavi z neskrčenimi seznami.

Za kompleksnejše procesiranje slovenskega jezika v jezikoslovne namene je možno uporabiti odprtokodno orodje NooJ (Silberztein, 2016). Orodje omogoča analizo tekstov v več kot 20 jezikih, tudi slovenščini, na pravopisnem, leksikalnem, oblikoslovnem, sintaktičnem in semantičnem nivoju. Dobrovoljc (2014) je predstavila uporabo orodja za slovenščino. Orodje omogoča oblikovanje korpusnih poizvedb po površinski in označeni strukturi besedila, denimo za luščenje podatkov iz površinsko skladenjsko razčlenjenih korpusov, govornih korpusov ali drugih korpusov, ki poleg slovničnih lastnosti besednih oblik vsebujejo tudi druge vrste in ravni jezikoslovnih oznak. NooJ odlikuje vmesnik za preprost opis raznolikih jezikovnih pojavov v obliki grafov.

3. Korpusi

Za enostavno uporabo so korpusi shranjeni v strukturirani obliki. Če format to dovoljuje, so posameznim besedam dodani metapodatki, ki omogočajo uporabo dodatnih lastnosti besed pri analizi, npr. leme in oblikoskladenjske lastnosti besed. Vsi korpusi, ki jih naše orodje trenutno podpira, so v formatu XML.

Standard jezika XML se je zelo uveljavil v jezikovnotehnološki skupnosti, saj omogoča dobro berljivost ter shranjevanje in predstavitev strukturiranih jezikovnih podatkov. Osnovni standard jezika XML je moč prilagoditi specifičnim potrebam z dodatkom strožjih omejitev strukture in označevanja podatkov. Za namen shranjevanja označenih jezikovnih korpusov je konzorcij TEI (Text Encoding Initiative) izdal priporočila za označevanje besedil, ki obsegajo zapise različnih zvrsti besedil in jezikoslovnih korpusov. Prva dva nivoja etiket dokumentov, pripravljenih po standardu TEI, sta i) informacija o uporabljeni različici standarda XML in kodiranju ter ii) korenski element TEI, ki vsebuje kolofon in besedilo.

V kolofonu so shranjeni metapodatki o besedilu, npr. bibliografski podatki, struktura dokumenta, uporabljena taksonomija, opis značilnosti vsebovanega besedila itd. Samo besedilo se primerno strukturirano nahaja v naslednjem elementu. TEI predpisuje mnogo možnih oznak, ki jih lahko uporabimo pri strukturiranju in opisovanju besedil. Pri gradnji korpusov so najpogosteje uporabljani elementi za odstavke <p>, stavke <s>, besede <w>, ločila <c>in presledke.

V korpusih, ki jih trenutno podpira orodje, vsaka beseda poleg zapisa besede iz obravnavanega besedila vsebuje še atributa *lemma* in *msd*. V atributu *lemma* je shranjena lema (geselska iztočnica) besede, medtem ko je v atributu *msd* shranjena oblikoskladenjska oznaka besede, ki sledi specifikacijam MULTEXT-East različica 4.0 (Erjavec, 2012). Oznake za slovenski jezik so bile razvite v okviru projekta JOS (Erjavec in Krek, 2008) in zajemajo več kot 1900 oznak, ki so lahko izražene v slovenščini ali angleščini. V nadaljevanju na kratko predstavimo podprte korpuse.

3.1. Gigafida in KRES

Gigafida je referenčni korpus, ki v trenutni različici 1.0 (Holdt et al., 2012) vsebuje skoraj 1,2 milijarde besed (natančneje $1\,187\,002\,502$ besed), zajetih iz besedil, ki so v tiskani obliki ali na internetu izšla v obdobju med leti 1990 do 2011 (Erjavec in Logar Berginc, 2012). Korpus je zapisan v formatu XML TEI P5 (format XML z dodatnimi specifikacijami namenjenimi strukturirani predstavitvi besedil), je lematiziran in oblikoskladenjsko označen. Gigafida vključuje referenčni korpus FidaPLUS iz leta 2006 (621 milijonov besed) (Arhar et al., 2007) in tudi prvi slovenski referenčni korpus FIDA (1997 - 2000) (Erjavec et al., 1998).

Korpus Gigafida s svojo velikostjo in raznolikostjo besedilnih zvrsti predstavlja celovito podobo slovenskega jezika, ni pa uravnotežen, saj je kar $77\,\%$ besed iz periodičnih virov in samo $6\,\%$ besed iz knjig. KRES je uravnotežen podkorpus Gigafide, ki vsebuje skoraj 100 milijonov besed (natančneje $99\,831\,145$ besed) iz besedil izdanih med letoma 1990 in 2011, pri čemer je bil spletni del korpusa zbran in izdelan leta 2010. Ker je KRES podkorpus Gigafide, ji je po strukturi identičen.

3.2. GOS

Korpus GOvorjene Slovenščine GOS vsebuje več kot milijon besed zapisanih iz okrog 120 ur posnetkov (Verdonik et al., 2011). Vsa besedila so transkripcije posnetkov različnih vsakodnevnih situacij, ki so bili večinoma posneti med letoma 2008 in 2010. Nekatere od situacij so: radijske

in televizijske oddaje, predavanja, sestanki, svetovanje, zasebni pogovori med družinskimi člani ali prijatelji itd. Posnet govor je v korpusu shranjen v dveh različicah: pogovorni in standardizirani, pri čemer je standardizirani zapis tudi lematiziran in oblikoskladenjsko označen. Za reprezentativnost korpusa je bilo poskrbljeno tako z vključitvijo številnih različnih diskurzov kot tudi z reprezentativnostjo govorcev iz različnih regij, spolov, starosti in izobrazbenih ravni.

3.3. Šolar

Korpus Šolar je namenjen raziskovanju pisne jezikovne zmožnosti šolajoče se populacije (Rozman et al., 2012). Zajema skoraj milijon besed (967 477 besed) zajetih iz 2703 pisnih izdelkov (eseji oz. spisi, obnove, prošnje, odgovori na vprašanja itd.) šolarjev zadnjega triletja osnovne šole in srednješolcev. Poleg oblikoskladenjskih oznak vsebuje tudi označene učiteljske popravke. Za razliko od prej omenjenih korpusov Šolar ni skladen s TEI. Ne vsebuje taksonomije, vsebuje pa dodatne metapodatke, po katerih lahko filtriramo besedila: regija, predmet, razred, leto, šola in tip besedila.

4. Zgradba orodja

V tem razdelku predstavimo najprej interno predstavitev korpusnih podatkov, ki omogočajo iskalnim algoritmom enoten dostop, in zgradbo programske rešitve.

4.1. Interna predstavitev podatkov

Format TEI P5 XML, v katerem so zapisani vsi omenjeni korpusi z izjemo Šolarja, je primeren za shrambo jezikovnih korpusov in pripadajočih metapodatkov. Za enostavnejše procesiranje je potrebno podatke iz takšnega formata preneseti v podatkovno strukturo, s katero je moč enostavno upravljati v programski kodi. Za razvoj programa smo uporabili programski jezik java, zato smo izkoristili njegovo objektno naravo. Ustvarili smo dva tipa objektov, enega za stavke in drugega za besede, pri čemer je objekt *stavek* sestavljen iz množice objektov *beseda* in atributa za podatek o taksonomiji in ostalih lastnostih stavka. Objekt *beseda* je sestavljen iz znakovnih nizov za besedo, lemo besede in kode MSD (morfosintaktična oznaka). Posamezne etikete XML zapisa so tako analogne objektom v Javi, atributi etiket pa atributom objekta.

4.2. Zasnova programa

Za čim enostavnejšo razširljivost z novimi korpusi je program razdeljen na več ločenih nivojev, ki med seboj komunicirajo preko smiselnih programskih klicev: grafični vmesnik, opis strukture in metapodatkov korpusov, branje podatkov in računanje statistik. Struktura in pripadajoče lastnosti korpusov so interno že vnaprej definirane, ker je s tem omogočena hitrejša obdelava, kot če bi te podatke generirali sproti ob izračunu vsake statistike. Rezultati se shranijo v tekstovni tabelarični obliki, ločeni z vejico v formatu CSV (angl. comma separated values). S tem je poenostavljeno dodajanje novih korpusov v prihodnosti, saj je v primeru novega korpusa potrebno zgolj definirati taksonomijo in ostale atribute ter dodati metodo, ki korpus prebere v prej opisano strukturo.

Program v prvem koraku učinkovito analizira metapodatke podanega korpusa, zazna za kateri korpus gre in na podlagi tega ponudi statistike, ki jih je moč izračunati. Vsi korpusi namreč ne vsebujejo vseh vrst podatkov, npr. pogovorni del korpusa GOS ni oblikoskladenjsko označen, kar močno omeji nabor možnih izračunov. Uporabnik nato izbere statistiko, ki ga zanima, in program jo izračuna.

4.3. Učinkovita raba pomnilnika

Največja težava procesiranja velikih jezikovnih korpusov je, da jih zaradi njihove velikosti ni mogoče hraniti v pomnilniku. Korpus Gigafida zasede 83.5 GB pomnilnika, povprečen računalnik pa ima med 4 GB in 8 GB pomnilnika. Z velikostjo korpusa je povezano tudi počasno branje podatkov z diska . V nasprotju z računanjem statistik, ki ga na večjedrnih in večnitnih procesorjih lahko paraleliziramo, branje s trdega diska ostaja ozko grlo, zaradi česar program veliko časa porabi za branje podatkov.

Za rešitev te težave smo uporabili paketno obdelavo vhodnih podatkov:

1. Program prebira vhodne podatke, dokler ne prebere določenega števila stavkov, nakar začasno prekine branje. Količina prebranega besedila je določena glede na velikost razpoložljivega pomnilnika v računalniku.

2. Na prebranih podatkih delno izračunamo zahtevano statistiko.

3. Prebrani podatki se zbrišejo, s tem se sprosti prostor za nov paket in program nadaljuje z branjem novih podatkov, s čimer se vrnemo na točko 1.

4. Ko podatkov zmanjka, program izpiše kumulativen rezultat za zahtevano statistiko.

Postopek je natančneje prikazan v algoritmu 1. Časovna zahtevnost takšnega pristopa je linearna - $\mathcal{O}(n)$, kjer n predstavlja velikost vhodnih podatkov.

Algoritem 1 Paketno procesiranje korpusa

```
while v korpusu so še neprebrani stavki do
    subcorpus ← stavek
    if subcorpus.size ≥ limit then
        FORK-JOIN(subcorpus)
        subcorpus = ∅
    end if
end while
```

Tudi tako optimiziran pristop hitro naleti na omejitev: vsak objekt v javi ima določeno režijo (angl. overhead). Posamezen objekt tipa *beseda*, ki hrani besedo, lemo besede in nekaj črk MSD kode, zavzame 136 bajtov. Tako lahko v 1 GB pomnilnika v najboljšem primeru shranimo 7 352 941 besed, kar predstavlja 0.62 % besed v korpusu Gigafida. K temu je potrebno dodati še podatkovne strukture, v katerih hranimo računane statistike.

Zaradi podpore sočasnemu dostopu in konstantemu času vstavljanja in posodabljanja vrednosti smo kot osnovno podatkovno strukturo za hranjenje statistik izbrali

razpršeno tabelo tipa HashMap. To je podatkovna struktura, v kateri je vsak vnos shranjen v obliki "ključ: vrednost". HashMap ne more vsebovati podvojenih ključev. Za dostop do ključa se uporablja razpršilna funkcija, kar omogoča, da se vnos in preverjanje, ali določen element že obstaja v podatkovni strukturi, izvedeta v konstantnem času $\mathcal{O}(1)$. Struktura HashMap podpira več sočasnih dostopov in je tako prilagojena paralelnemu procesiranju. Kljub temu pa je za velike korpuse, kot je Gigafida, pomnilniška poraba precejšnja in lahko na računalniku s 16 GB pomnilnika naenkrat računamo samo eno statistiko tipa frekvenca besed. Za nekatere statistike, ki zgenerirajo obsežne tabele rezultatov pa tudi to ni dovolj pomnilnika, zato program omogoča sprotno shranjevanje rezultatov na disk, kar sprosti pomnilnik na račun počasnejšega delovanja.

4.4. Paralelizacija

Vsi moderni procesorji imajo več jeder in podpirajo večnitno procesiranje. S poganjanjem algoritmov na več jedrih oz. nitih se ustrezno skrajša čas izvajanja. Če želimo izračunati frekvenco vseh besed to pomeni, da bomo za vsako besedo najprej preverili, ali smo nanjo že naleteli, nakar bomo število pojavitev te besede povečali za 1. Za korpus Gigafida to pomeni več kot milijardo operacij za vsako tovrstno statistiko, ki jo želimo izračunati.

Pri reševanju tega problema smo preverili več opcij: tokove (angl. streams) v Javi 8 ter paralelne arhitekture Fork-Join, Map Reduce in Akka. Večina obstoječih rešitev je namenjena obdelavi podatkov, ki se nahajajo na več ločenih sistemih (npr. porazdeljeno računanje na način Map Reduce) in s poganjanjem na samo enem sistemu ne morejo izkoristiti vgrajenih optimizacij (Stewart in Singer, 2012; Ranger et al., 2007). Odločili smo se za implementacijo Fork/Join paralelizacije (De Wael et al., 2014; Ponge, 2011; Lea, 2000), ker so se tokovi v Javi 8 izkazali za nezanesljive in so v določenih pogojih lahko tudi nekajkrat počasnejši kot Fork-Join (Langer, 2015; Zhitnitsky, 2015).

Fork-Join je v osnovi paralelna verzija principa deli in vladaj, kjer začetni problem rekurzivno delimo na manjše naloge, dokler ne dosežemo dovolj majhne velikosti podproblemov. Te rešimo neposredno, nakar rešitve združimo nazaj v rešitev začetnega problema. Pri vzporednih algoritmih se manjši deli ločeno in istočasno računajo s samostojnimi nitmi. Pogoj za takšno delitv je neodvisnost posameznih podproblemov. Osnovno delovanje je prikazano v algoritmu 3. Podatkovnih struktur pri takšni obdelavi ne spreminjamo ali prepisujemo v nove, manjše, temveč je podproblem definiran kot pogled na omejen del celotne podatkovne strukture. V Javi 8 je bil dodan princip ForkJoinPool, ki med drugim omogoča krajo opravil (angl. work stealing): če ena nit zaključi s svojim delom in mora čakati, da s svojim zaključi še druga, lahko v vmesnem času prevzame nedokončana opravila druge niti. Če so podproblemi dovolj majhni, je tako procesiranje bolj učinkovito, ker lahko učinkoviteje izkoriščamo celotno kapacitete procesorja.

4.5. Časovna zahtevnost postopkov

Ob zaključku izračuna ene statistike na celotnem korpusu izračunane frekvence uredimo po padajočem zaporedju, za kar uporabimo javansko metodo TimSort, ki ima v najslabšem primeru časovno zahtevnost $O(n \log n)$ (Korniichuk, 2015). Skupno časovno zahtevnost celotnega programa dobimo s seštevanjem časovnih zahtevnosti posameznih delov:

Algoritem 3 Fork-Join algoritem.

```
1: procedure REŠI(problem)
2:     if problem je dovolj majhen then
3:         reši problem direktno/sekvenčno
4:     else
5:         razdeli na podproblemA in podproblemB
6:         fork REŠI(podproblemA)
7:         fork REŠI(podproblemB)
8:         join rešitvi podproblemov A in B
9:     end if
10: end procedure
```

- branje podatkov: $\mathcal{O}(n)$ +
- Fork-Join: $\mathcal{O}(k)$ +
- računanje statistike: $\mathcal{O}(n)$ +
- TimSort: $O(n \log n)$

Pri tem k pri koraku Fork-Join predstavlja število korakov delitve večjega problema na manjše (načeloma reda $\log_t n$), kjer je t število vzporedno delujočih niti. Končna časovna zahtevnost celotnega programa je tako enaka $O(n \log n)$.

4.6. Izračun statistik

Algoritme, ki statistiko izračunajo na opisan način smo razdelili v naslednje razrede:

LetterCount - vsebuje metode za računanje distribucij posameznih črk in njihovih zaporedij,

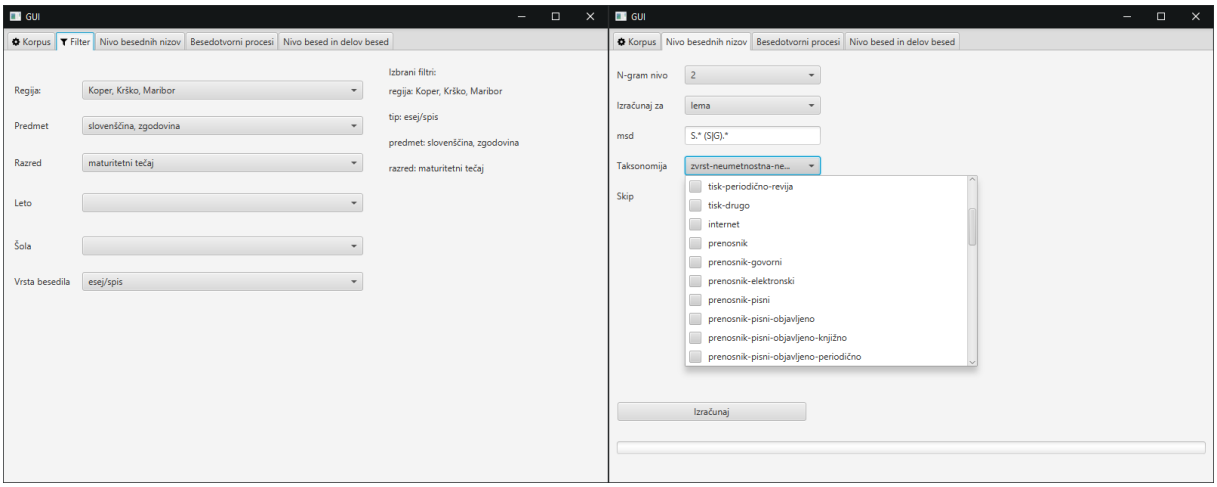
NGrams - vsebuje metode za računanje n-gramov besed in lem,

WordCount - vsebuje metode za računanje distribucij besed in lem,

WordLengthCount - vsebuje metode za računanje distribucij dolžin besed.

Metode v navedenih razredih kličemo z imenom metode in tremi parametri:

- podkorpusom oz. pogledom na del celotnega korpusa,
- referenco na HashMap, v katerega zapisujemo frekvence - ta ima v primeru računanja distribucije dolžin besed kot ključ objekt tipa Integer (celo število)
- z omejitvijo - to je lahko določena oznaka v taksonomiji, oblikoskladenjska oznaka iz tabele oznak JOS ali podatek, da določeno statistiko računamo za besedo, njeno lemo ali poljubno kombinacijo omenjenih kriterijev.



Slika 1: Prikaz zavihka za izbiro filtrov pri korpusu Šolar (levo) in zavihka Nivo besednih nizov (desno).

4.7. Grafični vmesnik

Grafični vmesnik orodja je strukturiran v zavihke in ilustriran na sliki 1:

Korpus - uporabnik izbere lokacijo korpusa in lokacijo kamor se bodo shranjevali rezultati. V primeru korpusa GOS se pokaže dodatna opcija, s katero označimo, če želimo statistike računati za govorni zapis.

Filter - zavihek je viden samo, če smo izbrali korpus Šolar, ki vsebuje dodatne označbe kot npr. regija, predmet, vrsta besedila itd.

Nivo besednih nizov - na tem zavihku lahko računamo statistike na nivoju črk ali n-gramov. V obeh primerih lahko za računanje uporabimo različnice ali leme, v primeru n-gramov lahko dodatno računamo še za oblikoskladenjske oznake, v primeru n-gramov stopnje 2 ali več pa je možno tudi računanje skip-gramov. Omogočeno je filtriranje po oblikoskladenjskih oznakah, ki podpira uporabo regularnih izrazov, s čimer lahko npr. računamo samo število pojavitev bigramov, kjer je prva beseda pridevnik v dvojini in druga lastno ime ženskega spola "P...d.. Slz.*", bigrami, kjer je prva beseda samostalnik, druga pa samostalnik ali glagol "S.* (S—G).*", pojavitve samostalnikov v imenovalniku "S...i" ali ekvivalenten "S.{3}i" itd. Možno je tudi filtriranje glede na taksonomijo besedila v korpusu oziroma, v primeru korpusa Šolar, filtriranje glede na regijo, predmet, razred, leto, šolo ali vrsto besedila. Na nivoju črk je možno tudi računanje frekvenc kombinacij samoglasnikov in soglasnikov poljubne dolžine.

Besedotvorni procesi - za pregibne besedne vrste program izračuna distribucijo oblik glede na tabelo oznak JOS. Pogoj za to je, da korpus vsebuje oznake msd. Izračuna se frekvenca vseh kombinacij lasnosti, s čimer dobimo odgovore na vprašanja kot so npr. "Se v korpusu pojavi več pridevnikov ženskega ali moškega spola?", "Kakšna je distribucija samostalnikov po sklonih?" ali tudi bolj specifična vprašanja kot npr. "Se večkrat pojavijo zanikani ali nezanikani dovršni glagoli velelne oblike?". Računanje lahko omejimo glede na taksonomijo.

Nivo besed in delov besed - program izračuna kolikokrat se v korpusu pojavi specifična predpona ali pripona na podlagi metode najdaljšega ujemajočega se podniza. Seznam predpon in pripon je podan programu kot dodaten vhod, program nato izračuna frekvenco. Računanje lahko omejimo glede na taksonomijo.

5. Rezultati

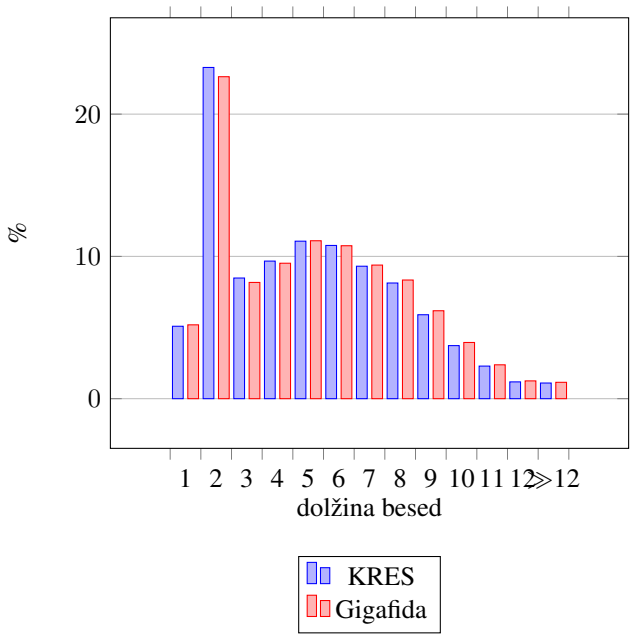
Kot primere izračunov, ki jih izvede naše orodje, navajamo nekaj statistik za slovenščino, izračunanih na korpusih Gigafida in KRES. Ker namen našega članka ni analiza teh frekvenc, pač pa demonstracija orodja, ki je prosto dostopno na `http://github.org/cjvt-ul/corpusStatistics`, predstavljamo primere izračunov za razrede statistik iz razdelka 4.6. Za prve tri razrede statistik navajamo najpogostejših 10 primerov posamezne izračunane statistike (tabele 5., 5., 5. in 5.), rezultate za WordLengthCount pa v obliki histograma navajamo na sliki 5..

	KRES		Gigafida	
	črka	%	črka	%
1.	a	10.12	a	10.01
2.	e	9.99	e	9.74
3.	o	9.07	o	9.03
4.	i	8.78	i	8.73
5.	n	6.74	n	6.69
6.	r	5.17	r	5.26
7.	s	4.57	t	4.47
8.	t	4.48	s	4.45
9.	l	4.46	l	4.36
10.	j	4.17	v	4.11

Tabela 1: Razred statistik LetterCount - 10 najpogosteje uporabljenih črk.

	KRES		Gigafida	
	števka	%	števka	%
1.	0	0.28	0	0.39
2.	1	0.27	1	0.33
3.	2	0.19	2	0.24
4.	3	0.12	3	0.15
5.	9	0.11	5	0.15
6.	5	0.11	4	0.12
7.	4	0.10	9	0.12
8.	8	0.08	6	0.10
9.	6	0.08	8	0.09
10.	7	0.07	7	0.09

Tabela 2: Razred statistik LetterCount - pogostost uporabe števk.



Slika 2: Razred statistik WordLengthCount - kako pogosto se pojavijo besede določene dolžine. Povprečna dolžina besede v korpusu KRES je $5,11$ črk in $5,18$ črk v korpusu Gigafida.

	KRES		Gigafida	
	bigram	%	bigram	%
1.	Slmei- Slmei-	0.75	Slmei- Slmei-	1.09
2.	Dm Sozem-	0.74	Dm Sozem-	0.74
3.	Vd Gp-ste-n	0.64	Vd Gp-ste-n	0.64
4.	Dm Somem-	0.62	Ppnzer- Sozer-	0.64
5.	Ppnzei- Sozei-	0.61	Dm Somem-	0.63
6.	Ppnzer- Sozer-	0.59	Ppnzei- Sozei-	0.63
7.	Rsn Rsn	0.55	Kag—- Kag—-	0.58
8.	Dt Sozet-	0.52	Ppnmeid Somei-	0.54
9.	L Rsn	0.50	L Rsn	0.51
10.	Kag—- Kag—-	0.50	Dt Sozet-	0.50

Tabela 3: Razred statistik NGrams - najpogostejši bigrami oblikoskladenjskih oznak. Pomen oznak je definiran v standardu MULTEXT-East (Erjavec, 2012) in dosegljiv na `http://nl.ijs.si/ME/V4/msd/html/msd-sl.html`.

	KRES		Gigafida	
	lema	%	lema	%
1.	biti	7.60	biti	7.34
2.	in	2.84	v	2.63
3.	v	2.47	in	2.56
4.	se	1.87	se	1.59
5.	na	1.51	na	1.58
6.	z	1.39	z	1.33
7.	da	1.28	za	1.31
8.	on	1.24	da	1.23
9.	za	1.19	ki	1.02
10.	ta	1.06	ta	1.01

Tabela 4: Razred statistik WordCount - 10 najpogosteje uporabljenih lem.

6. Zaključki

Predstavili smo orodje za učinkovit izračun frekvenčnih statistik na velikih korpusih. Orodje z večnitnostjo učinkovito izkorišča večjedrne procesorje in frekvence izračunava vzporedno, pri tem pa izračune deli tako, da ne preseže razpoložljivega pomnilnika.

Kot nadaljno delo vidimo možnost podpore še drugim slovenskim in tujim korpusom in drugim vhodnim formatom. Večji izziv predstavlja avtomatska detekcija formata in vrste korpusa iz formata XML. Takšna razširitev bi načeloma bila zmožna obdelati poljuben korpus. Možne razširitve programa so dodatne statistike, ki bi vključevale regularne izraze na besedah in oblikokladenjskih oznakah, ter različnim tipom uporabnikov in jezikom prilagojen uporabniški vmesnik.

Zahvala

Raziskavo je sofinancirala Javna agencija za raziskovalno dejavnost Republike Slovenije skozi projekt J6-8256 (Nova slovnica sodobne standardne slovenščine: viri in metode) in raziskovalni program P2-0209 (Umetna inteligenca in inteligentni sistemi).

7. Literatura

Špela Arhar, Vojko Gorjanc in Simon Krek. 2007. Fidaplus corpus of slovenian: the new generation of the slovenian reference corpus: its design and tools. V: *Proceedings of the Corpus Linguistics conference*.

Mattias De Wael, Stefan Marr in Tom Van Cutsem. 2014. Fork/Join Parallelism in the Wild: Documenting Patterns and Anti-patterns in Java Programs Using the Fork/Join Framework. V: *PPPJ'14, International Conference on Principles and Practices of Programming on the Java*

Platform: Virtual Machines, Languages, and Tools, str. 39–50, Cracow.

Kaja Dobrovoljc. 2014. Procesiranje slovenskega jezika v razvojnem okolju NooJ. V: Tomaž Erjavec in Jerneja Žganec Gros, ur., *Zbornik 9. konference Jezikovne tehnologije, Informacijska družba - IS 2014*, str. 79–84.

Tomaž Erjavec, Vojko Gorjanc in Marko Stabej. 1998. Korpus fida. V: *Proc. of the Intl. Multi-Conf. Intl. Society'98*, str. 124–127.

Tomaž Erjavec in Simon Krek. 2008. The JOS morphosyntactically tagged corpus of Slovene. V: *6th International Conference on Language Resources and Evaluation, Marrakech, Morocco, May 26 - June 1, 2008. LREC 2008: proceedings*, str. 322–326.

Tomaž Erjavec in Nataša Logar Berginc. 2012. Referenčni korpusi slovenskega jezika (cc)Gigafida in (cc)KRES. *Proceeding of the Eighth Language Technologies Conference*, str. 57–63.

Tomaž Erjavec. 2012. MULTEXT-East: morphosyntactic resources for central and eastern european languages. *Language resources and evaluation*, 46(1):131–142.

Google. 2012. Google books ngram corpus. `http://books.google.com/ngrams`.

Špela Arhar Holdt, Iztok Kosem in Nataša Logar Berginc. 2012. Izdelava korpusa gigafida in njegovega spletnega vmesnika. V: *Proceedings of 8th Eighth Language Technologies Conference IS-LTC*, zvezek 12.

Volodymyr Korniichuk. 2015. Timsort Sorting Algorithm. `http://www.infopulse.com/blog/timsort-sorting-algorithm/`.

Angelika Langer. 2015. Java performance tutorial – How fast are the Java 8 streams? `https://jaxenter.com/java-performance-tutorial-how-fast-are-the-java-8-streams-118830.html`.

Mark Lauer. 1995. Corpus statistics meet the noun compound: some empirical results. V: *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, str. 47–54. Association for Computational Linguistics.

Doug Lea. 2000. A java fork/join framework. V: *Proceedings of the ACM 2000 conference on Java Grande*, str. 36–43. ACM.

Yuhua Li, David McLean, Zuhair A Bandar, James D O'shea in Keeley Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE transactions on knowledge and data engineering*, 18(8):1138–1150.

Mark S Mayzner in Margaret Elizabeth Tresselt. 1965. Tables of single-letter and digram frequency counts for various word-length and letter-position combinations. *Psychonomic monograph supplements*.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig in Jon et al. Orwant. 2011. Quantitative analysis of culture using millions of digitized books. *science*, 331(6014):176–182.

Peter Norvig. 2013. English letter frequency counts: Mayzner revisited or etaoin srhldcu. `http://www.norvig.com/mayzner.html`.

Julien Ponge. 2011. Fork and join: Java can excel at painless parallel programming too! `http://www.oracle.com/technetwork/articles/java/fork-join-422606.html`.

Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski in Christos Kozyrakis. 2007. Evaluating mapreduce for multi-core and multiprocessor systems. V: *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on*, str. 13–24. Ieee.

Tadeja Rozman, Irena Krapš Vodopivec, Mojca Stritar in Iztok Kosem. 2012. *Empirični pogled na pouk slovenskega jezika*. Trojina, zavod za uporabno slovenistiko.

Barry Schiffman, Inderjeet Mani in Kristian J Concepcion. 2001. Producing biographical summaries: Combining linguistic knowledge with corpus statistics. V: *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, str. 458–465. Association for Computational Linguistics.

Max Silberztein. 2016. *Formalizing Natural Languages: The NooJ Approach*. John Wiley & Sons.

Robert Stewart in Jeremy Singer. 2012. Comparing fork/join and mapreduce. *Cite-seer, Tech. Rep.*, str. 1–20.

Darinka Verdonik, Ana Zwitter Vitez in Hotimir Tivadar. 2011. *Slovenski govorni korpus Gos*. Trojina, zavod za uporabno slovenistiko.

Yiming Yang in John Wilbur. 1996. Using corpus statistics to remove redundant words in text categorization. *JASIS*, 47(5):357–369.

Alex Zhitnitsky. 2015. How Java 8 Lambdas and Streams Can Make Your Code 5 Times Slower. `http://blog.takipi.com/benchmark-how-java-8-lambdas-and-streams-can-make-your-code-5-times-slower/`.