TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Infotehnoloogia eriala

Uku Raudvere

Lihtsate eesti keele lausete grammatika tuletamine korpusest geneetilise algoritmiga

Bakalaureusetöö (6 EAP)

Juhendaja: prof. Mare Koit

Autor: . . . . . ” . . . . . ” mai 2013

Juhendaja: . . . . . ” . . . . . ” mai 2013

Lubada kaitsmisele:
Professor: . . . . . ” . . . . . ” mai 2013

Tartu 2013

Sisukord

Sissejuhatus

Töö eesmärk on analüüsida geneetilisi algoritme kui tööriista grammatikate automaatseks tuvastamiseks ning konkreetsemalt leida lihtsate eestikeelsete lausete grammatika. Grammatikate tuvastamine näitelausete baasil on huvitav loomulike keelte analüüsi vahend juhul kui grammatika käsitsi koostamine ei ole praktiline, näiteks meditsiiniraportite kui valdkonnaspetsiifilise allkeele grammatika automaattuvastamine artiklis [Kat11]. Lisaks sellele on grammatikate tuletamine leidnud rakendusi andmete kadudeta pakkimise valdkonnas, näiteks [NMWM94].

Järgnevalt tutvustab peatükk 1 tööks vajalikke mõisteid ning toob nende mõistmist hõlbustavaid näiteid. Peatükk 2 täpsustab töös lahendatava ülesande ning tutvustab varasemaid töid, mis on tegelenud analoogilise ülesandega. Lisaks annab peatükk 2 ülevaate ülesande lahendamiseks kasutatud tehnilistest vahenditest. Peatükk 3 kirjeldab lahenduskäiku ning saadud tulemusi. Oluline fookus on geneetiliste algoritmide alamosadel ning nende hindamisel teadaoleva grammatika järgi.

Peatükk 1

Ülevaade mõistetest

Käesolev peatükk annab ülevaate töö seisukohast olulistest mõistetest ning illustreerib neid näidetega.

1.1 Formaalsed grammatikad

1.1.1 Definitsioon

Defineerime grammatika [Cho59] eeskujul.

Grammatika koosneb lõplikust hulgast ümberkirjutusreeglitest kujul $\varphi \to \psi$, kus φ ja ψ on sümbolite jadad. Ümberkirjutusreeglite hulka tähistame P. Sümbolid kuuluvad kas terminaalsesse tähestikku V_T, misjuhul nimetame neid terminaalideks, või mitteterminaalsesse tähestikku V_N, misjuhul nimetame neid mitteterminaalideks. Mitteterminaalne tähestik sisaldab startsümbolit S, mis ei sisaldu ühegi ümberkirjutusreegli paremas pooles. Tühisümbolit ehk jada pikkusega 0 tähistame ε.

Defineerime hulga $V = V_T \cup V_N \cup \{\varepsilon\}$, mis tähendab, et V on kõigi terminaalide ja mitteterminaalide hulk koos tühisümboliga. Kokkuleppeliselt kirjutame mitteterminaalse tähestiku sümbolid suurtähega ja terminaalse tähestiku sümbolid väiketähega. Märgime ära, et eksisteerivad hulgad V_T^*, V_N^* ja V^*, kus $*$ on Kleene'i tärn (mis tähendab, et näiteks V_T^* sisaldab kõikvõimalikke terminaalidest koosnevaid jadasid).

Grammatika defineerib keele. Keel on hulk lõpliku pikkusega lauseid. Keelde

kuuluvad laused koosnevad terminaalidest. Sealjuures keele iga sõna kuulub hulka V_T^* ja seega iga keele L puhul kehtib $L \subset V_T^*$.

Me ütleme, et lause y on vahetult tuletatav lausest x, kui $x = \chi A \omega$, $y = \chi \psi \omega$, kus χ, ψ, $\omega \in V^*$, ja grammatikas leidub produktsioon $A \to \psi \in P$ [Iso12]. Me tähistame seda $a \to b$ või, kui me soovime rõhutada, et y on x-st vahetult tuletatav kasutades n-ndat ümberkirjutusreeglit, $x \xrightarrow{n} y$.

Me ütleme, et lause x_n on tuletatav lausest x_1, kui leidub jada $x_1 \to x_2 \to x_3 \to \ldots \to x_n$. Me tähistame seda $x_1 \Rightarrow x_n$.

Lause s kuulub grammatika G poolt defineeritud keelde L, kui leidub tuletus startsümbolist S lauseni s ja s koosneb ainult terminaalidest. Teisiti öeldes kuulub lause s grammatika G poolt defineeritud keelde L kui $S \Rightarrow s$ ja $s \in V_T^*$.

Me ütleme, et grammatika on kontekstivaba, kui kõik produktsioonid on kujul $V \to \omega$, kus V on täpselt üks mitteterminaal ja ω on jada terminaale ja mitteterminaale. Keelt, mille grammatika on kontekstivaba, nimetame kontekstivabaks keeleks.

Me püstitame lihtsustava eelduse, et eesti keele grammatika on kontekstivaba. Praktikas on alust arvata, et loomulike keelte grammatikad ei ole üldjuhul täielikult kirjeldatavad kontekstivabade grammatikatega [Shi87].

Eelterminaaliks nimetame selle töö piires mitteterminaali, mis sisaldub ainult selliste ümberkirjutusreeglite vasemates pooltes, mille paremates pooltes on ainult terminaalid. Teisisõnu mitteterminaal $E \in V_N$ on eelterminaal siis ja ainult siis, kui iga produktsiooni $(E \to \omega) \in P$ korral $\omega \in V_T^*$.

1.1.2 Näide kontekstivabast grammatikast

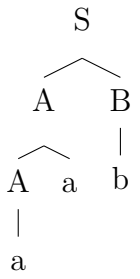
Järgnev kontekstivaba grammatika defineerib keele $L = \{a^+b^+\}$. Näiteks $\{ab, aab, abb, aabb\} \subset L$.

1. $S \to A\ B$
2. $A \to A\ a$
3. $A \to a$
4. $B \to B\ b$
5. $B \to b$

Alustades startsümbolist S, saame näiteks genereerida terminaalsete sümbolite jada aab viisil

$$S \xrightarrow{1} AB \xrightarrow{2} AaB \xrightarrow{3} aaB \xrightarrow{5} aab \ .$$

Sõna aab tuletuspuuks nimetame siis puud



Tuletuspuud on meie jaoks hea võimalus visualiseerida lause grammatilist struktuuri. Formaalsete keelte analüüsimisel tegeletakse põhjalikult ka puude muutmisega mingi semantilise tähenduse piirides.

1.2 Loomulike keelte kirjeldamine kontekstivabade grammatikatega

Eelnevalt kirjeldasime kontekstivaba grammatikaga abstraktset keelt. Selles peatükis näitame, kuidas kontekstivabade grammatikatega saab kirjeldada ja analüüsida loomulike keelte lauseid. Lisaks vaatleme kontekstivabade grammatikate puudusi selles rollis.

Kirjeldame [Eha10] eeskujul grammatika. Lepime kokku, et selmet välja kirjutada kõik produktsioonid kujul $A \to a$, kus A on eelterminaal ja a on terminaal, koostame leksikoni eelterminaalidest ja kõigist nendele vastavatest terminaalidest ning suhtume sellesse nagu oleksid vastavad produktsioonid defineeritud. Lisaks lepime kokku, järgmistes tähistustes mitteterminaalidele:

V tegusõna

VF tegusõnafraas

S nimisõna

SF nimisõnafraas

O omadussõna

K kaassõna

KF kaassõnafraas

Erinevalt eelmisest näitest on meil seega mitteterminaalidele omistatud tähenduslikud sildid.

Kuna märk S on siin kasutusel tähenduses "nimisõna", lepime kokku, et selles grammatikas on startsümboliks "Start".

Lisaks defineerime eelterminalide jaoks järgneva leksikoni.

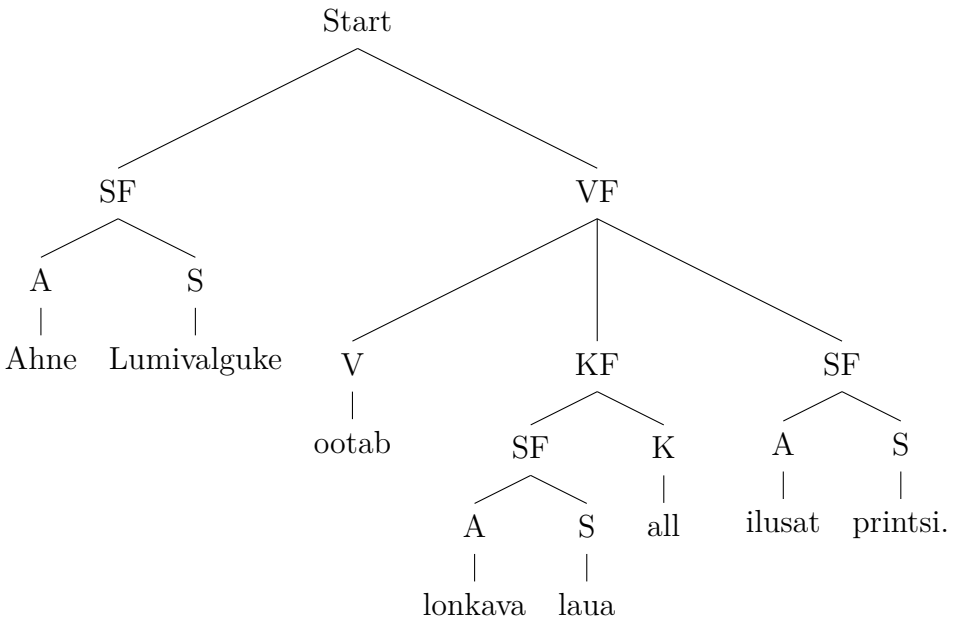
$$
\begin{aligned}
S = &\ \{ \text{Lumivalguke, prints, teemant, laud, nurk} \} \\
V = &\ \{ \text{armastab, tahab, ootab, nutab, vihastab} \} \\
A = &\ \{ \text{ilus, vapper, sädelev, ahne, lonkav} \} \\
K = &\ \{ \text{all, taga, tõttu} \}
\end{aligned}
$$

Eelnevast loeme me välja, et eelterminaali S kohta võiksid grammatikas leiduda produktsioonid $S \to$ Lumivalguke, $S \to$ prints, $S \to$ teemant, $S \to$ laud, $S \to$ nurk .

Lisaks kuuluvad näitegrammatikasse järgmised produktsioonid:

1. Start $\to$ SF VF
2. SF $\to$ A S
3. SF $\to$ S
4. VF $\to$ V SF KF
5. VF $\to$ V KF SF
6. VF $\to$ V KF
7. VF $\to$ V SF
8. VF $\to$ V
9. KF $\to$ SF K.

Valime näitelauseks lause "*Ahne Lumivalguke ootab lonkava laua all ilusat printsi.*" ja koostame selle tuletuspuu.



Näitelause eelterminaalidega kuju on seega "*A S V A S K A S*" ja selle tuletuskäik ülaltoodud grammatika järgi

$$\text{Start} \xrightarrow{1} \text{SF VF} \xrightarrow{2} \text{A S VF} \xrightarrow{5} \text{A S V KF SF} \xrightarrow{9} \text{A S V SF K SF} \xrightarrow{2}$$
$$\text{A S V A S K SF} \xrightarrow{2} \text{A S V A S K A S} \quad .$$

Malliga "*A S V A S K A S*" sobivad ülaltoodud leksikoni järgi näiteks ka laused "*Ilus prints tahab vapra nurga tõttu sädelevat lauda.*" ja "*Ilus laud vihastab ilusa laua taga ilusat lauda.*"

Paneme tähele, et meie grammatika ei sisalda endas teavet sõnade morfoloogia kohta - näidete genereerimisel muutsime me sõnatüvesid ja lisasime sõnadele käände- ja pöördelõppe vastavalt meile eelnevalt teada olevatele eesti keele reeglitele. Kui me eesti keele reegleid juba ei teaks, võiksime sõnu mitte muuta ("*Ahne*

Lumivalguke ootab lonkav laud all ilus prints.") või muuta neid juhuslikult ("*Ahneim Lumivalgukesed ootame lonkavate laud alt ilusam printsile.*"). Mõlemad juhud annavad üldiselt süntaktiliselt väära tulemuse. Hea grammatika peaks seega sisaldama teavet ka sõnade morfoloogia kohta.

1.3 Korpused

Korpuseks nimetatakse keelematerjali kogu. Kui korpuses on sõnadele, lausetele või muudele keeleosadele lisatud lingvistilist informatsiooni, nimetatakse korpust märgendatud korpuseks. Kui lingvistilist informatsiooni lisatud ei ole, nimetatakse korpust märgendamata korpuseks [Kar02].

Selles töös kasutame eesti keele morfoloogiliselt ühestatud korpust [Tar08]. Nimetatud korpuses on sõnadele käsitsi lisatud morfoloogiline informatsioon. Koos sõnadega on korpuses eraldi kategooriatesse märgitud ka kirjavahemärgid.

Morfosüntaktilised kategooriad on leitavad aadressilt `http://www.cl.ut.ee/korpused/morfliides/seletus`. Tuleb ära märkida, et seal on välja toodud ligi 600 erinevat grammatilist kategooriat.

Meie näitelause võiks korpuse reeglite järgi olla märgendatud järgmiselt:

Ahne	ahne+0 // _A_ pos sg nom //
Lumivalguke	Lumivalguke+0 // _S_ prop sg nom //
ootab	oota+b // _V_ main indic pres ps3 sg ps af //
lonkava	lonkav+0 // _A_ pos sg gen //
laua	laud+0 // _S_ com sg gen //
all	all+0 // _K_ post //
ilusat	ilus+t // _A_ pos sg part //
printsi	prints+i // _S_ com sg part //
.	. // _Z_ Fst //

Kirjutame lahti sõna "printsi" kohta käiva rea tabelis. "_S_ com sg part" tähendab, et sõna "printsi" on substantiiv singular partitiiv ehk nimisõna ainsuse osastavas käändes.

Märgendatud näitelausest saame eraldada alljärgneva malli.

1. _A_ pos sg nom
2. _S_ prop sg nom
3. _V_ main indic pres ps3 sg ps af
4. _A_ pos sg gen
5. _S_ com sg gen
6. _K_ post
7. _A_ pos sg part
8. _S_ com sg part
9. _Z_ Fst

Selle malli alusel on võimalik morfosüntakstiliste kategooriate tabelit kasutades genereerida korrektseid laused. Alljärgnev tabel illustreerib antud mallile vastava lause genereerimist.

Sõna vorm mallis	Sõna leksikonis	Sõna korrektses vormis
_A_ pos sg nom	ilus	Ilus
_S_ prop sg nom	teemant	teemant
_V_ main indic pres ps3 sg ps af	nutab	nutab
_A_ pos sg gen	vapper	vapra
_S_ com sg gen	laud	laua
_K_ post	tõttu	tõttu
_A_ pos sg part	vapper	vaprat
_S_ com sg part	nurk	nurka
_Z_ Fst	.	.

Saadud lause - "*Ilus teemant nutab vapra laua tõttu vaprat nurka .*" - on, ehkki semantiliselt väär, süntaktiliselt korrektne.

Järeldame, et loomuliku keele lausete struktuuri on võimalik vähemalt piiratud ulatuses kirjeldada kontekstivabade grammatikatega. Lisaks on võimalik, teades kontekstivaba grammatikat, mis kirjeldab loomuliku keele süntaksit, genereerida süntaktiliselt korrektseid lauseid.

1.4 Geneetilised algoritmid

Geneetilised algoritmid on algoritmide klass, mis on inspireeritud bioloogilisest evolutsioonist. Neid kasutatakse lahendamaks ülesandeid, mille jaoks ei ole teada efektiivset deterministlikku algoritmi.

1.4.1 Geneetiliste algoritmide sõnavara

Defineerime mõned mõisted geneetiliste algoritmide kohta [Mit96] eeskujul.

Populatsiooniks nimetame isendite hulka.

Isendiks nimetame populatsiooni ühte liiget, mida iseloomustab tema kromosoom.

Kromosoomiks nimetame isendi kodeeritud omadusi. Sageli ei tehta vahet isendi ja kromosoomi vahel. Meie räägime grammatikast kui isendist ja grammatika kodeeringust kui kromosoomist.

Geeniks nimetame kromosoomi osa, mis kodeerib ühte omadust või on muul põhjusel loogiliselt eristatav teistest geenidest.

Mutatsiooniks nimetame kromosoomi juhuslikku muutmist.

Ristamiseks nimetame kahe kromosoomi juhuslikku kombineerimist. Ristamise tulemuseks on üks või mitu uut kromosoomi.

Geneetilise algoritmi põhitsükkel on toodud algoritmis 1.

Algoritm 1 Geneetilise algoritmi põhitsükkel

```
1: P ← juhuslikult genereeritud isendid
2: while true do
3:    rista juhuslikult valitud P elemente
4:    muteeri juhuslikult valitud P elemente
5:    hinda P elemente
6:    if oleme piisavalt kaua tööd teinud then
7:       break
8:    end if
9:    if parim element hulgas P on piisavalt hea then
10:      break
11:   end if
12: end while
13: väljasta parim isend hulgast P
```

1.4.2 Klassikalised geneetilised algoritmid

Klassikalisel juhul on geneetiline algoritm meetod ühe bitisõnede hulga muutmiseks teiseks bitisõnede hulgaks. See tähendab, et kromosoomiks on bitisõne, mutatsiooniks on juhusliku biti muutmine ning ristamiseks on kahe sõne sama indeksi juurest pooleks jagamisel saadud alamsõnede rekombineerimine nii, et tekivad uued sama pikkusega sõned.

Selliseid geneetilisi algoritme on mõistlik kasutada juhul kui isend on lihtsasti bitisõneks teisendatav.

Kasutusel on erinevaid algoritme, mida iseloomustavad eeltoodud mõisted, kuid mitte nende klassikalised tähendused. Siiski nimetatakse ka neid algoritme geneetilisteks algoritmideks.

Peatükk 2

Ülesanne

See peatükk täpsustab ülesande eesmärgid ja kirjeldab selle lahendamiseks kasutatud vahendeid.

Käesoleva töö peamine eesmärk on tuvastada lihtsaid eestikeelseid lauseid kirjeldav grammatika.

2.1 Ülesande püstitus

Konkreetseks ülesandeks valime lihtsate eestikeelsete lausete grammatika tuletamise. Selleks valime välja korpusest [Tar08] laused, mis sisaldavad mitte rohkem kui ühte kirjavahemärki.

Nende seast valime juhuslikult 600 lauset positiivseteks näideteks.

2.1.1 Sisendid

Algoritmi sisenditeks on positiivsete näitelausete hulk ning negatiivsete näitelausete hulk.

2.1.2 Väljundid

Algoritmi väljundiks on grammatika G koos hinnangufunktsiooni f tulemusega $f(G)$.

Kuna näitelausete seas suure tõenäosusega ei leidu kõiki terminaale, mis leiduvad korpuses, ei saa grammatikat pidada eesti keele grammatikaks vaid pigem näitelauseid aktsepteerivaks grammatikaks. Grammatikate hindamine terve korpuse baasil või isegi kõigi lihtsate lausete baasil ei ole meie algoritmiga ajaliselt mõeldav.

2.1.3 Raskused

Geneetiliste algortimide spetsiifikaks on, et iga isendit tuleb hinnata. Meie ülesande puhul tähendab see, et iga grammatika puhul tuleb hinnata kõigi positiivsete ja negatiivsete lausete grammatilisust. Kuna me oleme grammatika formalismiks valinud üldkujul kontekstivabad grammatikad, nagu soovitab [Wya94], ja mitte mõnd normaalkuju, mis parandaks parsimiskiirust, tähendab iga isendi hindamine märkimisväärset ajakulu.

Me loeme grammatikat lihtsaks, kui see sisaldab vähe produktsioone ning vähe termiaale ja mitteterminaale. Konkreetsed mahud jätame sihilikult lahtiseks.

Lihtsamate grammatikate hindamine on lihtsam ja võtab vähem aega.

Kuna eesti keel on erinevalt paljudest teistes keeltest morfoloogiliselt keeruline ja morfoloogilisi kategooriaid on palju, on ka otsitavates grammatikates palju terminaale ja eeldatavasti palju mitteterminaale.

Seega on otsitava grammatika hindamine aeganõudev.

Ülesande mahtu saame otseselt piirata valides positiivsete ja negatiivsete näitehulkade mahud.

Negatiivsete näidete hulga valimine on keeruline. Ilmselge on, et eesti keele sõnade baasil on võimalik luua rohkem mittegrammatilisi lauseid kui grammatiliselt korrektseid lauseid. Samas pole võimalik kontrollida piiramatu arvu sihtgrammatikas mittegrammatiliste lausete hinnatavasse grammatikasse mittekuulumist. Seda probleemi on võimalik lahendada mitmeti - meie valime lihtsalt juhusliku hulga lauseid, mis ei leidu positiivses hulgas.

Mitteterminaalide hulga mahtu on mõistlik piirata, et hõlbustada grammatikate ristamist. Kui mitteterminaale on piiramatult, ei ole alust arvata, et ühes

grammatikas genereeritud mitteterminaal omaks mõne muu grammatika piires tähendust. Probleemiks on seega, et tõenäolist mitteterminaalide hulka tuleks hinnata enne eksperimendi algust.

Lisaks on tarvis pöörata tähelepanu grammatika kodeerimisele, muteerimisele ning ristamisele, et saavutada efektiivne versioon geneetilisest algoritmist.

2.2 Varasemad tööd

Selles peatükis püüame anda lühiülevaated töödest, mis meid enam mõjutasid ja konkreetsemalt tööde nendest aspektidest, mis mõjutasid meie tööd kõige enam. Lisaks püüame rõhutada aspekte, mida võiks tulevikus tõsisemalt kaaluda.

2.2.1 *Grammar Generation with Genetic Programming*

Töö "*Grammar Generation with Genetic Programming*" [Zan09] eesmärk oli automaatselt genereerida programmeerimiskeele grammatika.

Grammatikate kirjeldamise formalismiks oli valitud parsimisavaldiste grammatika (*parsing expression grammar, PEG*). Selle formalismi kohta on võimalik lugeda artiklist [For04], üldiselt on selle generatiivsetest grammatikatest eristavaks tunnuseks tõsiasi, et produktsioonide kirjeldused määravad parseri omadusi. Erinevalt kontekstivabadest grammatikatest on produktsioonide järjekord tähenduslik. PEG'e peetakse üldiselt sobilikeks formaalsete keelte grammatikate kirjeldamiseks, aga kuna nad ei võimalda kirjeldada mitteüheseid tuletusi, ei peeta neid reeglina sobilikeks loomulike keelte analüüsiks.

Grammatikate lokaalsetest maksimumidest välja toomiseks võeti kasutusele klient-server mudel, kus erinevad kliendid käitasid sama geneetilist algoritmi erinevate populatsioonide peal üksteisest sõltumatult ning aeg-ajalt tagastasid oma parimad grammatikad serverile, kus omakorda toimus nende hindamine vastavalt geneetiliste algoritmide põhitsüklile. Seda lahendust võib käsitleda kui saaremudelit (*the island model*) nagu see on defineeritud artiklis [FTV03]. Töö ei täpsusta, kas tegemist on homoogeensete või heterogeensete saartega, ei anna hinnangut kuidas saarte kasutamine lokaalsete maksimumide probleemi tegelikkuses mõjutas

ning ei põhjenda klient-server mudeli eelistamist teistele saaremudeli andmevahetuse variantidele (ringid, võred, tähed, juhuslikud topoloogiad jne).

Keel, mille grammatikat püüti tuvastada on esoteeriline programmeerimiskeel *brainfuck*. Järgnevalt esitame selle keele grammatika kontekstivaba grammatikana.

$$
\begin{aligned}
\text{PROGRAMM} &\rightarrow \text{JUHISED} \\
\text{JUHISED} &\rightarrow \text{JUHIS} \\
\text{JUHISED} &\rightarrow \text{JUHIS JUHISED} \\
\text{JUHIS} &\rightarrow \text{TSÜKKEL} \\
\text{JUHIS} &\rightarrow \text{LIHTKÄSK} \\
\text{TSÜKKEL} &\rightarrow [\ \text{JUHISED}\ ] \\
\text{LIHTKÄSK} &\rightarrow + \\
\text{LIHTKÄSK} &\rightarrow - \\
\text{LIHTKÄSK} &\rightarrow < \\
\text{LIHTKÄSK} &\rightarrow > \\
\text{LIHTKÄSK} &\rightarrow . \\
\text{LIHTKÄSK} &\rightarrow ,
\end{aligned}
$$

Järgnevalt on ära toodud töö käigus tuletatud grammatikaga samaväärne grammatika kontekstivaba grammatikana.

PROGRAMM → SÜMBOLID
SÜMBOLID → SÜMBOL SÜMBOLID
SÜMBOLID → SÜMBOL
SÜMBOL → <
SÜMBOL →]
SÜMBOL → .
SÜMBOL → ,
SÜMBOL → >
SÜMBOL → -
SÜMBOL → [
SÜMBOL → +

Ilmneb, et tuletatud grammatika on liiga lubav ning tunneb ära suvalise õigeid terminaale sisaldava sõne. Sellele järeldusele jõudis ka artikli autor. Lahendusena pakutakse välja grammatikate teadaolevate allosade käsitsi defineerimise ning keele erinevate allosade grammatikate eraldi otsimise.

2.2.2 *Representational issues for context free grammar induction using genetic algorithms*

Artikkel [Wya94] tegeleb peamiselt kontekstivabade grammatikate normaalkujude võrdlusega ja püüab analüüsida grammatikate erinevaid kodeerimisvõimalusi ning nende mõju geneetilise algoritmi tulemuslikkusele.

Peamiselt võrreldi Greibach'i normaalkuju (produktsioonid kujul $A \to a\Omega$, kus $A \in V_N$, $a \in V_T$ ja $\Omega \in V_N^*$, formaalne definitsioon antud [AU72], iga kontekstivaba keel on esitatav Greibach'i normaalkujul grammatikaga) (edaspidi GNF) kontekstivabade grammatikate üldkuju (edaspidi CFG) erinevate kitsendustega.

Peamiseks järelduseks on, et GNF kujul produktsioonidega grammatikate leidmine on keerulisem ja aeganõudvam kui üldisel CFG kujul konstantse pikkusega paremate pooltega produktsioonidega grammatikate leidmine. Lisaks saadi häid

tulemusi, kui muudeti CFG kujul produktsioonide paremate poolte pikkusi vastavalt otsitavale grammatikale.

2.3 Töövahendid

2.3.1 Python

Python [van95] on interpreteeritav multiparadigmaline tugevalt ja dünaamiliselt tüübitud kõrgtaseme programmeerimiskeel. Python on populaarne tänu lihtsale süntaksile ja suurele prototüüpimiskiirusele.

Selles töös kasutame Pythonit grammatikate genereerimiseks, nende hindamiseks ja tulemuste analüüsiks.

NLTK

NLTK (*Natural Language Toolkit*) [Bir09] on Pythoni teek loomulike keeltega töötamiseks. See sisaldab vahendeid mitmete korpuste analüüsiks, kontekstivabade grammatikate parserit, vahendeid kollokatsioonide leidmiseks, klassifikaatoreid ja palju muid keeletehnoloogia tööriistu.

2.3.2 Java

Java [Gos13] on kompileeritav objektorienteeritud tugevalt ja staatiliselt tüübitud programmeerimiskeel. Java tugevateks külgedeks on kõrgel tasemel virtuaalmasin ning hea mudel simultaansete programmide kirjutamiseks.

Javat kasutame lausete grammatilisuse kontrolli paralleliseerimiseks.

Gson

Gson [Gso13] on Java teek formaadi JSON Java objektideks deserialiseerimiseks ning Java objektide JSON formaadis sõnedeks serialiseerimiseks.

JSON (*JavaScript Object Notation*) [Cro06] on programmeerimiskeele Javascript alamosa, mida kasutatakse sageli objektide sõnedeks serialiseerimiseks.

JSON'i eelis teiste serialiseerimisprotokollide ees (näiteks XML) on selle suhteliselt väike maht ja lihtsaltparsitavus. Samas on tegemist tekstibaasil formaadiga

(vastandub binaarformaatidele nagu BSON või SDXF, mis võivad olla lühemad), mida on võimalik lugeda ja kirjutada paljudest keeltest ning hallata harilike tekstitöötlusvahenditega.

Pep Earley Parser

Pep [Mar07] on Java teek kontekstivabade grammatikate parsimiseks.

Pep kasutab parsimiseks Earley algoritmi [Ear70]. Earley algoritm jookseb halvimal juhul $O(n^3)$ ajas, kus n on sisendsõne pikkus. Earley algoritm on sobilik kuna sellega on võimalik parsida suvalisi kontekstivabu grammatikaid.

Erinevalt Earley algoritmist suudavad paljud rakenduslikud kompilaatorikompilaatorid töötada vaid kontekstivabade grammatikate mingi piiratud alamosaga.

2.3.3 ZeroMQ

ZeroMQ [Zer13] on teek ja kõrgtaseme protokoll protsessidesiseseks või -vaheliseks suhtluseks ühe arvuti piires või üle võrgu. ZeroMQ sarnaneb oma ülesehituselt klassikaliste soklitega, aga pakub omalt poolt garantiisid andmete kohalejõudmise kohta ning võimaldab andmete automaatset jaotamist paljude soklite vahel.

ZeroMQ'd on võimalik kasutada paljudest erinevatest keeltest, sealhulgas Pythonist ja Javast. Me kasutame ZeroMQ'd, et koodi Pythoni ja Java osade vahel sõnumeid vahetada.

Peatükk 3

Lahendus

See peatükk kirjeldab meie tööd eesti keele grammatika genereerimiseks geneetiliste algoritmidega. Kõigepealt defineerime täpsed geneetiliste algoritmide detailid ning üritame saadud spetsifikatsiooni hinnata lähtudes teadaolevast grammatikast. Viimaks rakendame defineeritud algoritmi korpusele ja analüüsime saadud tulemusi.

3.1 Geneetilised algoritmid

3.1.1 Hinnangufunktsioon

Me hindame grammatikat positiivsete ja negatiivsete näitelausete põhjal.

Valemis on P positiivsete näitelausete hulk, N negatiivsete näitelausete hulk ning P_+ ja N_+ vastavalt grammatika G poolt korrektselt klassifitseeritud positiivsete lausete alamhulk ning korrektselt klassifitseeritud negatiivsete lausete alamhulk.

Defineerime hinnangufunktsiooni

$$f(G) = \frac{1}{2} \cdot \left(\frac{|P_+|}{|P|} + \frac{|N_+|}{|N|} \right) ,$$

mis väljendab õigesti klassifitseeritud sõnade protsenti.

Hinnangufunktsiooni f väärtus $f(G) = 1$ tähendab, et grammatika tunneb ära

kõik sõnad positiivsete näidete seast ning ei tunne ära ühtegi näidet negatiivsete näidete seast. Sealjuures on nii positiivse kui negatiivse hindamise osakaalud võrdsed. Hinnangufunktsiooni väärtus $f(G) = 0,5$ tähendab, et grammatika on sisuliselt kasutu.

Näiteks, grammatikad, mille keelde kuulub väga väike hulk lauseid, saavutavad tulemuse $f(G) = 0,5$, sest nad klassifitseerivad õigesti kõik negatiivsed näitelaused. Teisest küljest saavutavad tulemuse $f(G) = 0,5$ ka väga lubavad grammatikad ehk grammatikad, mis aktsepteerivad peaaegu kõiki lauseid.

3.1.2 Mutatsioonid

Mutatsiooniks loeme grammatika ühe produktsiooni paremas pooles ühe elemendi muutmist.

Suureks mutatsiooniks loeme grammatika mitmes produktsioonis mitme elemendi muutmist.

Produktsiooni elemendi muutmine on elemendi kustutamine, lisamine või asendamine.

Selles töös kasutame erinevates etappides mutatsioone, suuri mutatsioone ja mõlemat korraga. Lisaks kasutame grammatika muutmiseks juhuslike produktsioonide lisamist ning kustutamist.

3.1.3 Ristamine

Ristamiseks valime kaks grammatikat $G1$ ja $G2$. Kummastki valime juhusliku mittetühja pärisalamhulga vastavalt $G1'$ ja $G2'$. Ristamise tulemuseks loeme grammatikad $(G1' \cup G2')$, $((G1 \setminus G1') \cup G2')$, $(G1' \cup (G2 \setminus G2'))$ ja $((G1 \setminus G1') \cup (G2 \setminus G2'))$.

3.1.4 Algoritmi üldine töökäik

Algoritmi põhilist töökäiku kirjeldab algoritm 2. See vastab suuresti geneetilise algoritmi põhitsüklile, mis on toodud algoritmis 1. Oluline erinevus on, et kanname igast põlvkonnast edasi konkreetselt N parimat isendit.

Algoritm 2 Meie algoritmi põhitsükkel pseudokoodis

1: *Grammatikad* ← juhuslikult genereeritud N isendit
2: *Parimad* ← [ ]
3: **while true do**
4: hinda *Grammatikad* ja lisa jadasse *Parimad*
5: sorteeri jada *Parimad*
6: eemalda jadast *Parimad* nõrgimad isendid kuni jada pikkus on N
7: tühjenda *Grammatikad*
8: teosta ristamine juhuslikult valitud *Parimad* elementidel ja lisa tulemused jadasse *Grammatikad*
9: teosta mutatsioon juhuslikult valitud *Parimad* elementidel ja lisa tulemused jadasse *Grammatikad*
10: teosta suur mutatsioon juhuslikult valitud *Parimad* elementidel ja lisa tulemused jadasse *Grammatikad*
11: eemalda produktsioone juhuslikult valitud jada *Parimad* elementide seast ja lisa tulemused jadasse *Grammatikad*
12: **if** parim element hulgas *Parimad* on hinnanguga 1 **then**
13: break
14: **end if**
15: **end while**
16: väljasta parim isend hulgast *Parimad*

Lokaalsesse maksimumi kinni jäämise vastu püüame rakendada agressiivsemalt "suuri mutatsioone". Tähelepanu tasub juhtida tõsiasjale, et mutatsioonide tulemused liiguvad järgmisesse põlvkonda ainult juhul, kui nad mahuvad N parima isendi hulka. Seega ei saa jadasse Parimad kuuluvate grammatikate keskmine hinnanguväärtus kunagi langeda.

3.2 Realisatsioon

Algoritmi põhitsükli implementeerisime programmeerimiskeeles Python.

Grammatikate kiirema hindamise huvides implementeerisime selle Javas. Kasutasime klient-server mudelit, kus Javas realiseeriseeritud lause grammatilisuse

hindamist peame serveriks ning Pythonis realiseeritud põhitsüklit loeme kliendiks.

Üheaegselt töötavad mitu serverlõime. Töö peamiseks keskkonnaks on kaheksa virtuaaltuumaga *Amazon aws High-CPU Extra Large Instance* [Ama]. Seetõttu on vaikimisi kasutusel 12 serverlõime.

Klient saadab juhuslikult valitud serverile JSON formaati serialiseeritud grammatika koos lausega, mille grammatika poolt defineeritud keelde kuuluvust kontrollida. Server saadab sellele vastuseks sõne "True", kui lause kuulub grammatika poolt defineeritud keelde, ning väärtuse "False", kui lause keelde ei kuulu.

3.3 Algoritmi hindamine

Eelkirjeldatud geneetiliste algoritmide alamosade spetsifikatsiooni efektiivsuse hindamiseks testisime seda kasutades eelnevalt teadaolevat grammatikat.

3.3.1 Algoritmi hindamiseks kasutatud grammatika

Algoritmi hindamiseks kasutasime kolme muutuja ja nelja operaatoriga avaldiste grammatikat.

1. $S \rightarrow x$
2. $S \rightarrow y$
3. $S \rightarrow z$
4. $S \rightarrow S \ + \ S$
5. $S \rightarrow S \ - \ S$
6. $S \rightarrow S \ * \ S$
7. $S \rightarrow S \ / \ S$
8. $S \rightarrow (\ S\ )$

Selle grammatika valisime kuna see on mittetriviaalne grammatika, mis sisaldab eritüübilisi produktsioone. Edasises analüüsis nimetame seda sihtgrammatikaks.

Erinevalt korpuse analüüsist piirame me mitteterminaalide arvu ühega.

3.3.2 Saadud tulemused

Testides saadud parim tulemus oli kuue põlvkonnaga leitud järgmine grammatika hinnanguga $f(G) = 0.7720$.

1. S → x (
2. S → y (-
3. S → -
4. S → x
5. S → z
6. S → S / / (x (
7. S → S S
8. S →)) /
9. S → z x y / (
10. S → S * S
11. S → (S)
12. S → x (+
13. S →) * * *
14. S → /) - (
15. S → y
16. S → * z
17. S → z z y S
18. S →) z x
19. S → S + + S *
20. S → +

Paljud produktsioonid selles grammatikas on kasutud ehk ei saa genereerida korrektseid sõnu, sest sisaldavad terminaalide järjendeid, mida korrektses aritmeetilises avaldises ei saa leiduda. Produktsioon 16 ei ole kasulik kuna see dubleerib koos produktsiooniga 7 produktsiooni 10 funktsionaalsust, sealjuures väiksema võimsusega. Kasulikud on produktsioonid 3, 4, 5, 7, 10, 11, 15 ja 20. Teisisõnu võiks selle grammatika minimeerida kujule

1. S → (S)
2. S → +
3. S → -

4. S → S * S
5. S → S S
6. S → x
7. S → y
8. S → z

Näiteks tuvastab minimeeritud grammatika korrektselt kõik sihtgrammatikasse kuuluvad laused, mis ei sisalda jagamistehet. Samas tuvastab see ka lauseid nagu + + + + + + + +, mis selgelt ei ole korrektne avaldis.

Paraku ilmnesid siinkohal tõenäolised vead parsimiseks valitud teegis, mis muutsid leitud grammatikate edasise arendamise võimatuks. Programmi mälukasutus tõusis ilma tuvastatava põjuseta hüppeliselt ning parser jooksis kokku.

Kõik katsed näitasid, et kümne põlvkonnaga on võimalik stabiilselt jõuda hinnanguni $0.6 < f(G) < 0.65$. See näitab, et kui meil õnnestuks algoritmi kauem käitada, oleks lootust saada häid tulemusi. Ühtlasi annab see tulemus alust arvata, et valitud hinnangufunktsioonid ja geneetilise algoritmi parameetrid on vähemalt lihtsate grammatikate tuletamiseks sobilikud.

3.4 Korpusest grammatika tuletamine

Selles peatükis analüüsime katseid tuletada korpusest [Tar08] valitud lihtsate lausete grammatikat.

3.4.1 Töökäik

Lihtsateks nimetame lauseid, mis sisaldavad mitte rohkem kui ühte kirjavahemärki. Korpusest valisime 600 lihtsat lauset positiivseks testhulgaks. Valitud lausete seas on 140 terminaali.

Otsustasime läbi viia eksperimendi, kus grammatikas on ülimalt kümme mittetermiaali.

3.4.2 Tulemuste analüüs

Kõik testid saavutasid tulemuse $f(G) = 0.5$. Vastavalt funktsiooni f definitsioonile tähendab see, et meil ei õnnestunud leida grammatikat, mis midagi ära tunneks.

Me eksperimenteerisime erinevate suurustega grammatikatega.

Kui esialgsed grammatikad olid suured, see tähendab sisaldasid palju pikki produktsioone, leidus neis lubavaid produktsioone, mis tundsid ära mõningaid negatiivseid sõnu. Need produktsioonid tõrjuti kiiresti välja, kuid produktsioone, mis tundnuks ära positiivseid sõnu asemele ei tulnud.

Kui esialgsed grammatikad olid väikesed, ei tundnud grammatikad ära ei positiivseid ega negatiivseid sõnu ning et kõigi grammatikate hinnanguväärtus stabiliseerus $f(G) = 0,5$ juures, puudus otsene evolutsiooniline surve paraneda.

Sellest aspektist võib lugeda eksperimendi ebaõnnestunuks. Eelkõige on ebaõnnestumise põhjuseks see, et kasutatud hinnangufunktsioon ei taganud grammatikate piisavalt täpset järjestamist ning seetõttu ei olnud populatsioonis suunatud arengut. Samas ei olnud nende testide juures grammatikate parserite mälukasutus probleemiks ning väheste täiendustega hinnangufunktsioonile saaks meie realisatsiooni kasutada edasisteks eksperimentideks.

3.5 Mõtteid edasiseks

Geneetiline algoritm osutus väga tundlikuks algparameetrite suhtes, erinevate grammatikate tuletamisel muutus algoritmi viljakus oluliselt sõltuvalt lähtepopulatsiooni produktsioonide keskmisest arvust ja keskmisest pikkusest.

Võimalik, et kasu oleks hinnangufunktsioonist, mis võimaldaks isendeid järjestada peenema granulaarsusega. Eelkõige aitaks see vabaneda korpuste juures ettetulnud probleemist, kus kõik vaadeldud grammatikad said hinnangu $f(G) = 0,5$ ja järgmisesse põlvkonda liikusid neist juhuslikult valitud N tükki. Eelkõige oleks vaja hinnangufunktsiooni defineerides arvestada, et tõenäosus, et mitu grammatikat saavad täpselt sama tulemuse, oleks minimaalne.

Kokkuvõte

Käesolev töö kirjeldas grammatikate genereerimist geneetilise algoritmiga. Meie peamiseks eesmärgiks oli genereerida lihtsate eesti keele lausete grammatika.

Töö andis ülevaate mõningatest varasematest katsetest selles valdkonnas.

Lisaks andsime pealiskaudse ülevaate geneetilistest algoritmidest ning kontekstivabadest grammatikatest. Ühtlasi vaatlesime loomuliku keele kirjeldamist kontekstivaba grammatikaga.

Me andsime hinnangu oma lähenemisele katsetades teadaoleva grammatika tuletamist. Katsetasime oma lähenemist vaadeldes kuidas algoritm suudab näidetest tuletada grammatikat, mis on meile eelnevalt teada. Need katsed andsid märku, et meie algoritm on ülesande täitmiseks sobiv.

Kirjeldasime morfoloogiliselt märgendatud korpuse baasil läbi viidud eksperimenti, mis ei andnud soovitud tulemusi. Meil ei õnnestunud genereerida tähenduslikku grammatikat. Hindasime, et vaja oleks detailsemat hinnangufunktsiooni.

Summary

This work concentrated on generating context-free grammars with genetic algorithms. Its main purpose was to generate a grammar for simple Estonian sentences.

We gave an overview of some previous works in the field. These works discussed ways to represent grammars for genetic algorithms and potential problems, like staying in a local maximum or generating too liberal grammars.

We also gave a brief overview of genetic algorithms and of context-free grammars.

We evaluated our version of genetic algorithms by inducing a known grammar of three-variable algebraic expressions. We concluded that our approach is promising, but the implementation is problematic.

Our experiment of inducing a grammar from a morphologically tagged corpus was not successful. We concluded that the results might be improved by using a more detailed evaluation function.

Kirjandus

[Ama] Amazon Web Services. `http://aws.amazon.com/`. 2013-04-22.

[AU72] Alfred V. Aho and Jeffrey D. Ullman. *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1972.

[Bir09] Steven Bird. *Natural language processing with Python*. O'Reilly, Beijing Cambridge Mass, 2009.

[Cho59] Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2(2):137–167, 1959.

[Cro06] D. Crockford. The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627 (Informational), July 2006.

[Ear70] Jay Earley. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, February 1970.

[Eha10] Martin Ehala. Generatiivne grammatika ja Noam Chomsky. *Oma Keel*, 2010.

[For04] Bryan Ford. Parsing expression grammars: A recognition-based syntactic foundation. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '04, pages 111–122, New York, NY, USA, 2004. ACM.

[FTV03] Francisco Fernández, Marco Tomassini, and Leonardo Vanneschi. An empirical study of multipopulation genetic programming. *Genetic Programming and Evolvable Machines*, 4(1):21–51, March 2003.

[Gos13] James Gosling. *The Java language specification*. Addison-Wesley, Upper Saddle River, NJ, 2013.

[Gso13] Gson. `https://code.google.com/p/google-gson/`, 2013. 2013-05-06.

[Iso12] Ain Isotamm. *Translaatorite tegemise süsteem*. Tartu Ülikooli Kirjastus, Tartu, 2012.

[Kar02] Fred Karlsson. *Üldekeeleteadus*. Eesti keele sihtasutus, Tallinn, 2002.

[Kat11] R.J. Kate. Unsupervised grammar induction of clinical report sublanguage. In *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, volume 2, pages 53–58, 2011.

[Mar07] Scott Martin. Pep is an earley parser. `http://www.ling.ohio-state.edu/~scott/projects/pep/docs/api/`, 2007. 2013-05-06.

[Mit96] Melanie Mitchell. *An introduction to genetic algorithms*. MIT Press, Cambridge, Mass, 1996.

[NMWM94] C.G. Nevill-Manning, Ian H. Witten, and D.L. Maulsby. Compression by induction of hierarchical grammars. In *Data Compression Conference, 1994. DCC '94. Proceedings*, pages 244–253, 1994.

[Shi87] Stuart M. Shieber. Evidence against the context-freeness of natural language. In Walter J. Savitch, Emmon Bach, William Marsh, and Gila Safran-Naveh, editors, *The Formal Complexity of Natural Language*, volume 33 of *Studies in Linguistics and Philosophy*, pages 320–334. Springer Netherlands, 1987.

[Tar08] Tartu Ülikooli arvutilingvistika uurimisrühm. Morfoloogiliselt ühestatud korpus. `http://www.cl.ut.ee/korpused/morfkorpus/`, 2008. 2013-04-22.

[van95] Guido van Rossum. Python tutorial, Technical Report CS-R9526. Technical report, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

[Wya94] Peter Wyard. Representational issues for context free grammar induction using genetic algorithms. *Grammatical Inference and Applications*, 1994.

[Zan09] Sandro De Zanet. Grammar Generation with Genetic Programming. *scg.unibe.ch*, 2009.

[Zer13] The intelligent transport layer. `http://www.zeromq.org/`, 2013. 2013-05-06.

Lisa A

Lähtekood

Lähtekood on tööle lisatud laserplaadil.

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Uku Raudvere (sünnikuupäev: 24.09.1988)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Lihtsate eesti keele lausete grammatika tuletamine korpusest geneetilise algoritmiga", mille juhendaja on Mare Koit

   (a) reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

   (b) üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, 13.mai 2013