Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Aleksander Ključevšek

Statistična analiza slovenskih jezikovnih korpusov

diplomsko delo
na univerzitetnem študiju

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

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Veliki besedilni korpusi vsebujejo številne informacije o jeziku in njegovi rabi, nekatere lahko iz njih izluščimo s statistično analizo. Ker so korpusi lahko zelo veliki, je za izračun zanimivih statistik potrebno zasnovati učinkovite vzporedne algoritme. Pri analizi slovenskega jezika moramo, zaradi njegove pregibnosti, razviti prilagojene algoritme. Razvijte prototip orodja za statistično analizo slovenskih besedilnih korpusov, ki bo zmožno učinkovito izračunati statistike na nivoju besed, oblikoskladenjskih lastnosti in delov besed, pri tem pa upoštevati omejitve glede zvrsti in lastnosti upoštevanih besedil iz korpusa. Orodje preizkusite na korpusu Gigafida.

Za mentorstvo, pomoč in svetovanje pri pripravi diplomskega dela se najlepše zahvaljujem izr. prof. dr. Marku Robniku Šikonji.

Za pomoč pri razumevanju, nasvete in nazorne razlage se zahvaljujem somentorju dr. Simonu Kreku.

Še posebej pa se zahvaljujem mami, stari mami in staremu očetu. Najlepša hvala za vso požrtvovalno podporo in brezpogojno stanje ob strani skozi vsa - pogosto ne najlažja - leta. Diplomsko delo posvečam vam.

Kazalo

Povzetek

Naslov: Statistična analiza slovenskih jezikovnih korpusov

Področje procesiranja naravnega jezika je pomembna in obsežna panoga računalništva, vendar je večina obstoječih orodij razvitih in prilagojenih za obdelavo angleških besedil. Razvili smo orodje za statistično analizo velikih jezikovnih korpusov, ki upošteva značilnosti slovenščine kot močno pregibnega 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. Z orodjem smo analizirali korpus Gigafida, ki vsebuje 1,2 milijarde besed, na več nivojih: na nivoju besednih nizov, nivoju besed, n-gramov, predpon in končnic ter tudi besedotvorne procese v slovenščini.

Ključne besede: statistična analiza jezika, jezikovni korpus, Gigafida, paralelni algoritmi.

Abstract

Title: Statistical analysis of Slovene language corpuses

Natural language processing is an important area of computational linguistics and artificial intelligence . Mostly, its existing applications are developed for and based on English texts. We developed an application for the statistical analysis of large text corpora, which takes into account the unique characteristics of Slovene as a strongly inflected language. Since modern text corpora consist of several billion words, we paid special attention to efficient parallel algorithms that are capable of processing these collections in a relatively short amount of time. We analyzed the Gigafida corpus - consisting of 1.2 billion words - on multiple levels: string level, word level, n-gram level, prefix and suffix level, as well as word formation processes of Slovene.

Keywords: statistical language analysis, text corpus, Gigafida, parallel algorithms.

Poglavje 1

Uvod

Jezik verjetno predstavlja najpomembnejše orodje človeške vrste. Od začetkov v obliki preprostih zvokov in gest, se je skozi čas razvil v kompleksne kombinacije črk in besed, izrazov in pravil, ki nam omogočajo komunikacijo in sodelovanje. Študij jezika ni nič novega; že od prvih pojavov formalnih opisov jezikov v 5./6. stoletju pr. n. št. so se ljudje ukvarjali z analizo posameznih gradnikov jezikov [1]. S pojavom računalnikov se je razvilo področje procesiranja naravnega jezika (NLP - natural language processing), ki se med drugim ukvarja tudi z računalniškim razumevanjem in statistično analizo naravnega jezika.

Kar dela jezik še posebej zanimivo raziskovalno področje je, kako se spreminja s časom in kontekstom. Čeprav obstajajo številni priročniki, ki opisujejo pravila pravilne uporabe jezika, nanj in na ta pravila vplivajo množice piscev in govorcev, ki ga uporabljajo vsak dan. Vsaka generacija jezik prilagodi svojim potrebam in razmeram v družbi, nato tako spremenjen jezik prevzamejo nove generacije, ki v besedišče vnesejo nove izraze ali spremenijo kakšno pravilo. Sčasoma si takšne spremembe utrejo svojo pot v nove priročnike. S številnimi možnostmi za komunikacijo se razvijajo tudi 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 tistega, ki ga najdemo v

romanu.

S statistično analizo besedil, ki je tema tega diplomskega dela, lahko dobimo vpogled v same temelje jezika: kako so sestavljeni stavki, kako pogosto se uporabljajo posamezne besedne vrste in v kakšnih kombinacijah, ali obstaja razlika med besedotvornimi procesi v različnih tipih literature oz. v različnih tipih komunikacije itd. S tem lahko ne samo spremljamo razvoj in spremembe jezika, ampak lahko odkrijemo lako tudi nova pravila.

1.1 Predstavitev problema

Za statistično analizo jezika se uporabljajo korpusi, ki vsebujejo različne vrste besedil. Kot najpriročnejši format za shranjevanje se je uveljavil XML, ker omogoča vključevanje metapodatkov. Da bi bil posamezen korpus čim bolj reprezentativen in statistično pomemben, mora vsebovati zadostno število besed. Tako korpusi dosežejo obseg med nekaj sto milijoni do več milijard besed. Obdelava tolikšne količine podatkov lahko postane računsko zelo zahtevna, če se za opravilo ne uporabijo dovolj optimizirani algoritmi in primerna strojna oprema.

Osnovni cilj diplomskega dela je razvoj aplikacije in algoritmov za statistično obdelavo velikih korpusov. Glavne zahteve so bile;

- zmožnost obdelave korpusov velikosti do dveh milijard besed,
- aplikacija mora teči na enem računalniku,
- aplikacija mora izkoristiti razpoložljive sistemske kapacitete (paralelizacija, izraba pomnilnika) in
- izračun statistik mora biti opravljen v razumno hitrem času.

1.2 Pregled vsebine

Diplomsko delo je razdeljeno na 5 poglavij. V drugem poglavju podrobneje predstavimo področje, sorodna dela in uporabljene korpuse. V tretjem po-

glavju se posvetimo algoritmom in izzivom, ki so povezani z implementacijo učinkovitih načinov obdelave velikega števila podatkov. Sledi poglavje, namenjeno rezultatom, na koncu, v 5. poglavju, sledijo še kritična analiza in ideje za nadaljno delo.

Poglavje 2

Predstavitev področja

Statistična analiza besedil se je razvila že v starih časih. Preskok v učinkovitosti raziskovalnih orodij se je zgodil s pojavom računalnikov. V prvem delu poglavja predstavimo sorodna dela, ki se ukvarjajo s statistično obdelavo besedil, v drugem pa opišemo kaj korpusi so, nakar predstavimo korpus Gigafida.

2.1 Sorodna dela

Eno izmed najbolj poznanih analiz je v 60-ih letih prejšnjega stoletja opravil dr. Mark Mayzner, ki je iz 100 različnih angleških virov (časopisi, revije, knjige itd.) ročno izločil po 200 zaporednih besed za skupno 20.000 besedni korpus. Te besede je ročno prenesel na luknjane kartice in jih s pomočjo naprave za procesiranje kartic analiziral glede na dolžino in pozicijo črk znotraj besed [2]. Leta 2012 je dr. Mayzner kontaktiral Google s predlogom, da njegovi raziskovalci izvedejo enako analizo - tokrat s celotnim Googlovim korpusom podatkov in procesorsko močjo modernih strežniških sistemov. Vodja raziskovalne skupine Peter Norvig je sprejel izziv in analizo izvedel na unigramih v zbirki Google books Ngrams. Izbor je skrčil na 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 bolj obsežen kot Mayznerjev. Če bi poskusili tako obsežen korpus obdelati

z računalnikom, ki ga je uporabljal Mayzner, bi za posamezno statistiko potrebovali 700 let. Med pomembnejše statistike sodijo [3]:

- 5 najpogostejših besed je bilo:
  - the, ki je predstavljala 7,14 % celotnega korpusa,
  - of 4,16 %,
  - and 3,04 %,
  - to 2,60 %,
  - in 2,27 %.
- 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.
- Najpogosteje uporabljena črka je črka E. Deleži petih najpogostejših črk so pričakovano[1]:
  - E 12,49 % ,
  - T 9,28 %,
  - A 8,04 %,
  - O 7,64 %,
  - I 7,57 %.

Za kompleksnejše procesiranje slovenskega jezika je možno uporabiti odprtokodno orodje NooJ [4]. Orodje omogoča analizo tekstov in korpusov v več kot 20 jezikih na pravopisnem, leksikalnem, oblikoslovnem, sintaktičnem in

[1]V angleškem jeziku obstaja brezpomenska fraza "Etaoin shrdlu", ki predstavlja zaporedje 12 najpogosteje uporabljenih črk v angleških tekstih razporejenih po padajoči frekvenci uporabe. Izvira še iz časov uporabe stavnih strojev. Leta 1966 je Irvin Fang analiziral časopise in TV scenarije ter prišel do malenkost spremenjenega zaporedja "etaoni rshdlc", medtem ko je Norvig kot rezultat dobil "etaoin srhldcu".

semantičnem nivoju. Uporaba orodja s slovenskim jezikom je predstavljena v članku "Procesiranje slovenskega jezika v razvojnem okolju NooJ", v katerem avtorica poleg pogostejših uporab orodja kot možnost omeni tudi oblikovanje "kompleksnih 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" [5].

Veliki korpusi lahko služijo kot baza besed in njihovih lem[2], s čimer se olajša opravilo lematizacije[3] besed. V primeru naloge lematiziranja novih, še neoznačenih besed, se je naloge moč lotiti s strojnim učenjem, pri čemer že znane in lematizirane besede uporabimo kot učno množico. Avtomatska lematizacija besed je bila opisana in preizkušena v članku " Learning to lemmatise Slovene words" avtorjev S. Dzeroski in T. Erjavec [6]. Avtorja sta uporabila obstoječ in označen korpus za učenje in validiranje naučenih oblikoslovnih pravil. Za učenje in testiranje sta uporabila označeno verzijo Orwellove knjige 1984, ki v slovenskem prevodu vsebuje 90.792 besed. Dosežena točnost je bila 79,2 %.

Korpusi načeloma niso označeni z oznakami, ki bi pripomogle k razumevanju ali obdelavi na semantični ravni, so pa pomembni pri gradnji WordNet-ov. WordNet je semantični leksikon, v katerem so samostalniki, glagoli, prislovi in pridevniki združeni v skupine elementov, ki so med seboj semantično ekvivalentni [4] [7]. Korpus FIDA sta T. Erjavec in D. Fišer uporabila pri gradnji slovenskega WordNeta; posamezne besede sta uvrstila v 6 kategorij glede na frekvenco njihove pojavitve v korpusu, v pomoč pri iskanju sopomenk pa so bile tudi oznake lem besed [8].

Več poročil o statistični obdelavi korpusov in uporabi tako dobljenih rezultatov najdemo v tuji literaturi.

[2]Lema je osnovna oblika besede. Besede *pišem, pišeš, piševa, pišejo, ...* so različne oblike leme *pisati*

[3]Lematizacija je postopek določanja leme besedam.

[4]Primer semantično ekvivalentne skupine: varovanje, varstvo, zaščita [**?**].

Uporabljena metoda	**Stopnja povezanosti (r)**
Človeška presoja	0,8848
Informacijska vsebina	0,7941
Razdalja med koncepti	0,6004
Hibridni pristop	0,8282

Tabela 2.1: Primerjava rezultatov ugotavljanja semantične podobnosti (J. Jiang, D. Conrath 1997).

Statistična analiza je bila uspešno uporabljena pri ugotavljanju semantične podobnosti med besedami in koncepti v angleškem jeziku. Ena izmed možnosti uporabe korpusov pri določanju sopomenk je določanje povezav med besedami glede na to, kako pogosto se v korpusih pojavljajo skupaj. J. Jiang in D. Conrath [9] sta razvila hibridni pristop za izračun podobnosti med besedami in koncepti, s katerim sta združila pristop z izračunom informacijske vsebine skupine sopomenk [5] in razdalje med koncepti v hierarhični taksonomiji. Avtorja sta združila podatke iz Brownovega korpusa [10] in WordNet-a [7] ter izračunala stopnjo povezanosti med 30 pari samostalnikov. Rezultate sta primerjala z rezultati obeh omenjenih metod in z rezultati, ki so jih za iste pare besed določili ljudje v eni od preteklih raziskav (Resnik 1995). Rezultati so prikazani v tabeli 2.1.

V članku, poimenovanem "How to Avoid Burning Ducks" oz. prevedeno v slovenščino "Kako se izogniti gorečim racam" [6], sta avtorja primerjala dve uveljavljeni metodi razstavljanja tvorjenk - jezikovni pristop in korpusni pristop - in za primerjavo preizkusila hibridni pristop, ki je vključeval tudi rezultate statistične analize korpusov. Jezikovni pristop se je izkazal kot natančnejši pri razstavljanju tvorjenk na posamezne dele, korpusni pa kot

[5] Vrednost informacijske vsebine skupine sopomenk je izračunana na podlagi verjetnosti pojavitve določene skupine v velikih korpusih.

[6] Celoten naslov: "How to Avoid Burning Ducks: Combining Linguistic Analysis and Corpus Statistics for German Compound Processing" oz. prevedeno v slovenščino: "Kako se izogniti gorečim racam: združevanje jezikovne analize in korpusne statistike za obdelavo nemških tvorjenk".

boljši pri avtomatskem prevajanju besednih zvez iz nemščine v angleščino. Problem je še posebej izrazit, ker besedotvorje v nemškem jeziku temelji na ustvarjanju tvorjenk. Tako sestavljene nove besede niso enostavno prevedljive v druge jezike, ki imajo drugačna pravila pri oblikovanju in uporabi besed. Korpus se pri prevajanju iz nemščine uporablja tako, da se tvorjenka razdeli na dve besedi, ki ju je moč najti ločeni v korpusu[7]. Korpusni pristop je za delitev uporabljal več pravil, med drugim je bila določena najkrajša veljavna dolžina besed, črke, ki so se pri delitvi opustile, črke, med katerimi delitev ni bila dovoljena itd. Hibridni pristop je med drugim uporabil frekvence posameznih besed v korpusu, tako da so se tvorjenke delile na besede, ki so se v korpusu pojavile najpogosteje. Korpusni pristop je v raziskavi dosegel 78,73 % natančnost, medtem ko je hibridni pristop za isto besedilo dosegel 92,29 %. [11]

M. Lauer [12] je pokazal, da lahko tudi preposte korpusne statistike dodajo informacije o sintaksi samostalniških fraz. V prvem delu članka je predstavil štiri obstoječe algoritme, ki za sintaktično analizo samostalniških fraz uporabljajo korpuse, v preostanku pa je delovanje algoritmov preizkusil na zbirki 244 izrazov. Trije algoritmi temeljijo na modelu sosednjosti (adjacency model), medtem ko zadnji temelji na modelu odvisnosti (dependency model). V vseh primerih se je za statistično značilno boljšega izkazal slednji.

V primeru treh samostalnikov s_1, s_2 in s_3 njihova obravnava sledi sledečim pravilom:

- če ni semantično sprejemljiv nobeden izmed parov $[s_1\ s_2]$ ali $[s_2\ s_3]$, potem zgradi alternativno strukturo;
- če je $[s_2\ s_3]$ semantično boljši od $[s_1\ s_2]$, izberi $[s_2\ s_3]$;
- sicer izberi $[s_1\ s_2]$.

[7]Primer tvorjenke, ki nima ekvivalentne besede v angleščini, je beseda “schadenfreude” (škodoželjnost). Sestavljena je iz dveh besed: “schaden” (škoditi) in “freude” (veselje). Za razumljiv prevod v angleščino je potrebno tvorjenko razstaviti na ti dve besedi in avtomatsko prevesti vsako posebej, s čimer dobimo angleški prevod: “malicious joy” ali “harm joy”.

Primer dveh samostalniških fraz v angleškem jeziku sestavljenih samo iz samostalnikov, kjer zaporedje obravnave vpliva na pomen izraza:

- [strawberry [ice cream]]
- [[real estate] tycoon]

V članku opisani algoritmi za sintaktično analizo tvorjenk:

- Najenostavnejši algoritem poskusi drugje v korpusu poiskati vsako od dveh možnih kombinacij. Tista, ki jo najde, je izbrana kot boljša opcija [13].
- Algoritem Libermana in Sproata [14] uporablja podatek o frekvenci besed v tvorjenki. Pristop primerja medsebojno informacijo med paroma sosednjih besed in izbere tistega z večjo vrednostjo.
- Resnik [15] izbira izbirno asociacijo (selectional association) med dvema predikatoma, pri čemer je beseda definirana na podlagi prispevka besede k pogojni entropiji predikata. Asociacija znotraj posameznega para je izračunana kot maksimum izbirne asociacije vseh možnih načinov obravnave para kot predikata.
- Lauer [12] namesto modela sosednjosti uporabi model odvisnosti, pri čemer je stopnja sprejemljivosti pridobljena iz statistične metrike celotnega korpusa - mere podobne medsebojni informaciji.
  - določi sprejemljivost struktur $[s_1\ s_2]$ in $[s_1\ s_3]$,
  - če je sprejemljivejši slednji, izberi $[s_2\ s_3]$,
  - sicer izberi $[s_1\ s_2]$.

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 [16].

Pri identifikaciji pravilnega pomena besede v povezavi z WordNet-om je iz korpusov moč izračunati relativne verjetnosti pomenov besed in jih v

odločitvenem procesu uporabiti kot apriorno verjetnost v Bayesovem modelu. Rezultati so se pri nekaterih besedah približali do 1 %-2 % točnosti človeške identifikacije, vendar je točnost padla pri besedah, ki imajo lahko več semantično dovolj različnih pomenov [17].

Schiffman in sod. [18] so korpusno statistiko 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 iz sodnih zapisnikov predsednika Clintona na obravnavi po razkritju njegove afere. V zapisniku je bilo omenjeno veliko število ljudi in njihovih aktivnosti.

Y. Yang in J. Wilbur [19] sta s pomočjo korpusne statistike izračunala vrednost pomembnosti posameznih besed in s tem skrčila seznam besed uporabljenih pri opisih namenjeni 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.

2.2 Predstavitev korpusov

Besedilni korpusi so velike zbirke besedil zgrajene po vnaprej določenih merilih. Zajeti so iz različnih virov znotraj določenega obdobja in so lahko eno ali večjezični. Najobsežnejši in najbolj reprezentativni za določen jezik so referenčni korpusi, kamor spada tudi slovenski korpus Gigafida [20], ki smo ga uporabili pri analizi v tem diplomskem delu.

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 korpusov, kot npr. leme besed in oblikoskladenjske specifikacije.

Gigafida je referenčni korpus, ki vsebuje skoraj 1,2 milijarde besed[8], zajetih iz besedil, ki so izšla v obdobju med leti 1990 do 2011 ter so izšla v

[8] 1.187.002.502 besed

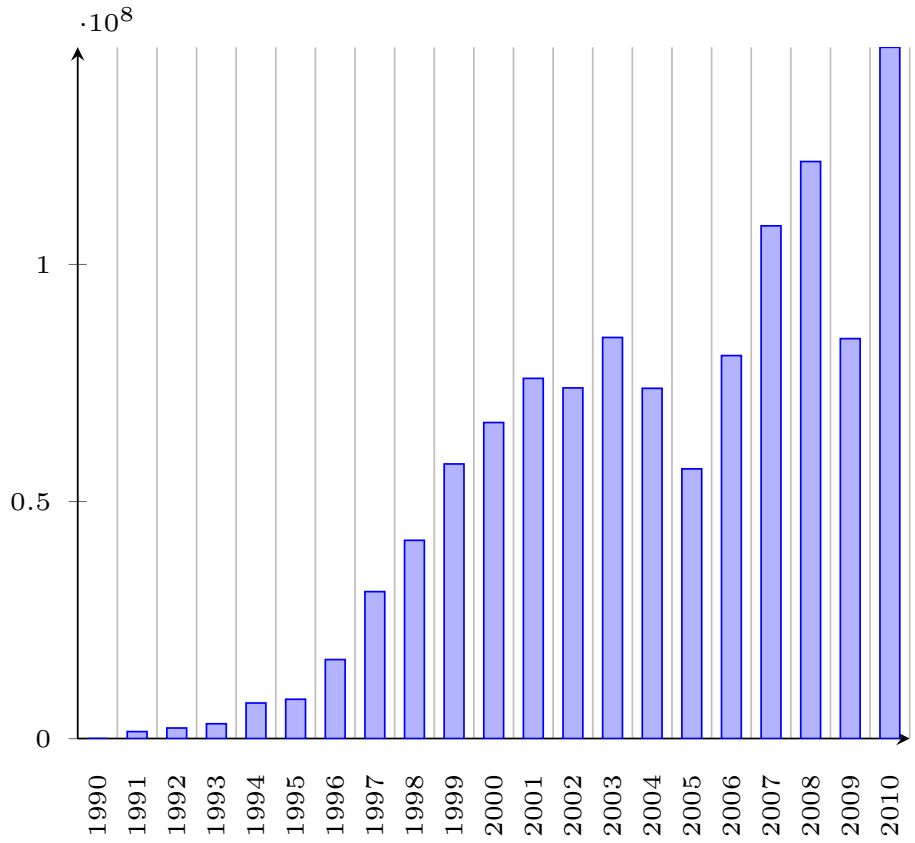
Taksonomija	**Število besed**	**Delež besed**
tisk	1.001.244.035	84,35 %
knjižno	71.356.531	6,26 %
leposlovno	23.969.196	2,02 %
strokovno	50.387.335	4,24 %
periodično	918.936.054	77,42 %
časopis	663.664.965	55,91 %
revija	255.271.089	21,51 %
drugo	7.951.450	0,67 %
internet	185.758.467	15,65 %
SKUPAJ	1.187.002.502	100 %

Tabela 2.2: Taksonomija korpusa Gigafida [20].

tiskani obliki ali na internetu [21]. Število besed glede na leto izvora je prikazano na sliki 2.1, medtem ko je točna taksonomija Gigafide predstavljena v tabeli 2.2. Korpus je zapisan v formatu XML TEI P5 [9], je lematiziran in oblikoskladenjsko označen. Gigafida vključuje referenčni korpus FidaPLUS iz leta 2006 (621 milijonov besed), ki dalje vključuje prvi slovenski referenčni korpus FIDA (1997 - 2000). Število besed glede na leto izdaje je prikazano na sliki 2.1.

Standard za jezik XML formalno določa zapis in strukturo podatkov, ki jih s tem jezikom opišemo. Še posebej primeren je za shranjevanje in predstavitev strukturiranih podatkov, za katere omogoča enostavno berljivost, tako za ljudi kot tudi za računalnike. Dve glavni komponenti XML zapisa sta etiketa (tag) in atribut (attribute). Osnovni podatkovni model je drevesni,

[9]Format XML z dodatnimi specifikacijami namenjenimi strukturirani predstavitvi besedil.



Slika 2.1: Število besed dodanih v korpus Gigafida po letih [20].

```
<?xml version="1.0" encoding="UTF-8"?>
<clanek>
    <stavek vir="ccKRES: Delo (2004-01-20)">Sokol ima svoje
        območje točno označeno in lahko zajema več
        tisoč hektarjev.</stavek>
</clanek>
```

Slika 2.2: Primer veljavne XML datoteke, v kateri je zapisan enostaven stavek. Stavku je dodan tudi atribut vir.

pri čemer je možno poljubno globoko gnezdenje etiket s ciljem doseganja smiselne hierarhije podatkov. Vsaka etiketa je sestavljena iz začetne in končne oznake, med katerima se lahko nahaja podatek, nova etiketa ali več etiket, lahko pa tudi nič - v takšnem primeru etiketa ne nosi posebne informacije, ali pa je ta shranjena v njenih atributih. Sam standard ne določa imen posameznih elementov, zgolj strukturo datotek, pravila gnezdenja, uporabo atributov in pripadajoče vrednosti. Primer enostavnejšega zapisa besedila v formatu XML je prikazan na sliki 2.2. Prikazana je vrstica s podatki o verziji uporabljenega standarda XML in kodiranju (encoding). Sledita ji dve etiketi, ki sta gnezdeni: članek in stavek. Posamezen članek lahko vsebuje tudi več stavkov, prikazan je eden.

Ena izmed uporabnih lastnosti tega formata je enostavna razširljivost standarda. Poleg osnovnih specifikacij je moč osnovni standard prilagoditi specifičnim potrebam z dodatkom strožjih omejitev strukture in označevanja podatkov. Zaradi do sedaj omenjenih razlogov je XML še posebej primeren za shranjevanje označenih jezikovnih korpusov. V ta namen je konzorcij TEI (Text Encoding Initiative) izdal priporočila za označevanje besedil, katerih glavni namen je uporaba v raziskovalne namene. Priporočila obsegajo standarde za zapis različnih zvrsti besedil in za jezikoslovne korpuse. Prva dva nivoja etiket dokumentov, pripravljenih po standardu TEI, sta :

- informacija o uporabljeni različici standarda XML in kodiranju
- korenski element TEI,
  - kolofon
  - besedilo.

V kolofonu so shranjeni metapodatki o besedilu. Vanj se shranjujejo bibliografski podatki, informacije, kako je dokument strukturiran, 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) in ločila (c) ter presledke. Vsaka beseda poleg vrednosti - besede iz obravnavanega besedila - vsebuje še atributa lemma in msd. V atributu lemma je shranjena lema[10] besede, medtem ko je v atributu msd shranjena oblikoskladenjska oznaka besede, ki sledi specifikacijam MULTEXT-East različica 4.0. Oznake za slovenski jezik so bile razvite v okviru projekta JOS in zajemajo več kot 1900 oznak, ki so lahko izražene v slovenščini ali angleščini.

Primer stavka iz prejšnjega primera (slika 2.2), zapisan v formatu TEI P5, je prikazan na sliki 2.3. V prvem primeru je bilo moč razbrati zgolj osnovno strukturo članek - stavek, medtem ko drugi vsebuje točno označene etikete in atribute z vsemi pripadajočimi metapodatki. V kolofonu je naveden publicist, datum, ko je bilo delo izdano in avtor (v tem primeru neznan). Element besedilo je sestavljen iz nazorno označenih elementov: odstavkov, stavkov, besed, presledkov in ločil. Vsaki besedi je dodana lema in oblikoskladenjske oznake. Oblikoskladenjske oznake so natančno določene in so za primer samostalnika prikazane v tabeli 2.3.

Oznake Somei prve besede - Sokol - nam povejo, da gre za:

- S - samostalnik,

[10] geselska iztočnica

pozicija	**atribut**	**vrednost**	**koda**
0	besedna vrsta	samostalnik	S
1	vrsta	občno ime	o
		lastno ime	l
2	spol	moški	m
		ženski	z
		srednji	s
3	število	ednina	e
		dvojina	d
		množina	m
4	sklon	imenovalnik	i
		rodilnik	r
		dajalnik	d
		tožilnik	t
		mestnik	m
		orodnik	o
5	živost	ne	n
		da	d

Tabela 2.3: Tabela oblikoskladenjskih oznak za samostalnik [22].

- o - občno ime,
- m - moški spol,
- e - ednina,
- i - imenovalnik.

Korpus Gigafida sestavlja 39.427 XML datotek zgrajenih v opisanem formatu.

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0" xml:lang="sl">
   <teiHeader>
      <fileDesc>
         <sourceDesc>
            <bibl>
               <author n="???">neznani novinar</author>
               <date>2004-01-20</date>
               <publisher n="Delo">Delo</publisher>
            </bibl>
         </sourceDesc>
      </fileDesc>
   </teiHeader>
   <text xml:lang="sl">
      <body>
         <p>
            <s>
               <w msd="Somei" lemma="sokol">Sokol</w>
               <S/>
               <w msd="Ggnste-n" lemma="imeti">ima</w>
               <S/>
               <w msd="Zp-set" lemma="svoj">svoje</w>
               <S/>
               <w msd="Soset" lemma="območje">območje</w>
               <S/>
               <w msd="Rsn" lemma="točno">točno</w>
               <S/>
               <w msd="Rsn" lemma="označeno">označeno</w>
               <S/>
               <w msd="Vp" lemma="in">in</w>
               <S/>
               <w msd="Rsn" lemma="lahko">lahko</w>
               <S/>
               <w msd="Ggnste" lemma="zajemati">zajema</w>
               <S/>
               <w msd="Rsr" lemma="več">več</w>
               <S/>
               <w msd="Kbg-mt" lemma="tisoč">tisoč</w>
               <S/>
               <w msd="Sommr" lemma="hektar">hektarjev</w>
               <c>.</c>
               <S/>
            </s>
         </p>
      </body>
   </text>
</TEI>
```

Slika 2.3: Primer stavka iz slike 2.2 zapisanega v formatu TEI P5. Za poenostavljen prikaz je bilo odstranjenih nekaj elementov z metapodatki o licenci (Creative Commons), taksonomiji (opisana v tabeli 2.2) itd.

Poglavje 3

Algoritmi

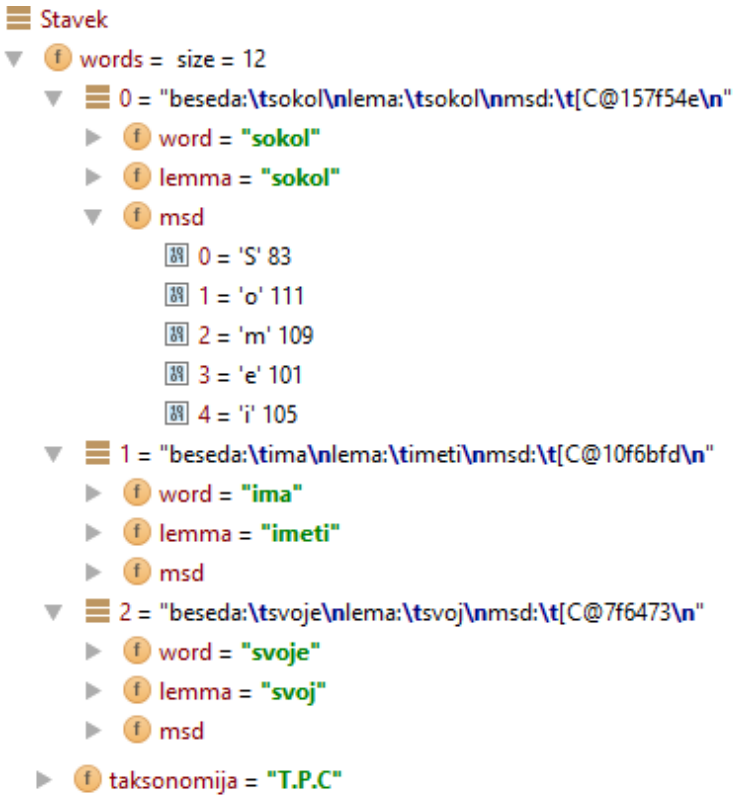
V tem poglavju predstavimo glavne probleme, s katerimi smo se soočili, opišemo pristope, s katerimi smo se lotili reševanja problemov, in podamo podatke o časovni ali prostorski zahtevnosti rešitve, kjer je to smiselno.

3.1 Predstavitev podatkov

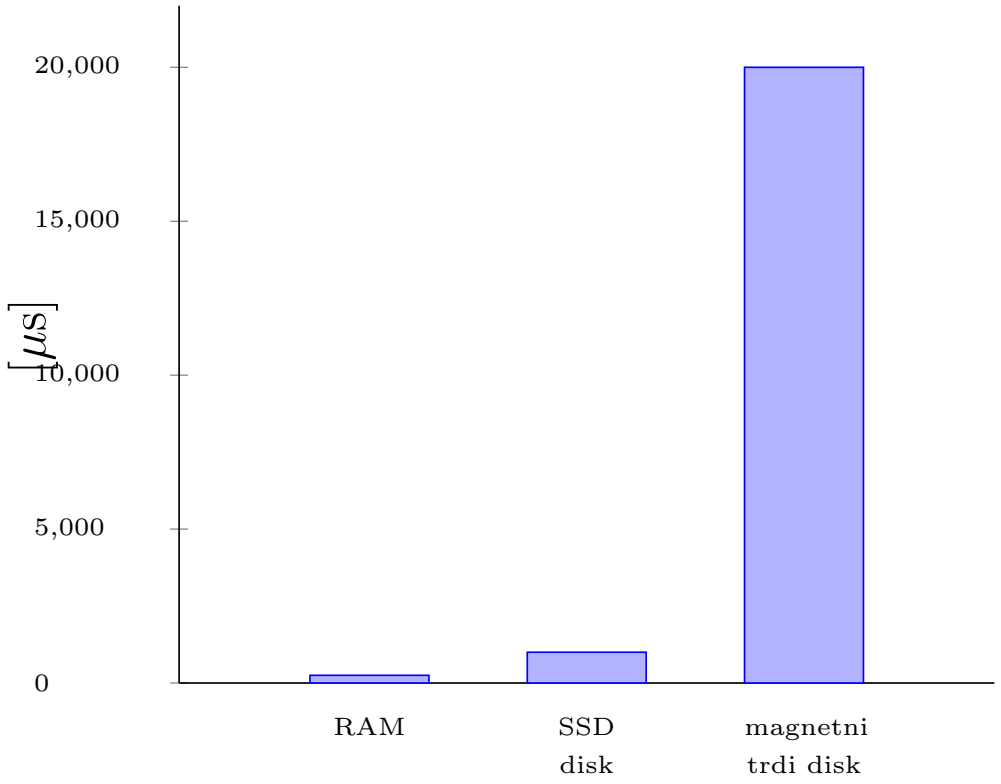
Format TEI P5 XML, v katerem je zapisan korpus Gigafida, je primeren za shrambo jezikovnih korpusov in pripadajočih metapodatkov. Za enostavnejše procesiranje je potrebno podatke iz takšnega formata preneseti v podatkovno strukturo, s katero je moč enostavno upravljati v programski kodi. Za razvoj programa smo uporabili programski jezik java, zato je bilo smiselno, da smo izkoristili objektno naravo jave. Ustvarili smo dva tipa objektov: za stavke in za besede, pri čemer je objekt *stavek* sestavljen iz množice objektov *besed* in atributa za podatek o taksonomiji stavka. Objekt *beseda* je sestavljen iz znakovnih nizov za besedo in lemo besede ter tabele znakov, v katero se zapiše koda MSD. Posamezne etikete XML zapisa so tako analogne posameznim objektom v Javi, atributi etiket pa atributom objekta. Programska koda odgovorna za kreiranje objektov in atributov je prikazana na sliki 3.1, primer takšne predstavitve stavka pa na sliki 3.2.

```
public class Sentence
{
	private List<Word> words; // ArrayList
	private String taxonomy;

	public Sentence(List<Word> words, String taxonomy)
	{
		this.words = words;
		this.taxonomy = taxonomy;
	}
}

public class Word
{
	private String word;
	private String lemma;
	private char[] msd;

	public Word(String word, String lemma, String msd)
	{
		this.lemma = lemma;
		this.msd = msd.toCharArray();

		// normaliziramo besede - veliko zacetnico
		// ohranimo samo za lastna imena
		if (!(this.msd[0] == 'S'  this.msd[1] == 'l'))
			this.word = word.toLowerCase();
		else
			this.word = word;
	}
}
```

Slika 3.1: Razreda objektov tipa stavek (Sentence) in beseda (Word). Na sliki niso prikazane "get" in "set" metode.



Slika 3.2: Prve tri besede stavka iz korpusa po pretvorbi v objekte.



Slika 3.3: Čas potreben za prenos 1MB podatkov[23].

3.2 Velikost korpusov

Najočitnejši problem pri procesiranju velikih jezikovnih korpusov kot je Gigafida je, da jih zaradi njihove velikosti ni mogoče hraniti v pomnilniku. Korpus Gigafida je velik 83,5 GB, medtem ko ima povprečen računalnik med 4 GB in 8 GB pomnilnika. Z velikostjo je povezano tudi počasno branje podatkov iz diska - vsaj v primerjavi s hitrostjo procesorja in pomnilnika. V nasprotju z računanjem statistik, ki jih na večjedrnih in večnitnih procesorjih lahko paraleliziramo, branje s trdega diska ostaja ozko grlo. Razlika v hitrosti branja različnih medijev je prikazana na sliki 3.3. Zaradi tega program veliko časa porabi za branje podatkov.

Za rešitev tega problema smo uporabili paketno obdelavo vhodnih podat-

kov:

1. Program prebira vhodne podatke, dokler ne prebere določenega števila stavkov, nakar začasno prekine branje. To število je določeno glede na količino pomnilnika v računalniku.

2. Na prebranih podatkih izračuna vnaprej določeno statistiko.

3. Prebrani podatki se brišejo, s tem se sprosti prostor za nov paket. Program nadaljuje z branjem novih podatkov - vrnemo se na točko 1.

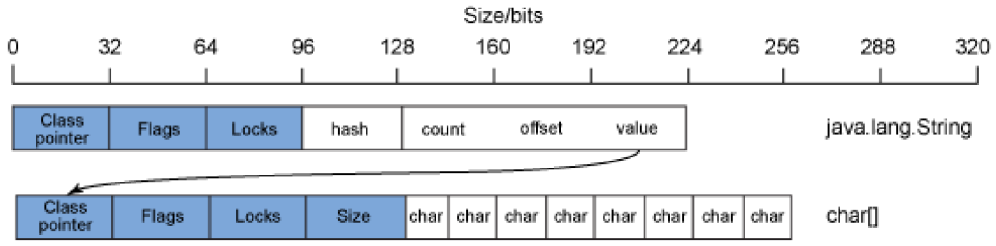
Postopek je natančneje prikazan v algoritmu 1. Klic Fork-Join bomo predstavili v naslednjem razdelku pri algoritmu 2. Časovna zahtevnost takšnega pristopa je linearna - $\mathcal{O}(n)$, kjer n predstavlja velikost vhodnih podatkov.

Algoritem 1 Paketno procesiranje korpusa

```
1: while v korpusu so še neprebrani stavki do
2:     subcorpus ← stavek
3:     if subcorpus.size ≥ limit then
4:         FORK-JOIN(subcorpus)
5:         subcorpus = ∅
6:     end if
7: end while
```

Tudi tako optimiziran pristop hitro naleti na omejitev: vsak objekt v javi ima določeno režijo (overhead). Posamezen objekt tipa besede, 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



Slika 3.4: Zgradba 64-bitnega String objekta [24].

statistik izbrali HashMap[1]. Režija HashMap podatkovne strukture je 360 kB za HashMap z 10.000 elementi, kar ne vključuje velikosti in režije posameznih vnosov. Tipi vnosov v HashMap-u so v večini primerov "String: AtomicInteger"[2]. Za primer si poglejmo sestavo 64-bitnega String objekta, ki je prikazan na sliki 3.4. Takšen objekt vsebujejo več polj za metapodatke. To pomeni, da String, v katerega shranimo 8 znakov (128 bitov), potrebuje 256 bitov za zapis teh znakov v polje črk in 224 bitov za metapodatke. Tako za reprezentacijo 128 bitov (16 bajtov) potrebujemo 480 bitov (60 bajtov) - režijsko razmerje je 3,75 :1 [24]. Spomnimo, da potrebujemo več 10 milijonov takšnih objektov, ki jih je potrebno sočasno hraniti v pomnilniku. Zaradi tega lahko na računalniku s 16 GB pomnilnika računamo samo eno statistiko naenkrat, za nekatere (bigrami in trigrami) pa tudi to ni dovolj pomnilnika. Porabo pomnilnika smo delno optimizirali z uporabo alternativnega Garbage Collectorja: G1 Collector (Garbage First Collector), ki je namenjen programom, ki zasedejo veliko pomnilnika (več kot 6 GB) in je bolj učinkovit pri čiščenju objektov, ki niso več potrebni, iz pomnilnika, na račun počasnejšega izvajanja programa.

[1]HashMap je podatkovna struktura, v kateri je vsak vnos shranjen v obliki "ključ: vrednost". HashMap ne more vsebovati podvojenih vnosov 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 $\mathcal{O}(1)$.

[2]AtomicInteger je tip celega števila (Integer), ki omogoča sočasen dostop in spreminjanje iz več niti programa brez napak, do katerih prihaja pri nesinhroniziranih objektih.

3.3 Paralelizacija

Vsi moderni procesorji imajo več jeder in podpirajo večnitno procesiranje. S poganjanjem algoritma na več jedrih oz. nitih se ustrezno skrajša čas izvajanja, kar je še posebej koristno pri algoritmih, ki so sestavljeni iz velikega števila ukazov. 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 milijarda takšnih operacij za vsako statistiko, ki jo želimo izračunati.

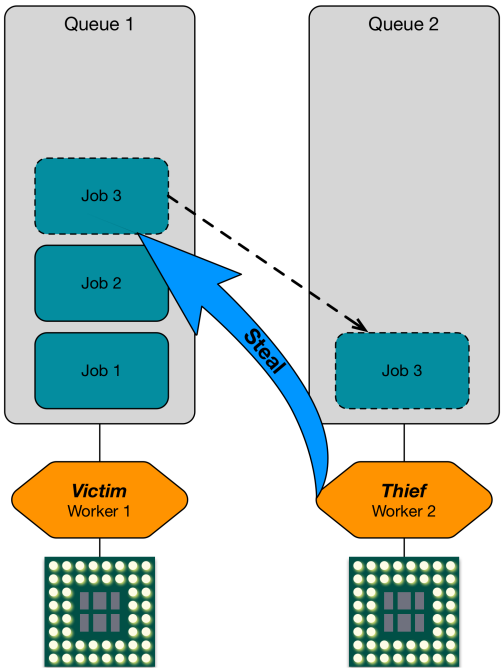
Pri reševanju tega problema smo pogledali več opcij: tokove (streams) v Javi 8, Fork-Join, Map Reduce in Akka. Večina obstoječih rešitev je namenjena obdelavi podatkov, ki se nahajajo na več ločenih sistemih (Map Reduce) in s poganjanjem na samo enem sistemu ne morejo izkoristiti vgrajenih optimizacij [25, 26]. Odločili smo se za implementacijo Fork/Join [27, 28, 29], ker so se tokovi v Javi 8 izkazali za nezanesljive in so pogosto tudi nekajkrat počasnejši kot Fork-Join [30, 31].

Algoritem 2 Fork-Join algoritem.

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

Fork-Join je v osnovi paralelna verzija principa deli in vladaj[3]. Osnovno

[3]Deli in vladaj je metoda načrtovanja algoritmov, 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.



Slika 3.5: Shema principa kraje opravil med nitmi [32].

delovanje je prikazano v algoritmu 2. Samih podatkovnih struktur v tem koraku nikoli ne spreminjamo ali prepisujemo v nove, manjše, temveč je podproblem definiran kot pogled na omejen del celotne podatkovne strukture. V Javi je Fork-Join na voljo od verzije 6 dalje, pri čemer je bil v vsaki novejši verziji Jave delno izboljšan. V Javi 8 je bil dodan ForkJoinPool, ki omogoča Fork-Join opravilom kreiranje podopravil in čakanje na dokončanje posameznega opravila. S tem se omogoči kraja opravil (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, saj nobena od niti ne miruje dolgo časa. Princip kraje opravil je prikazan na sliki 3.5. Medtem ko ima nit 1 (Worker 1) v svoji vrsti še 3 opravila, je nit 2 (Worker 2) že zaključila z vsemi svojimi. Namesto da bi čakala, pogleda v vrsto prve niti, vidi, da ima čakajoča opravila in vzame opravilo s konca vrste niti 1. S tem sta obe niti ves čas zaposleni, delo je enakomernejše porazdeljeno med

niti in celotna naloga je opravljena hitreje in učinkoviteje.

3.4 Računanje statistik

V večini primerov je bilo potrebno izračunati frekvenco določenega podatka: besede, leme ali črke v odvisnosti od nekega pogoja kot je npr. frekvenca črk v tiskanih medijih ali frekvenca dolžin besed v tiskanih medijih v primerjavi z internetom. V takšnih primerih smo uporabili podatkovno strukturo HashMap, ki smo jo opisali v razdelku 2. Postopek, ki smo ga uporabili, je prikazan v algoritmu 3. V prvih dveh zankah se sprehodimo skozi vse stavke (stavki so samo zbirke besed, ki sodijo skupaj) in besede v njih, kar je $\mathcal{O}(n)$, kjer n predstavlja število besed v podkorpusu[4]. Preverjanje pogojev "if" in ali je element že v HashMap strukturi, dostop do elementa ter spreminjanje vrednosti se zgodi v konstantnem času - $\mathcal{O}(1)$. Časovna zahtevnost opisanega algoritma je tako $\mathcal{O}(n)$.

Algoritme, ki statistiko izračunajo na opisan način smo razdelili v smiselne 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 omenjenih razredih kličemo z imenom metode in tremi parametri:

- podkorpusom oz. pogledom na del celotnega korpusa,

[4]Algoritem 3 kličemo iz vrstice 3 algoritma 2, zato je vhod majhen del korpusa.

- referenco na HashMap, v katerega zapisujemo frekvence - ta ima v primeru računanja distribucije dolžin besed kot ključ objekt tipa Integer,
- omejitvijo - to je lahko določena oznaka v taksonomiji, oblikoskladenjska oznaka iz tabele oznak JOS ali podatek, če določeno statistiko računamo za besedo ali njeno lemo ali poljubna kombinacija omenjenih kriterijev.

Algoritem 3 Algoritem za računanje frekvenc.

```
1: procedure IZRAČUNAJ FREKVENCO BESED(subcorpus)
2:     frekvence = ∅                ▷ inicializacija frekvenc besed v strukturi
3:     for stavek ∈ subcorpus do
4:         for beseda ∈ stavek do
5:             if beseda ∈ frekvence then      ▷ če je beseda že v HashMap
6:                 frekvence[beseda] + +        ▷ povečaj št. pojavitev za 1
7:             else
8:                 frekvence ← beseda                ▷ sicer dodaj besedo
9:                 frekvence[beseda] = 1        ▷ nastavi št. pojavitev na 1
10:            end if
11:        end for
12:    end for
13: end procedure
```

Ob zaključku izračunane frekvence uredimo po padajočem zaporedju, za kar uporabimo TimSort, ki ima v najslabšem primeru časovno zahtevnost $O(n \log n)$ [33]. Skupno časovno zahtevnost celotnega programa dobimo s seštevanjem časovnih zahtevnosti posameznih delov:

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

- TimSort: $O(n \log n)$

Pri tem je "k" pri koraku Fork-Join zanemarljivo majhna številka, ki predstavlja majhno število korakov delitve večjega problema na manjše. Končna časovna zahtevnost celotnega programa je tako enaka $O(n \log n)$.

Poglavje 4

Rezultati

Za začetek smo želeli primerjati slovenščino z angleščino kot sta jo predstavila Mayzner in Norvig [2, 3]. Deset najpogostejših besed v korpusu Gigafida je prikazanih v tabeli 4.1.

beseda	**število pojavitev**	**delež korpusa**
je	36.840.586	3,11 %
v	31.087.684	2,63 %
in	30.228.292	2,55 %
na	18.728.379	1,58 %
se	16.042.376	1,36 %
za	15.469.509	1,31 %
da	14.622.806	1,24 %
so	12.346.876	1,04 %
ki	12.080.996	1,02 %
pa	11.157.362	0,94 %

Tabela 4.1: Najpogostejše besede v korpusu Gigafida.

Tako kot v angleščini tudi v slovenščini med najpogostejšimi besedami ne najdemo samostalnikov ali glagolov. Zanimivo je, da distribucija besed ne sledi Zipfovemu zakonu. Poimenovan po ameriškemu jezikoslovcu Georgu Kingsleyu Zipfu (1902-1950), ki ga je populiziral[1], zakon opisuje razmerje med distribucijo pogostosti uporabe besede in njenim mestom v ranžirni vrsti besed [34]. Najpogosteje uporabljena beseda se bo tako v jeziku pojavljala 2-krat pogosteje kot druga najpogosteje uporabljena beseda, 3-krat pogosteje kot tretja itd.

Deset najpogostejših črk v celotnem korpusu in po taksonomijah za leposlovje in internet je prikazanih v tabeli 4.2. Vidimo, da med njimi obstajajo razlike - v leposlovju je tretja najpogostejša črka i, medtem ko je v ostalih dveh kategorijah to črka o. Omenili smo, da v angleščini obstaja brezpomenska fraza "etaoin shrdlu", ki predstavlja zaporedje 12 najpogosteje uporabljenih črk v angleških tekstih. Slovenski ekvivalent bi bila fraza "aeoinr stlvjk".

Distribucija dolžin besed je prikazana v tabeli 4.3. Tudi tu so opazne razlike med leposlovnimi besedili in besedili z interneta. Internetna besedila pogosteje uporabljajo daljše besede kot leposlovna.

Za besede v korpusu smo izračunali frekvence za besedne vrste. V tabeli 4.4 je prikazanih 10 najpogostejših samostalnikov, pridevnikov in glagolov. Zanimivo je, da 5 od 10 samostalnikov opisuje čas in da med preostalimi 3 opisujejo delo. Seznam najpogostejših pridevnikov je patriotsko obarvan, saj se verzije besede slovenski pojavijo 3-krat. 4 besede so različice besede nov. Pri glagolih je daleč najpogostejši "je", ki predstavlja več kot tretjino glagolov. Za primerjavo smo izračunali frekvence lem. Prikazane so v tabeli 4.5. Dobljeni rezultati se skladajo z najpogostejšimi besedami v tabeli 4.4. Leme treh najpogostejših samostalnikov opisujejo čas, prva in tretja najpogostejša lema pridevnikov sta lemi "nov" in "velik", med glagoli pa je lema "biti" kar 20-krat pogostejša kot naslednja najpogostejša lema pridevnika.

[1]Pred Zipfom sta vzorec opazila francoski stenograf Jean-Baptiste Estoup (1868-1950) in nemški fizki Felix Auerbach (1856-1933).

celoten korpus		leposlovje		internet	
črka	**frekvenca**	**črka**	**frekvenca**	**črka**	**frekvenca**
a	10,13 %	a	10,83 %	a	10,52 %
e	9,8 %	e	10,71 %	e	9,94 %
o	9,07 %	i	8,85 %	o	9,15 %
i	8,78 %	o	8,83 %	i	8,66 %
n	6,75 %	n	6,32 %	n	6,78 %
r	5,34 %	l	5,35 %	r	5,26 %
s	4,59 %	s	5,1 %	s	4,63 %
t	4,53 %	r	4,87 %	t	4,58 %
l	4,44 %	j	4,55 %	l	4,37 %
v	4,15 %	t	4,22 %	v	4,16 %

Tabela 4.2: Najpogostejše črke v korpusu Gigafida.

Primerjali smo najpogostejše besede v celotnem korpusu, leposlovju in na internetu. Rezultati prikazani v tabeli 4.6 kažejo, da v tem pogledu ni opazne razlike med posameznimi kategorijami. Razlike so vidne predvsem v različnih vrednostih frekvenc.

Analizirali smo tudi pare - bigrame - besed in lem. Rezultati so prikazani v tabeli 4.7. Rezultati se skladajo s tabelo najpogostejših besed, ki so pogosto predstavljane tudi kot del bigramov.

celoten korpus		**leposlovje**		**internet**	
dolžina	**frekvenca**	**dolžina**	**frekvenca**	**dolžina**	**frekvenca**
2	22,63 %	2	27,81 %	2	24,03 %
5	11,1 %	5	12,84 %	5	10,89 %
6	10,75 %	4	11,17 %	6	10,45 %
4	9,52 %	6	10,74 %	4	9,43 %
7	9,39 %	3	10,28 %	7	9,19 %
8	8,34 %	7	7,94 %	3	8,58 %
3	8,17 %	8	6,33 %	8	8 %
9	6,18 %	1	4,19 %	9	5,98 %
1	5,18 %	9	4,04 %	1	4,99 %
10	3,95 %	10	2,4 %	10	3,81 %

Tabela 4.3: Frekvenca dolžin besed v korpusu Gigafida.

samostalnik		pridevnik		glagol	
beseda	**frekvenca**	**beseda**	**frekvenca**	**beseda**	**frekvenca**
leta	4,07 %	novo	2,72 %	je	34,99 %
let	2,22 %	mogoče	2,7 %	so	11,73 %
del	1,94 %	sam	2,61 %	bi	6,14 %
dan	1,87 %	slovenski	2,41 %	bo	5,56 %
leto	1,72 %	nove	2,1 %	ni	4,17 %
strani	1,7 %	slovenske	2,01 %	bodo	2,35 %
dela	1,66 %	novi	1,99 %	sem	2,17 %
čas	1,61 %	zadnjih	1,69 %	bil	2,06 %
delo	1,57 %	slovenskih	1,61 %	bilo	1,88 %
mesto	1,54 %	nova	1,58 %	smo	1,86 %

Tabela 4.4: Najpogostejši samostalniki, pridevniki in glagoli v korpusu Gigafida.

samostalnik		**pridevnik**		**glagol**	
lema	**frekvenca**	**lema**	**frekvenca**	**lema**	**frekvenca**
leto	6,07 %	nov	5,6 %	biti	64,47 %
dan	2,58 %	velik	5,36 %	imeti	3,18 %
čas	2,52 %	slovenski	3,95 %	morati	1,52 %
Slovenija	2,28 %	dober	3,75 %	iti	1,09 %
delo	2,2 %	sam	2,51 %	začeti	0,78 %
človek	2,08 %	zadnji	2,42 %	priti	0,77 %
mesto	1,98 %	star	1,83 %	vedeti	0,74 %
država	1,78 %	evropski	1,69 %	dobiti	0,73 %
svet	1,68 %	državen	1,6 %	povedati	0,68 %
del	1,39 %	visok	1,59 %	želeti	0,65 %

Tabela 4.5: Najpogostejše leme samostalnikov, pridevnikov in glagolov v korpusu Gigafida.

celoten korpus		**leposlovje**		**internet**	
črka	**frekvenca**	**črka**	**frekvenca**	**črka**	**frekvenca**
je	8,76 %	je	8,64 %	je	8,4 %
v	7,39 %	v	7,47 %	v	6,81 %
in	7,19 %	in	7,25 %	in	6,73 %
na	4,45 %	na	4,56 %	da	4,21 %
se	3,81 %	se	3,8 %	na	4,19 %
za	3,68 %	za	3,63 %	se	3,83 %
da	3,48 %	da	3,27 %	za	3,71 %
so	2,94 %	so	2,93 %	pa	3,16 %
ki	2,87 %	ki	2,91 %	ki	2,81 %
pa	2,65 %	pa	2,46 %	so	2,42 %

Tabela 4.6: Najpogostejše besede v korpusu Gigafida.

bigrami besed		**bigrami lem**	
bigram	**frekvenca**	**bigram**	**frekvenca**
se je	4,9 %	da biti	6,73 %
da je	3,95 %	biti biti	6,65 %
ki je	3,38 %	ki biti	4,9 %
pa je	3,06 %	se biti	4,73 %
ki so	2,56 %	pa biti	3,8 %
da se	2,44 %	biti v	3,14 %
da bi	2,43 %	on biti	3,09 %
je v	2,35 %	biti se	2,93 %
je bil	2,29 %	ki on	2,09 %
so se	1,77 %	biti on	1,84 %

Tabela 4.7: Frekvence najpogostejših bigramov v korpusu Gigafida.

Poglavje 5

Sklep

V diplomski nalogi smo reševali problem učinkovitega računanja statistik za velike jezikovne korpuse. V javi smo razvili program, s katerim smo izvedli statistično analizo slovenskega jezikovnega korpua Gigafida. Predstavili smo dobljene rezultate in jih v nekaj primerih primerjali z rezultati, ki jih je za analizo angleškega jezika dobil Norvig [3]. Poseben poudarek smo namenili učinkovitim podatkovnim strukturam in razvoju paralelnih algoritmov, ki izkoristijo sodobne večjedrne in večnitne procesorje. Program vhodne podatke obdela paketno: odvisno od kapacitete pomnilnika prebere del korpusa, podatke pretvori v objekte, z uporabo metode Fork-Join podatke razdeli na manjše podprobleme, ki si jih jedra in niti procesorja razdelijo med sabo in s tem učinkoviteje izračunajo zahtevano statistiko. Po izračunu statistike se prebrani podatki brišejo, v sproščen prostor pa program prebere naslednji paket podatkov. To ponavlja dokler ne prebere celotnega korpusa. Podatke na koncu sortira in jih zapiše na disk.

Delo opisano v diplomi predstavlja dobro osnovo, ki bi se jo dalo v prihodnosti še razširiti in izboljšati. V trenutni verziji je program vezan na specifikacije zapisa korpusa Gigafida. Z večjo dinamičnostjo bi bilo moč doseči, da bi lahko vgrajene statistike računali tudi na korpusih z drugačnimi specifikacijami. Gigafida je zgrajena iz velikega števila datotek, ki po velikosti obsegajo od 9,2 kB (77 besed) do 237,1 MB (3.919.356 besed). Na podoben način

- iz velikega števila različno velikih datotek - so sestavljeni tudi drugi veliki korpusi, kar oteži implementacijo zanesljivega prikazovalnika napredka, ki ga v našem programu ni. Implementiran je prikaz zaporedne številke trenutno brane datoteke v razmerju s številom vseh datotek, vendar ta številka iz omenjenih razlogov ni zanesljiva. Trenutno ozko grlo pri procesiranju obsežnih korpusov predstavlja trdi disk; računanje se na večjedrnih in večnitnih procesorjih izvede hitro, medtem ko je branje gigabajtov podatkov omejeno s hitrostjo diskov. Branje korpusa s hitrega SSD diska bi nekajkrat zmanjšalo čas izvajanja.

Program je zastavljen modularno in za pogoste metode uporablja generike. S tem je omogočeno enostavno dodajanje novih statistik v prihodnosti. V primeru, da bi želeli s korpusom pogosto računati različne statistike, bi se splačalo podatke iz XML datotek prenesti v relacijsko podatkovno bazo. S tem bi zmanjšali fizično velikost korpusa, ker za posamezen element ne bi bilo več potrebno dvakrat ponavljati oznake etikete, znebili bi se redundantnih podatkov v kolofonu in imen atributov. Končna baza bi bila dovolj majhna, da bi jo lahko na računalniku z veliko količino pomnilnika ves čas hranili v pomnilniku.

Literatura

[1] Oxford University Press. Kaccāyana-vyākaraṇa. Dostopno na http://www.oxfordreference.com/view/10.1093/oi/authority.20110803100028172 [obiskano 2016-08-14].

[2] M.S. Mayzner in M.E. Tresselt. Tables of Single-letter and Diagram Frequency Counts for Various Word-length and Letter-position Combinations. *Psychonomic Monograph Supplements*, Psychonomic Press, University of Texas at Austin, 1965.

[3] Peter Norvig. English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU. Dostopno na http://www.norvig.com/mayzner.html. [obiskano 2016-08-07]

[4] Max Silberztein. NooJ: A Linguistic Development Environment. Dostopno na http://www.nooj-association.org [obiskano 2016-08-28].

[5] Kaja Dobrovoljc. Procesiranje slovenskega jezika v razvojnem okolju NooJ. Objavljeno v *9. Konferenca jezikovne tehnologije Informacijska družba - IS 2014*, strani 79–84, Ljubljana, 2014.

[6] Saso Džeroski in Tomaž Erjavec. Learning to lemmatise Slovene words. *Learning language in logic*, strani 69-88., Berlin Heidelberg, 2000.

[7] Princeton University. WordNet. Dostopno na https://wordnet.princeton.edu/ [obiskano 2016-08-25].

[8] Tomaž Erjavec in Darja Fišer. Building Slovene WordNet. *Proceedings of the 5th International Conference on Language Resources and Evaluation LREC*, strani 1678–1683, Genoa, 2006.

[9] Jay J. Jiang in David W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. *Proceedings of International Conference Research on Computational Linguistics*, strani 19–33, 1997.

[10] W Nelson Francis in Henry Kucera. The brown corpus: A standard corpus of present-day edited american english. *Providence, RI: Department of Linguistics, Brown University [producer and distributor]*, 1979.

[11] Fabienne Fritzinger in Alexander Fraser. How to Avoid Burning Ducks: Combining Linguistic Analysis and Corpus Statistics for German Compound Processing. *WMT '10 Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, (July):224–234, 2010.

[12] Mark Lauer. Corpus Statistics Meet the Noun Compound: Some Empirical Results. *Proceeding ACL '95 Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, strani 47–54, Association for Computational Linguistics, Stroudsburg, PA, 1996.

[13] James Pustejovsky, Sabine Bergler, in Peter Anick. Lexical Semantic Techniques for Corpus Analysis. *Computational Linguistics*, 19(2):331–358, 1993.

[14] Mark Liberman in Richard Sproat. The Stress and Structure of Modified Noun Phrases in English. *Lexical Matters*, strani 131–181, 1992.

[15] Philip Stuart Resnik. Selection and Information : A Class-Based Approach to Lexical Relationships. *IRCS Technical Reports Series*, stran 200, 1993.

[16] Yuhua Li, Zuhair Bandar, David McLean, James O'Shea, in Keeley Crockett. Sentence Similarity based on Semantic Nets and Corpus Statistics. *IEEE transactions on knowledge and data engineering*, 18(8):1138–1150, 2006.

[17] Claudia Leacock, G a Miller, in Martin Chodorow. Using corpus statistics and WordNet relations for sense identification. *Computational Linguistics*, 24(1):147–165, 1998.

[18] Barry Schiffman in Kristian J. Concepcion. Producing Biographical Summaries : Combining Linguistic Knowledge with Corpus Statistics. *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, strani 458–465, 2001.

[19] Yiming Yang in John Wilbur. Using corpus statistics to remove redundant words in text categorization. *Journal of the American Society for Information Science*, 47(5):357–369, may 1996.

[20] Nataša Logar Berginc, Miha Grčar, Marko Brakus, Tomaž Erjavec, Špela Arhar Holdt, Simon Krek in Iztok Kosem. *Korpusi slovenskega jezika Gigafida, KRES, ccGigafida in ccKRES: gradnja, vsebina, uporaba.* Trojina, Zavod za uporabno slovenistiko, 2012.

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

[22] Tomaž Erjavec, Darja Fišer, Simon Krek, Nina Ledinek. The JOS Linguistically Tagged Corpus of Slovene. Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), 2010. Dostopno na `http://nl.ijs.si/jos/index-sl.html` [obiskano 2016-08-17].

[23] Jonas Bonér. Latency Numbers Every Programmer Should Know, 2016. Dostopno na `https://gist.github.com/jboner/2841832` [obiskano 2016-09-02].

[24] Chris Bailey. From Java code to Java heap Understanding and optimizing your application's memory usage Background: Memory usage of a Java process. Technical report, IBM, 2012.

[25] Robert Stewart in Jeremy Singer. Comparing fork/join and MapReduce, 2012. *Department of Computer Science, Heriot-Watt University*

[26] Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski, in Christos Kozyrakis. Evaluating MapReduce for Multi-core and Multiprocessor Systems. Objavljeno v *2007 IEEE 13th International Symposium on High Performance Computer Architecture*, strani 13–24. IEEE, 2007.

[27] Mattias De Wael, Stefan Marr, in Tom Van Cutsem. Fork/Join Parallelism in the Wild: Documenting Patterns and Anti-patterns in Java Programs Using the Fork/Join Framework. Objavljeno v *PPPJ'14, International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools*, strani 39–50, Cracow, 2014.

[28] Julien Ponge. Fork and Join : Java Can Excel at Painless Parallel Programming Too, 2011. Dostopno na `http://www.oracle.com/technetwork/articles/java/fork-join-422606.html`.

[29] Doug Lea. A Java Fork/Join framework. *Java Grande*, strani 36–43, 2000. Dostopno na `http://portal.acm.org/citation.cfm?doid=337449.337465`.

[30] Angelika Langer. Java performance tutorial – How fast are the Java 8 streams?, 2015. Dostopno na `https://jaxenter.com/`

java-performance-tutorial-how-fast-are-the-java-8-streams-118830.html [obiskano 2016-08-13].

[31] Alex Zhitnitsky. How Java 8 Lambdas and Streams Can Make Your Code 5 Times Slower, 2015. Dostopno na http://blog.takipi.com/benchmark-how-java-8-lambdas-and-streams-can-make-your-code-5-times-slow [obiskano 2016-08-12].

[32] Dominik Charousset in Thomas Schmidt. Scheduler — CAF: C++ Actor Framework, 2016. Dostopno na http://actor-framework.readthedocs.io/en/latest/Scheduler.html [obiskano 2016-09-02].

[33] Volodymyr Korniichuk. Timsort Sorting Algorithm, 2015. Dostopno na http://www.infopulse.com/blog/timsort-sorting-algorithm/ [obiskano 2016-08-26].

[34] M. E. J. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary physics*, 46(5):323–351, 2005.