TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Informaatika õppekava

Maiko Plinte

Anagrammide genereerija eesti keele jaoks

Bakalaureusetöö
(9 EAP)

Juhendaja: Heiki-Jaan Kaalep

Tartu 2015

Anagrammide genereerija eesti keele jaoks

Lühikokkuvõte:
Selles töös kirjeldatakse mõnda olemasolevat anagrammide genereerijat, võrreldakse eesti keele morfoloogiat inglise keele omaga ning arutatakse, kas nende keelte erinevus tingib vajaduse ka anagrammide genereerimise puhul sõnadega teisiti ümber käia. Kirjeldatakse ka töö käigus valminud eesti keele anagrammide genereerijat.

Võtmesõnad:
Anagramm, morfoloogia, eesti keel

Anagram generator for the Estonian language

Abstract:
This paper describes a couple of already existing anagram generators, compares English morphology to that of Estonian and considers if the differences between the two languages are a good enough reason for handling words differently while generating anagrams. An Estonian anagram generator was made for this paper.

Keywords:
Anagram, morphology, Estonian

Sisukord

Sissejuhatus

Sõna või väljendi anagramm on teine sõna või väljend, mis koosneb samadest tähtedest, millest algne. Heaks anagrammiks loetakse enamasti sellist, mis originaalsõnaga mingil moel seostub, näiteks „*Stipend*“ = „*Spend it*“ [1]. Anagramme kasutatakse tihtipeale kunstivaldkonnas kas pseudonüümidena või teostele nimesid pannes. Seda meetodit on kasutanud näiteks kirjanik Joanne Rowling, kelle „Harry Potteri“ romaanisarjas leidus anagramm „Tom Marvolo Riddle“ = „*I am* Lord Voldemort“ ning Vladimir Nabokov, kes on oma teostesse sisse kirjutanud tegelased „Vivian Darkbloom“ ja „Blavdak Vinimori“, mis on mõlemad tema enda nime anagrammid. Lühikesed anagrammid võivad tunduda väga lihtsad ja neid on võimalik ka vaid sõnale peale vaadates leida, pikemate anagrammide väljamõtlemine on aga üsna keeruline loomeprotsess, mille käigus kasutatakse tihtipeale arvutite abi. Käesoleva töö käigus valmiva anagrammide genereerija üks eesmärkidest ongi anda kasutajale algmaterjal, millest on võimalik endale meelepärased anagrammid välja valida.

Inglise keele jaoks on hulganisti erinevaid programme, mis võimaldavad näiteks kasutatud tähtedel silma peal hoida, lasta programmil saadaolevate tähtede põhjal sõnu soovitada või anagramme genereerida. Hetkel on aga olemas ainult üks vabalt kättesaadav anagrammide genereerija, mis kasutab vaikimisi eestikeelset sõnaloendit. See on Eesti Keele Instituudi poolt loodud veebirakendus[1], kus kasutaja saab vaid sõna või fraasi sisestamise peale näha sisendi eestikeelseid anagramme. Vajaduse korral on muidugi võimalik ka eestikeelseid anagramme ilma spetsiaalse eesti keele jaoks mõeldud anagrammide genereerijata luua. Kuna enamus anagrammide genereerijatest pole keelespetsiifilised, saaks neile eestikeelse sõnaloendi etteandmisel sisendist eestikeelseid anagramme küll. Selliseid allalaaditavaid programme, kus kerge vaevaga sõnaloendit muuta saab, on võimalik põgusa otsimise tulemusel leida, näiteks AG[2] või Softology[3] anagrammi generaator, ning hea inglise keele oskusega tehnikataiplik inimene oleks tõenäoliselt võimeline ka vajadusel selle kõigega hakkama saama. Väga paljud eestlased aga annaksid ilmselt alla, kui paari Google'i päringuga eestikeelset programmi ei leia, seetõttu on eestikeelsel alternatiivil potentsiaali inimeste elu mugavamaks teha ning neile võimalusi juurde anda.

Kuna eesti keele spetsiifika (käänamine, sõnatuletus) tingib vajaduse pöörata sõnadele rohkem või teistsugust tähelepanu kui inglise keele puhul, siis selle bakalaureusetöö käigus peaks valmima eesti keele iseärasustega arvestav anagrammide genereerija.

Töö eesmärk on anda lühike ülevaade anagrammidest ning luua spetsiifiliselt eesti keele jaoks mõeldud uus anagrammide genereerija, mis kasutaks sõnadega tegelemisel teistsugust, eesti keele omapäradega arvestavat lähenemist.

Käesolev töö koosneb kolmest osast: esimeses osas kirjeldatakse nelja juba olemasolevat anagrammide genereerijat ning võrreldakse neid omavahel.

[1] http://www.eki.ee/tarkvara/anagramm/anagramm.cgi
[2] http://www.gson.org/freeware/
[3] http://www.softology.com.au/anagram.htm

Teises peatükis tuuakse välja, kuidas eesti ja inglise keele erinevused anagrammide genereerimisel ilmsiks tulevad ja arutatakse, kas eestikeelses programmis tuleks sõnaloendi saamisele teistmoodi läheneda.

Kolmandas peatükis kirjeldatakse töö käigus valminud anagrammide genereerija algoritmi ja seal kasutatud ideid ning ressursse. Tehakse ka lühike kokkuvõte programmist ning selle käitumisest.

Neljandas peatükis tuuakse välja uue anagrammide genereerija probleemsemad kohad, pakutakse mõnele probleemile võimalik lahendus ning peatüki teises osas kirjeldatakse programmi tulevikuplaane ning arendamismõtteid.

Lisadeks on loodud anagrammide genereerija lähtekood ning tema kasutusjuhend ja ka kõik peatükis 3.1.3 kirjeldatud failid.

1. Olemasolevad anagrammide genereerijad

Järgnevalt kirjeldatakse mõnda juba varem valminud avatud lähtekoodiga anagrammide genereerijat.

1.1 AG

AG[4] on soomlase Andreas Gustafssoni loodud anagrammide genereerija, mis on kirjutatud keeles C aastal 1992.

AG hoiab sõnu ja lauseid täisarvumassiividena, kus iga element näitab, mitu korda vastav täht sõnas või lauses esineb. Sõnaloendi läbikäimisel jäetakse kohe kõrvale sõnad, kus esineb tähti, mida sisendis pole. Edasiste kontrollide käigus eemaldatakse ka need sõnad, mis sisaldavad mingit tähte rohkem kordi kui sisend. Kõik sobilikud kandidaadid lisatakse paisktabelisse. Anagrammide genereerimiseks hakatakse paisktabelit läbima ning iga kandidaatsõna puhul vaadatakse, kas ta koosneb samadest tähtedest, millest sisendki. Kontrollimine toimub algse täisarvumassiivi kaudu: sõna iga tähe kohta lahutatakse massiivis vastava tähe esinemiste arvust 1. Kui massiivis esineb negatiivseid arve, ei sobi see sõna anagrammi moodustamiseks, kuna sisaldab mingit tähte rohkem kordi kui sisend. Kui massiivis leidub positiivseid täisarve, on leitud osaline anagramm ning programm üritab rekursiivselt ülejäänud tähtedehulgale samuti anagrammi leida. Kui massiiv koosneb nullidest, on järelikult anagramm leitud ning see väljastatakse ekraanile. [2]

1.2 AN

AN on Richard Jonesi ja Julian Assange'i poolt 1995. aastal keeles C kirjutatud anagrammide genereerija[5] [6].

Anagrammide leidmiseks tehakse kõigepealt sisendfraas väiketäheliseks ning eemaldatakse tühikud ning kõik muud ühikud, mis ei ole tähed. Järgnevalt käiakse rida-realt läbi sõnaloendi fail ning juhul, kui käesolev sõna on moodustatav sisendfraasi tähtedest ning pikem kui soovitav sõna miinimumpikkus, lisatakse see sobivate sõnade massiivi. Sarnaselt eelmisele programmile kasutatakse tähtede sagedustabelit, kus igale tähele vastab tema esinemissagedus. Edasi hakatakse sobivate sõnade massiivi läbi töötama nii, et võetakse sobiv sõna ning eemaldatakse kasutaja sisestatud fraasi tähtede hulgast need tähed, mis valitud sõnas esinevad, edasi otsitakse rekursiivselt sõnu, mis koosnevad ülejäänud tähtedest või nende alamhulgast. Kui jõutakse olukorrani, kus kõik kasutaja sisestatud tähed on ära kasutatud ning rekursiivne tsükkel on lõpetanud, võetakse järgmine sõna sobivate sõnade massiivist ning korratakse tsüklit senikaua, kuni sobivate sõnade loend on läbitöötatud. [3]

1.3 Anagram Finder

Anagram Finder[7] on 2002. aasta märtsis välja tulnud anagrammide genereerija, mille kirjutas keeles C++ John Walker.

Anagrammide leidmist alustab Anagram Finder pikkuse järgi sorteeritud sõnaloendi sisselugemisega. Kohe jäetakse kõrvale sõnad, mis on pikemad kui kasutaja sisestatud fraas

[4] http://www.gson.org/freeware/
[5] http://manpages.ubuntu.com/manpages/hardy/man6/an.6.html
[6] http://fatphil.org/words/an.html
[7] http://www.fourmilab.ch/anagram

ning sõnad, mis võrreldes sisestatud fraasiga sisaldavad mingit tähte rohkem kordi või sisaldavad tähti, mida seal ei ole. Selle kontrollimiseks kasutatakse nii kandidaatsõna kui ka sisestatud fraasi tähtede esinemissageduse sõnastikku, kus igale tähele vastab tema esinemiste arv. Anagrammide avastamiseks kasutatakse jällegi rekursiivset funktsiooni, mis võtab ühe sobiva sõna ning vaatab, mis tähed esinevad kasutaja poolt sisestatud fraasis, aga mitte sobivate sõnade loendist valitud sõnas. Edasi kontrollitakse, missuguseid sõnu sobivate sõnade hulgast saab nende ülejäänud tähtedega moodustada. Kui kõik sisendfraasi tähed on täpselt üks kord ära kasutatud, on järelikult anagramm leitud ning valitakse järgmine kandidaatsõna. [4]

1.4 Eesti Keele Instituudi anagrammide genereerija

Kirjutamise hetkel on ainult üks eestikeelne anagrammide genereerija ning selle on teinud Indrek Hein Eesti Keele Instituudist. Programmi kirjelduseks on märgitud „Teeb küll vaid kuni kahest "sõnast koosnevaid" variante ja sõnavormide loend vajaks nii puhastamist kui täiendamist, aga kasutada saab.“ [5].

Kahest sõnast koosnevus on jutumärkide sisse pandud, kuna see rakendus ei tegele liitsõnade moodustamisega, kuid selle sõnaloend[8] sisaldab hulganisti liitsõnu. Näiteks andes rakendusele sisendiks sõna „liiduvabariik“, antakse vastuseks nii terviksõna „liiduvabariik“, kuna see on sõnastikus olemas, kui ka kahesõnaline „liidu ++ vabariik“. Sama juhtub, kui küsida sõna „vabariik“ anagramme: vastuste hulgas on nii „vabariik“ kui ka „vaba ++ riik“.

Anagrammide leidmine ise on kirjutatud Perlis. Sisendist eemaldatakse kõik, mis ei ole tähed, seejärel muudetakse väiketäheliseks ning pannakse tähed tähestikulisse järjekorda. Edasi luuakse sõnaloendi põhjal anagrammisõnastik kõikidest sõnadest, mis võivad anagrammi kuuluda. Selle anagrammisõnastiku võtmed käiakse läbi ning iga võtme korral vaadatakse, kas ülejäänud võtmete hulgas on üks teine võti, millega kahepeale kokku on nende tähestikuliselt järjestatud tähtedejada võrdne sisendi omaga. Kui on, väljastatakse rida kujul „S[a1] ++ S[a2]“, kus S on anagrammisõnastik ning a1 ja a2 võtmed, mis on kahepeale kokku sisendi anagrammid. Sisendi „kanapea“ puhul on üks väljastatud ridadest „akna kana ++ pae pea“, kus võti a1 oleks sel juhul „aakn“ ning võti a2 „aep“ ning sisendi anagrammideks „akna pae“, „akna pea“, „kana pae“, „kana pea“. [6]

Sõnaloend on 196840-sõnaline, kuid nagu rakenduse kirjelduses on öeldud, vajaks see nii puhastamist kui täiendamist. Sõnaloend on saadud korpusest, mistõttu pole seal sõna kõiki vorme. Sõnapaari „ametlik poni“ anagrammideks pakutakse teiste hulgas ka sõnapaare „elanto pikim“ ja „teolan pikim“, kuigi ei Eesti Keele Instituudi õigekeelsussõnaraamat, Eesti keele seletav sõnaraamat ega ka Filosofti morfoloogiline analüsaator ei tunne sõnu „elanto“ ega „teolan“, samas sõnu „poni“ ega „ametlikkus“ sõnastikus ei leidu. Järgmises peatükis tabelis 2 toodud 37-st sõna „tüdruk“ erinevast vormist on Eesti Keele Instituudi anagrammide genereerija sõnaloendis vaid 18 (puudu on 19: „tüdrukusse“, „tüdrukus“, „tüdrukuni“, „tüdrukuna“, „tüdrukuta“, „tüdrukuisse“, „tüdrukuis“, „tüdrukuist“, „tüdrukuile“, „tüdrukuil“, „tüdrukuilt“, „tüdrukuiks“, „tüdrukuini“, „tüdrukuina“, „tüdrukuteta“, „tüdrukutesse“, „tüdrukuteks“, „tüdrukuteni“, „tüdrukutena“).

[8] http://www.eki.ee/tarkvara/anagramm/anagrammisonastik.txt

1.5 Olemasolevate anagrammigeneraatorite võrdlus

Kuigi eelmistes alapeatükkides kirjeldatud neli anagrammide genereerijat on kirjutatud kolmes erinevas programmeerimiskeeles ning ka teostus on kõigil erinev, töötavad nad ikkagi väga sarnaselt. Kõik vaatavad terve sõnadeloendi lineaarselt läbi ning valivad sealt välja kandidaatsõnad, mida saaks kasutaja sisestatud fraasis esinevatest tähtedest moodustada. Seejärel hakatakse väljavalitud sõnu omavahel kokku panema ning kontrollima, kas nad annavad kokku sisendfraasi anagrammi. Kolm algselt inglise keele jaoks tehtud anagrammide genereerijat: AG (vt peatükk 1.1), AN (vt peatükk 1.2) ja Anagram Finder (vt peatükk 1.3) kasutavad sõnade võrdluseks tähtede sagedusloendit, Eesti Keele Instituudi lehel olev eesti keelele mõeldud variant kasutab aga anagrammisõnastikku.

2. Eesti ja inglise keele erinevused

Eesti keel on inglise keelest väga erinev: keeleõpetamisega tegeleva ettevõtte Adelante inglise keele õpiku „KUULA & KORDA Inglise keel madalamale kesktasemele 1“ järgi on inglise keeles ainult kaks käänet: nimetav ehk üldkääne ning omastav kääne. Üldkääne on käändelõputa ning omastav kääne moodustatakse üldiselt ülakoma ja tähe „s“ abil. Sõnadele, mille lõpus on juba „*s*“, lisatakse ainult ülakoma [7].

Et anagramme otsides eemaldatakse tavaliselt kõik märgid, mis ei ole tähed, jäävad enamasti kõrvale ka ülakomad. Reeglipärase mitmusega sõnade puhul tähendab see, et tihtipeale sõna mitmusevorm, omastava käände vorm ning mitmuse omastava käände vorm langevad kokku. Sõna „*girl*“ on reeglipärase mitmusega, selle sõna kõik inglisekeelsed vormid on järgmised:

Tabel 1. Reeglipärase mitmusega sõna “*girl*” käänamine inglise keeles **[7].**

	Ainsus	**Mitmus**
Üldkääne	*girl*	*girls*
Omastav kääne	*girl’s*	*girls’*

Kui ülakoma kõrvale jätta, oleksid tabelis 1 toodud sõnadest kolm samasugused ning sõnaloendisse sobiksid vaid „*girl*“ ja „*girls*“. Eesti keeles on aga teatavasti neliteist käänet:

Tabel 2. Sõna "tüdruk" käänamine eesti keeles.

	Ainsus	**Mitmus**
Nimetav	tüdruk	tüdrukud
Omastav	tüdruku	tüdrukute
Osastav	tüdrukut	tüdrukuid
Sisseütlev	tüdrukusse	tüdrukuisse, tüdrukutesse
Seesütlev	tüdrukus	tüdrukuis, tüdrukutes
Seestütlev	tüdrukust	tüdrukuist, tüdrukutest
Alaleütlev	tüdrukule	tüdrukuile, tüdrukutele
Alalütlev	tüdrukul	tüdrukuil, tüdrukutel
Alaltütlev	tüdrukult	tüdrukuilt, tüdrukutelt
Saav	tüdrukuks	tüdrukuiks, tüdrukuteks
Rajav	tüdrukuni	tüdrukuini, tüdrukuteni
Olev	tüdrukuna	tüdrukuina, tüdrukutena
Ilmaütlev	tüdrukuta	tüdrukuteta
Kaasaütlev	tüdrukuga	tüdrukutega

Sõna „*girl*“ ja selle eestikeelse vaste „tüdruk“ puhul peaks inglisekeelsesse sõnaloendisse panema kaks sõna („*girl*“, „*girls*“), eestikeelsesse aga tabeli 2 järgi koguni 37. Mõnedest omadussõnadest on võimalik moodustada lisaks ka lühike sisseütleva käände vorm: sõna „pea“ ainsuse sisseütlevas käändes on „peasse“, selle lühivorm on „pähe“.

Ka tegusõnade puhul on eesti keeles hulganisti rohkem võimalikke pöördevorme. Morfoloogilise analüsaatori dokumentatsioonis on välja toodud kõik pöördevormid, mis ta ära tunneb:

Tabel 3. Eesti keele verbide pöördevormid **[8]**.

Käändevorm	Sõna
kindel kõneviis olevik 3. isik ainsus aktiiv jaatav	loeb
kindel kõneviis olevik 2. isik ainsus aktiiv jaatav	loed
infinitiiv jaatav	lugeda
gerundium jaatav	lugedes
käskiv kõneviis olevik 2. isik mitmus aktiiv jaatav	lugege
käskiv kõneviis olevik 1. isik mitmus aktiiv jaatav	lugegem
käskiv kõneviis olevik 3. isik ainsus/mitmus aktiiv jaatav	(ta/nad) lugegu
tingiv kõneviis olevik aktiiv jaatav	(ma/sa/ta/me/te/nad) loeks
tingiv kõneviis olevik 2. isik ainsus/3. isik mitmus aktiiv jaatav	(sa/nad) loeksid
tingiv kõneviis olevik 1. isik mitmus aktiiv jaatav	(me) loeksime
tingiv kõneviis olevik 1. isik ainsus aktiiv jaatav	(ma) loeksin
tingiv kõneviis olevik 2. isik mitmus aktiiv jaatav	(te) loeksite
supiin aktiiv jaatav kõne sisseütlev	lugema
supiin aktiiv jaatav kõne saav	lugemaks
supiin aktiiv jaatav kõne seesütlev	lugemas
supiin aktiiv jaatav kõne seestütlev	lugemast
supiin aktiiv jaatav kõne ilmaütlev	lugemata
kindel kõneviis olevik 1. isik mitmus aktiiv jaatav	loeme
kindel kõneviis olevik 1. isik ainsus aktiiv jaatav	loen
kesksõna minevik aktiiv jaatav	lugenud
tingiv kõneviis minevik aktiiv jaatav	(ma/sa/ta/me/te/nad) lugenuks
tingiv kõneviis minevik 2. isik ainsus/3. isik mitmus aktiiv jaatav	(sa/nad) lugenuksid
tingiv kõneviis minevik 1. isik mitmus aktiiv jaatav	lugenuksime
tingiv kõneviis minevik 1. isik ainsus aktiiv jaatav	lugenuksin
tingiv kõneviis minevik 2. isik mitmus aktiiv jaatav	lugenuksite
kaudne kõneviis minevik aktiiv jaatav	(ma/sa/ta/me/te/nad) lugenuvat
käskiv kõneviis olevik 2. isik ainsus aktiiv jaatav	loe
kindel kõneviis lihtminevik 3. isik ainsus aktiiv jaatav	luges
kindel kõneviis lihtminevik 2. isik ainsus/3. isik mitmus aktiiv jaatav	(sa/nad) lugesid
kindel kõneviis lihtminevik 1. isik mitmus aktiiv jaatav	lugesime
kindel kõneviis lihtminevik 1. isik ainsus aktiiv jaatav	lugesin
kindel kõneviis lihtminevik 2. isik mitmus aktiiv jaatav	lugesite
kindel kõneviis olevik passiiv eitav	loeta
käskiv kõneviis olevik passiiv jaatav	loetagu
tingiv kõneviis olevik passiiv jaatav	loetaks
kindel kõneviis olevik passiiv jaatav	loetakse
supiin passiiv jaatav	loetama

kesksõna olevik passiiv jaatav	loetav
kaudne kõneviis olevik passiiv jaatav	loetavat
kindel kõneviis olevik 2. isik mitmus aktiiv jaatav	loete
kindel kõneviis lihtminevik passiiv jaatav	loeti
kesksõna minevik passiiv jaatav	loetud
tingiv kõneviis minevik passiiv jaatav	loetuks
kaudne kõneviis minevik passiiv jaatav	loetuvat
kesksõna olevik aktiiv jaatav	lugev
kindel kõneviis olevik 3. isik mitmus aktiiv jaatav	loevad
kaudne kõneviis olevik aktiiv jaatav	(ma/sa/ta/me/te/nad) lugevat

Inglise keeles on aga ka verbivorme vähem ning nende tegemiseks kasutatakse tihti eessõnu:

Tabel 4. Inglise keele verbivormid **[9]**.

	Past	**Present**	**Future**
Simple	painted	paints/paint	will paint
Continuous	was/were painting	am/are/is painting	will be painting
Perfect	had painted	have/has painted	will have painted
Perfect Continuous	had been painting	have/has been painting	will have been painting

Tabelis 4 on sõna „*paint*“ erinevatest vormidest unikaalseid sõnu vaid neli (*paint, paints, painted, painting*), eesti keeles on aga sõnast „lugema“ koguni 47 erinevat verbivormi (vt. Tabel 3).

Kuna eesti ja inglise keel ning nende sõnastikud/sõnavara on nii erinevad, võib tekkida õigustatult küsimus, kas anagrammide genereerimisele võiks või peaks ehk teistmoodi lähenema, selmet lineaarselt kõikvõimalikest sõnadest koosnev sõnaloend läbi käia.

3. Algoritm

Käesolevas peatükis kirjeldatakse töö käigus valminud anagrammide genereerijat.

3.1 Algoritmi osad

Alapeatükk 3.1 kirjeldab ideid ja ressursse, millele algoritm tugineb.

3.1.1 Knuthi algoritm

Knuthi algoritm on lihtne moodus ühesõnaliste anagrammide leidmiseks. Selle järgi saab sama pikkusega anagrammid kokku grupeerida järgmiselt:

1) Otsi ühe ja sama pikkusega sõnade loend
2) Käi sõnaloend läbi ning pane igas sõnas tähed tähestikulisse järjekorda ignoreerides tühikuid, muid kirjavahemärke ja tõstutundlikkust (näiteks sõna ’Knuth’ sorteeritult on ’hkntu’), saades sõnast A sorteeritud tähejada A*
3) Sorteeri kõik paarid (A*, A) omavahel tähestikulises järjekorras

Tulemuseks on paaride järjend, kus kõik anagrammid on koos, sest iga kahe anagrammi A1 ja A2 puhul on A1* ja A2* võrdsed. [10]

Juhul, kui sõnade loendis ei ole ainult sama pikkusega sõnad, piisab ühesõnaliste anagrammide leidmiseks järgmisest algoritmist:

1) Olgu meil sõnaloend
2) Sorteeri loendis sõnad pikkuse järjekorda
3) Käi loend läbi ning pane igas sõnas tähed tähestikulisse järjekorda, ignoreerides tühikuid ja tõstutundlikkust, saades sõnast A sorteeritud tähejada A*
4) Sorteeri kõik paarid (A*, A) omavahel tähestikulises järjekorras

Tulemuseks on paaride loend, kus anagrammid on kokku grupeeritud. Kui paarid pelgalt sorteerimise asemel paisktabelisse panna, nii et võtmeks on sorteeritud tähejada A* ja väärtuseks massiiv kõikidest sõnadest A, mis tähtede tähestikulisse järjekorda pannes moodustavad A*, tekib anagrammisõnastik. Juhul, kui algses sõnaloendis leidub vähemalt kaks sõna, mis on üksteise anagrammid, on anagrammisõnastiku võtmete hulk väiksem kui sõnaloendis olevate sõnade hulk, järelikult on ka anagrammisõnastiku võtmete lineaarne läbikäimine ning sobivuse kontroll kiirem.

3.1.2 Filosofti morfoloogiatarkvara

Filosoft[9] on loonud eesti keelele morfoloogiatarkvara. Morfoloogilise analüüsi väljundiks on iga sõna kõik võimalikud analüüsid konteksti arvestamata [11]. Analüüs on leksikonipõhine ning selle käigus määratakse ära sõna struktuur, sõnaliik ja kääne või pööre.

Antud morfoloogiatarkvarast on Githubis olemas ka vabavaraline versioon[10] ning Pythoni[11] liides[12]. Käesolevas töös kasutatakse morfoloogilise analüsaatori Pythoni liidest ilma oletamiseta, et kontrollida, kas tükkidest moodustatud sõna on grammatiliselt korrektne.

[9] http://www.filosoft.ee
[10] https://github.com/Filosoft/vabamorf

3.1.3 Sõnaloend

Eesti morfoloogia on suures osas aglutinatiivne [12], mis tähendab, et sõnavorme moodustatakse morfeemide liitmise teel. Näiteks sõna „vandeseltslaslikkustki“ saab jagada tükkideks „vande, selts, las, likk, us, t, ki“ [11].

Käesoleva töö käigus valminud anagrammide genereerija kasutab eesti morfoloogia aglutinatiivsust ära ning valmissõnade loendi asemel on kasutusel kõikvõimalikud tükid: sõnad, sõnatüved, liited ja lõpud, millest oleks võimalik reeglipäraseid eesti keele sõnu koostada. Tükid on võetud Filosofti morfoloogiatarkvarast, kus nad on kasutuses sõnaloendi moodustamise ühe vaheetapina [13].

Olemas on järgmised failid (noolsulgude vahel on failis olevate tükkide arv):

1) pref.ok <454> - prefiksid, näiteks ’aja’, ’ala’, ’alg’, ’all’
2) nomm.ok <117532> - nimisõnade tüved, näiteks ’kahtlane’, ’kahtlas’, ’kahtlase’
3) mmm.ok <6986> - muutumatute sõnade (määr-, kaas-, side- ja hüüdsõnad) tüved, näiteks ’kahtlemata’, ’kahtlematult’, ’kahtlemisi’, ’kahtlevalt’
4) verb.ok <16328> - tegusõnade tüved, näiteks ’kaits’, ’kaitse’, ’kan’, ’kanda’
5) suf.ok <784> - sufiksid, näiteks ’masina’, ’masin’, ’tö’, ’töö’
6) lopud.ok <145> - käände- ja pöördelõpud, näiteks ’a’, ’akse’, ’b’, ’d’

Neid tükke kombineerides on võimalik saada väga paljusid eestikeelseid sõnu, näiteks ’kaitse+töö+d = kaitsetööd’ või ’all+töö+võtja’.

Algsetest failidest on eemaldatud mitmed kohanimed: näiteks *Bénin*, *Føroyar*, *Côte=d’Ivoire* ja *Al-Imārāt*, kuna nendes esinevad võõrtähed tekitasid juba failide lugemisel probleeme ning väga suure tõenäosusega ei leiaks sellised sõnad spetsiifiliselt eesti keele jaoks loodud anagrammide genereerijas palju kasutust. Lisaks on eemaldatud mitmed fraasid, mis sisaldasid failides võrdusmärki, nagu *Côte=d’Ivoire* ja *La=Manche.* Võrdusmärk tähendab siin, et need sõnad kuuluvad kokku ning moodustavad ühe fraasi,

3.1.4 Väljund

Kuigi väljundit on mugav lugeda, kui iga anagramm eraldi real on, andis anagrammide genereerija esialgne versioon autori nimele üle 2000 anagrammi, kui anagrammi moodustava sõna minimaalseks pikkuseks oli seatud 4. Kui anagrammide hulk on suur, võib see aga kasutajat heidutada ning suure tõenäosusega ei loeta kõiki ridu läbi.

Autori nime anagrammide hulgas olid ka järgmised sõnapaarid:
pint imalkoe, pint maiolek, pint emakilo, pint imealko, pint omalike, pint imekola, pint ilmakoe, pint emakoil, pint moekail, pint imekaol, pint alkoime, pint limakoe, pint omakile, pint laimkoe, pint kolaime, pint moekali, pint maikoel, pint aimkoel, pint aolikem, pint emaloik, pint emakoli, pint maolike.

[11] https://www.python.org/
[12] https://github.com/estnltk/pyvabamorf

Selle muutmiseks otsustas töö autor eeskuju võtta Eesti Keele Instituudi anagrammide genereerijast, kus mitu anagrammi on ühe rea peale koondatud. Kuna eelmises anagrammiloendis algavad kõik anagrammid sama sõnaga, võiks selle samuti ühe reaga kirjutada:

„Pint + imalkoe, maiolek, emakilo, imealko, omalike, imekola, ilmakoe, emakoil, moekail, imekaol, alkoime, limakoe, omakile, laimkoe, kolaime, moekali, maikoel, aimkoel, aolikem, emaloik, emakoli, maolike“. Nii on kõik sõnad, mis jäävad plussmärgist paremale poole, ka omavahel anagrammid. Juhul, kui anagramme leidub, saab sama teha ka sõnaga „pint“ ning jätta kõik tema anagrammid plussmärgist vasakule poole.

3.2 Algoritm

Järgnevalt kirjeldatakse lühidalt anagrammide genereerijas kasutatavat algoritmi:

1. Eemaldatakse sisendsõnelt tühikud, muud kirjavahemärgid ja seejärel tehakse väiketäheliseks
2. Loetakse failidest sisse kõik sõnatükkide loendid (vt peatükk 3.1.3)
3. Käiakse tükiloendid läbi ning valitakse välja vaid need tükid, mis koosnevad sisendsõne tähtede alamhulgast
4. Üritatakse väljavalitud tükke kombineerides sõnu moodustada, sõna lisatakse anagrammisõnastikku, kui kõik järgnevad tingimused on täidetud
    - saadud sõna on pikem kui määratud minimaalne pikkus
    - saadud sõna on lühem kui sisendsõne pikkuse ja minimaalse pikkuse vahe (juhul, kui sisendsõne on 10 tähte pikk ning kombineerimise tulemusel saadud sõna 7 tähte pikk, võib saadud sõna kõrvale jätta, kui minimaalne sõnapikkus on 4, kuid kombineerimine läheb edasi, sest on võimalik, et tekib 10-täheline anagramm)
    - morfoloogiline analüsaator ütleb, et sõna on grammatiliselt korrektne (vt peatükk 5.1.2)
    - moodustatud sõna koosneb sisendsõne tähtede alamhulgast
5. Sobivate sõnade anagrammisõnastiku võtmete hulgast hakatakse anagramme otsima
    - võetakse esimene võti ning hakatakse seda ülejäänud võtmetega kombineerides proovima, kas nad annavad kahe peale kokku sisendsõne anagrammi
        1. kui annavad, tagastatakse anagrammisõnastiku mõlemale võtmele vastavad väärtused
        2. kui kahe võtme tähtede hulk on ikka veel sisendsõne tähtede alamhulk, üritatakse rekursiivselt leida ka kolmandat võtit, mis koos esimese kahega annaksid kokku sisendsõne anagrammi, ja nii edasi
6. Trükitakse leitud anagrammid välja (vt peatükk 3.1.4)

3.2.1 Koodi kommenteerimine

Faili alguses on imporditud peatükis 3.1.2 kirjeldatud morfoloogilise analüsaatori Pythoni liidesest „pyvabamorf“ meetod „*analyze*“. Tavalise sõnastiku asemele on imporditud „defaultdict“, kuna sellega on sõnastikuga suhtlemine lihtsam: ei pea võtme olemasolu kontrollima, vaid saab ka kohe sõnastikust päringut teha (Lisa 1, rida 42). Real 4 on

määratud väljundsõnade miinimumpikkus, milleks on hetkel autori suva järgi pandud 4. Põhjenduseks on see, et kui kahe- ja kolmetähelised sõnad anagrammide genereerimisest kõrvale jätta, on tulemused kvaliteetsemad, sest on suurem tõenäosus, et nad kas tähendavad midagi või on humoorikad.

Kasutajaga suhtlemiseks on tehtud tsükkel (Lisa 1, rida 145), mille kasutaja saab lõpetada sisestades tähe 'q'. Tsüklis küsitakse kasutajalt fraasi ning siis antakse see programmi põhimeetodile, mis hakkab sisendist anagramme genereerima. Failidest sobivate tükkide väljafiltreerimine toimub meetodis „sobivad", mis algab programmi kuuendalt realt. Edasi antakse kõik sobivad tükid meetodile „sonad" (rida 34), mis hakkab neid kombineerima viisil prefiks + nimisõna, verb või muutumatu sõna + sufiks + käändelõpp. Pärast iga sõnatükkide kombineerimiskatset kontrollitakse, kas moodustis on pikem kui minimaalne soovitud sõnapikkus, kas see on morfoloogilise analüsaatori arvates korrektne eestikeelne sõna (rida 30) ja kas ta sisaldub kasutaja sisestatud fraasis (rida 31).

Ridadel 42-45 ja ka hiljem korduv tegevus on Knuthi algoritmi rakendamine (peatükk 3.1.1): sõna sorteeritakse tähestikulisse järjekorda, millest saab anagrammisõnastiku „sd" võti. Võtmele vastav väärtus tehakse hulgaks (rida 43), et lahti saada korduvatest sõnadest, hulgale lisatakse uus sõna ning uuendatud hulk lisatakse järjendina tagasi anagrammisõnastikku.

Lõpetuseks antakse real 139 kõik sobilikuks peetud sõnad meetodile „anagrammid" (rida 113), mis hakkab neid kombineerima ning kontrollima, kas sõnade kombinatsioon sisaldub sisestatud fraasis (rida 118) või on juba anagramm (rida 119).

3.2.2 Algoritmi kommenteerimine

Nagu eelmisest alapeatükist näha võib, on käesoleva töö käigus valminud anagrammide genereerijal esimeses peatükis kirjeldatud teiste anagrammide genereerijatega nii sarnaseid jooni kui ka erinevusi.

Kõikide algoritmide puhul kuulusid sarnasuste hulka failide lineaarne läbivaatamine ning sealt sobivate ridade väljavalimine ning nende sobivate sõnade hulga põhjal anagrammide moodustamine. Eesti Keele Instituudi anagrammide genereerijaga on peatükis 3.2 kirjeldatud algoritmil ühine veel Knuthi algoritmi kasutamine ühetäheliste anagrammide leidmiseks. Ülejäänud kolme algoritmiga (peatükid 1.1, 1.2 ja 1.3) on samasugune veel ka lõplik anagrammide genereerimine: kasutatakse rekursiivset funktsiooni sobivate sõnade rittapanekuks ning sisendsõnaga võrdlemiseks.

Suurim erinevus on leksikonipõhine: kõik varemvalminud programmid kasutasid anagrammide moodustamiseks sõnu, mis on failis valmiskujul, uus algoritm aga üritas eesti keele morfoloogiale tuginedes sõnade saamisele teisiti läheneda ning olemasoleva sõnaloendi asemel kasutati erinevaid eesti keele sõnade tükke, mida sõnade kokkupanemiseks kasutati.

3.2.3 Algoritmi kokkuvõte

Muude anagrammigenereerijatega võrreldes teistsugune leksikonile lähenemine tasus ennast ära, kuna anagrammide genereerimiseks saadaolevate sõnade arv on suurem kui Eesti Keele Instituudi anagrammigenereerija kasutuses olevas sõnaloendis. Selle arvelt on aga töökiirus aeglasem, sest rohkemate sõnade käsitlemine võtab paratamatult rohkem aega, lisakoormuseks on veel tükkidest sõnade moodustamine. Programmi tööaeg sõltub suuresti sisendist: kui sisestada kümme „a" tähte, ei leita ühtegi anagrammi ning vastus tuleb väga

kiiresti. Mida rohkem sisendsõna moodustamiseks sobivaid tükke failidest leitakse, seda kauem programm töötab.

Programm on veel algusjärgus: ainukene funktsionaalsus on hetkel anagrammide genereerimine, neljandas peatükis kirjeldatakse tuleviku potentsiaalseid lisandusi.

4. Anagrammigenereerija tulevik

Järgnevalt tuuakse välja loodud anagrammigenereerija puhul mõned kohad, mis tulevikus üle tuleks vaadata.

4.1 Probleemid

- **Mitmed morfoloogilise analüsaatori poolt lubatud liitsõnad on küsitavad**

  Kui liitsõnu nagu „maiolek“, „ emakilo“ ja „moekail“ võib sellele vaatamata, et nad on grammatiliselt ning tähenduselt korrektsed, veidi kentsakateks pidada, siis sõnad nagu „pikieol“ (morfoloogiline analüsaator annab algvormiks „pikiidu“) tunduvad lausa täielikult tähenduseta moodustised.

- **Sõnaloend**

  Kuna sõnade moodustamiseks kasutatav tükkide loend pole antud anagrammide genereerija jaoks tehtud, vaid on võetud morfoloogilise analüsaatori küljest, oleks anagrammide genereerimist kindlasti võimalik optimeerida sõnaloendi analüüsimise ja sobivamaks tegemise kaudu.
  Hetkel on sõna „silma“ prefiksite loendis ja sõnad „silmaterake“, „silmaterakes“, „silmaterakese“ ning „terake“, „terakes“ ja „terakese“ nimisõnatükkide loendis. Anagrammide genereerimise käigus toimuv liitsõnade moodustamine saaks sõnade „silma“ ja „terake“ liitmisel ise sõna „silmaterake“ loomisega hakkama.

- **Tükkide hulgas on samu sõnu**

  Sõna „kaitse“ esineb nii nimisõnatükkide kui ka verbitükkide loendis. Esialgu oli see õigustatud, kuna kõikides tükiloendites oli ka muu info sõnatüübi ning võimalike lõppude kohta, ning morfoloogiline analüsaator käsitleb verbe ja nimisõnu erinevalt. Antud anagrammide genereerija aga hetkel mitte, nii et korduvad sõnad võiks eemaldada või üritada algselt failides olnud andmeid optimeerimiseks kasutada.

- **Liitsõnade moodustamine**

  Kuna eesti keeles on liitsõnade moodustamine küllaltki keeruline, tuleks üle vaadata liitsõnade moodustamise viis.

- **Kas märki „-“ anagrammide genereerimisel arvestada?**

  Paljudele fraasidele tuleb sidekriips vahele panna: „aeg-ajalt“, „tühi-tähi“, „linka-lonka“, „see-eest“, „emb-kumb“. Kui millegipärast otsitaks anagrammi sõnadest „taga ajel“ ning minimaalne sõnapikkus oleks kolm, ei tekiks sidekriipsu mittearvestamisest probleemi, sest anagrammiks saadaks ka „aeg + ajalt“. Mõned sidekriipsuga kirjutatavatest fraaside osadest ei tähendagi aga ükshaaval midagi, näiteks morfoloogiline analüsaator ei tunne fraasist „emb-kumb“ ära sõna „emb“, kui ta

omaette esineb, nii et kõikide sidekriipsuga kirjutatavate sõnade kõrvalejätmisega jääks paratamatult kõikide võimalike anagrammide hulk väiksemaks.

Kui sidekriipsu arvestada tavalise tähena, väheneks paljudel juhtudel samuti anagrammide hulk: otsides nime „Mari-Ann“ anagramme ning tahtes saada vaid vasteid, milles on ka sidekriips, jääb vastuste arv väga väikeseks, kuna kõikide võimalike sõnade arvuga võrreldes on sidekriipsu nõudvate sõnade arv siiski üsna väike.

Juhul, kui sisendist siiski sidekriips eemaldada ning anagramme genereerides sõnaloendis olevaid sidekriipse ignoreerida, võidakse aga kurta, et tegu pole täieliku anagrammiga, kui sisendis sidekriipsu pole, aga vastuses on.

4.2 Arendamisvõimalused

Järgnevalt tuuakse välja mõned arendamismõtted, milleni veel jõutud ei ole.

- Pythoni sisseehitatud moodul Pickle[13] lubab Pythoni andmestruktuure faili kirjutada ja sealt lugeda. Võiks proovida seda moodulit kasutades kõikidest tükiloenditest anagrammisõnastikud moodustada, siis ei peaks rakenduse töölepanekul enam tükiloendifailidest midagi otsima, vaid saaks anagrammisõnastikest kiiremini kõik sobivad tükid kätte.
- Kindlasti võiks tulevikus antud programmi ka veebirakendusena vabalt kättesaadavaks teha
- Veebirakenduse kasutajaliides peaks kindlasti kasutajale pakkuma võimalust anagrammis esinevate sõnade miinimumpikkuse muutmiseks
- Kasutajal võiks olla võimalik ka peatükis 3.1.4 kirjeldatud kahe erineva väljund-formaadi vahel valida, lisaks võib pakkuda ka võimalust välja printida kõik permutatsioonid
- Kasutajal võiks olla võimalus sisestada sõna(d), mis peavad kindlasti anagrammis olema: näiteks öeldes rakendusele, et anagrammid peavad sisaldama sõna „hobune“, saadakse vastuseks ainult sellised anagrammid, kus antud sõna sees on
- Kasutajal võiks olla võimalus üksikuid sõnu sõnaloendisse lisada. Potentsiaalne kasutusvõimalus oleks järgmine: kui kasutaja nime sõnaloendis ei esine, on tal võimalik see sinna lisada ning seda eelmise punktiga kombineerides saaks kasutaja ka öelda, et tahab näha ainult anagramme, kus uus sõna sees on. Nii oleks võimalik kiiresti ja lihtsalt leida erinevate sõnade anagramme, mis kasutaja enda nimega seostuvad
- Kuna Filosofti morfoloogiline analüsaator pakub võimalust tundmatute sõnade puhul oletada, kuidas see käänduks, saaks seda võimalust eelmise punktiga kombineerida, nii et kui kasutaja lisab tundmatu sõna sõnastikku, on tal ka võimalus lasta analüsaatoril oletada, kuidas see käändub, selle asemel et ise kõik võimalikud käändevormid käsitsi sõnaloendisse lisada
- Kui vastuste hulgas on „ninakil + olek“ ja „ninakilolek“, antakse hetkel vastuseks mõlemad, võiks aga ainult esimese variandi anda, sest nii väheneks võimalus, et tekivad imelikud, kahtlase tähendusega liitsõnad.

[13] https://docs.python.org/2/library/persistence.html

5. Kokkuvõte

Eesti ja inglise keel on küllalt erinevad, et võiks proovida anagrammide genereerimisel leksikoni kasutamisele erinevalt läheneda, seda ka käesoleva töö käigus tehti.

Spetsiaalselt eesti keele jaoks anagrammide genereerija tegemine oli edukas: olemas on anagrammide genereerija, mis läheneb asjale teisiti kui lihtsalt sõnaloendis olemasolevate valmissõnade lineaarne läbivaatamine ning selline lähenemine on end osaliselt ka õigustanud, kuna lõpptulemusena tekib erinevaid anagramme rohkem kui teises olemasolevas eesti keele anagrammide genereerijas, anagrammide genereerimiseks kulub aga suurema sõnade arvu ning sõnade kokkupanemise tõttu rohkem aega. Samuti on programmil veel nii kasutajamugavuse, funktsionaalsuse kui ka optimeerimise osas kindlasti arenguruumi.

6. Tsiteeritud teosed

[1] J. Gearhart, 1999.

[2] A. Gustafsson, „GSON Free software,“ [Võrgumaterjal]. Saadaval: http://www.gson.org/freeware/. [Kasutatud 13 mai 2015].

[3] [Võrgumaterjal]. Saadaval: http://manpages.ubuntu.com/manpages/hardy/man6/an.6.html. [Kasutatud 13 mai 2015].

[4] J. Walker, „Anagram Finder,“ 2002. [Võrgumaterjal]. Saadaval: http://www.fourmilab.ch/anagram. [Kasutatud 13 mai 2015].

[5] „Eesti Keele Instituudi tarkvara,“ [Võrgumaterjal]. Saadaval: http://www.eki.ee/tarkvara. [Kasutatud 13 mai 2015].

[6] I. Hein, „Eesti Keele Instituudi anagrammide genereerija lähtekood,“ Eesti Keele Instituut, [Võrgumaterjal]. Saadaval: http://www.eki.ee/tarkvara/anagramm/anagramm.txt. [Kasutatud 13 mai 2015].

[7] K. Liebert, *KUULA & KORDA Inglise keel madalamale kesktasemele 1*, ADELANTE KOOLITUS, 2009, pp. 65-66.

[8] „Vabamorfi kasutatavad märgendid,“ Filosoft, [Võrgumaterjal]. Saadaval: https://github.com/Filosoft/vabamorf/blob/master/doc/tagset.html. [Kasutatud 13 mai 2015].

[9] „English verb tense system,“ [Võrgumaterjal]. Saadaval: http://clickonenglish.blogspot.com/2013/02/english-verb-tense-system.html. [Kasutatud 13 mai 2015].

[10] D. Knuth, *The Art of Computer Programming Vol. 3*, Addison-Wesley Publishing Company, 1973, p. 576.

[11] H.-J. Kaalep, „Morfoloogililne analüüs,“ [Võrgumaterjal]. Saadaval: http://kodu.ut.ee/~hkaalep/heilile/keeletehnoloogia_informaatikutele_2015.pdf. [Kasutatud 13 mai 2015].

[12] Ü. Viks, „Eesti keele avatud morfoloogiamudel,“ Eesti Keele Instituut, [Võrgumaterjal]. Saadaval: http://www.eki.ee/teemad/avatud_mrf.html. [Kasutatud 13 mai 2015].

[13] „Sõnaloendi loomine,“ Filosoft, [Võrgumaterjal]. Saadaval: https://github.com/Filosoft/vabamorf/tree/master/dct/data/mrf. [Kasutatud 13 mai 2015].

Lisad

I. Kasutusjuhend

„readme.txt“

Nõuded:

Python 3.4 või uuem
Pyvabamorf'i liides:
liidese töölesaamiseks järgida juhiseid lehel 'https://github.com/estnltk/pyvabamorf'

Käivitamiseks tuleb failil topelklõps teha.

II. Leksikonifailid

Peatükis 3.1.3 kirjeldatud failid „pref.ok“, „nomm.ok“, „mmm.ok“, „verb.ok“, „suf.ok“, „lopud.ok“.

III. Programmi lähtekood

```
from pyvabamorf import analyze
from collections import defaultdict

min_pikkus = 4

def sobivad(fail, sisend):      ### Leiab sõnastikust sobivad sõnad
    sobivad = []
    for rida in fail:
        rida = rida.strip()
        if sisaldab(sisend, rida) is not None:
            if len(sisend) >= len(rida):
                sobivad.append(rida)
    return sorted(sobivad, key=len, reverse=True)

# kas sisend sisaldab seda sõna?
# kui sisaldab, siis eemaldab sisendist selle sõna tähed
def sisaldab(sisend, sona):
    if len(sisend) >= len(sona):
        sona = sona.strip().lower()
        while sona:
            taht, sona = sona[0:1] , sona[1:]
            if taht not in sisend:
                return None
            sisend = sisend.replace(taht, '', 1)    ### eemaldab kasutatud tähe
        return sisend                       ### ülejäänu

def sona(sisend,s):                 ### kas on sõna?
    if len(sisend) >= len(s):    ### kas on lühem kui sisend
        if s:
            if analyze(s, guess=False, phonetic=False,compound=False)[0]['analysis']: # on sõna?
                if sisaldab(sisend,s):      ### kui kõik uue sõna tähed sisalduvad sisendis
                    return True

def sonad(sisend, jj): ### jj - järjendite järjend
    verbs, sufs, prefs, noms, mms, lopps = jj[0], jj[1], jj[2], jj[3], jj[4], jj[5]
    sd = defaultdict(list)
    for pref in prefs+['']:
        for n in noms+['']:
            uus1 = pref+n           ### liida prefiks ja tüvi
            if sona(sisend, uus1):  ### kui uus sõna on analüsaatori arvates sõna
                if len(uus1) >= min_pikkus: ### ja uus sõna on pikem miinimumpikkusest
                    s = sd[''.join(sorted(uus1))]
                    a = set(s)
                    a.add(uus1)
                    sd[''.join(sorted(uus1))] = list(a)      ### lisa sobivate sõnade loendisse
                for suf in sufs+['']:       ### liida uuele sõnale suffiks
                    uus2 = uus1+suf         ### ja nii edasi
                    if sona(sisend, uus2):
                        if len(uus2) >= min_pikkus:
                            s = sd[''.join(sorted(uus2))]
                            a = set(s)
                            a.add(uus2)
                            sd[''.join(sorted(uus2))] = list(a)
                        for lopp in lopps+['']:
                            uus = uus2 + lopp
                            if sona(sisend, uus):
                                if len(uus2) >= min_pikkus:
                                    s = sd[''.join(sorted(uus))]
                                    a = set(s)
                                    a.add(uus)
```

```
                                sd[".join(sorted(uus))] = list(a)
        for v in verbs+['']:
            uus1 = pref+v
            if sona(sisend, uus1):
                if len(uus1) >= min_pikkus:
                    s = sd[".join(sorted(uus1))]
                    a = set(s)
                    a.add(uus1)
                    sd[".join(sorted(uus1))] = list(a)
                for suf in sufs+['']:
                    uus2 = uus1+suf
                    if sona(sisend, uus2):
                        if len(uus2) >= min_pikkus:
                            s = sd[".join(sorted(uus2))]
                            a = set(s)
                            a.add(uus2)
                            sd[".join(sorted(uus2))] = list(a)
                        for lopp in lopps+['']:
                            uus = uus2 + lopp
                            if sona(sisend, uus):
                                if len(uus2) >= min_pikkus:
                                    s = sd[".join(sorted(uus))]
                                    a = set(s)
                                    a.add(uus)
                                    sd[".join(sorted(uus))] = list(a)
        for m in mms+['']:
            uus1 = pref+m
            if sona(sisend, uus1):
                if len(uus1) >= min_pikkus:
                    s = sd[".join(sorted(uus1))]
                    a = set(s)
                    a.add(uus1)
                    sd[".join(sorted(uus1))] = list(a)
                for suf in sufs+['']:
                    uus2 = uus1+suf
                    if sona(sisend, uus2):
                        if len(uus2) >= min_pikkus:
                            s = sd[".join(sorted(uus2))]
                            a = set(s)
                            a.add(uus2)
                            sd[".join(sorted(uus2))] = list(a)
                        for lopp in lopps+['']:
                            uus = uus2 + lopp
                            if sona(sisend, uus):
                                if len(uus2) >= min_pikkus:
                                    s = sd[".join(sorted(uus))]
                                    a = set(s)
                                    a.add(uus)
                                    sd[".join(sorted(uus))] = list(a)
    return sd


def anagrammid(sisend, sonad, min_pikkus, sonadedict):
# leiab sisendile mitmesõnalised anagrammid,
# mis koosnevad vähemalt min_pikkusega sõnadest
    i = 0
    for sona in sonad:
        yle = sisaldab(sisend, sona)
        if yle == '':          ### Sõna ongi anagramm
            yield sona
```

```
121             elif yle is not None and len(yle) >= min_pikkus:
122                 for j in anagrammid(yle, sonad[i:], min_pikkus, sonadedict):
123                     if sonadedict[j] == []:
124                         break
125                     else:
126                         yield (' + '.join((' '.join(sonadedict[sona]), ' '.join(sonadedict[j]))))
127             i += 1
128 
129     def main(sisend):
130         sisend = sisend.lower().strip().replace(' ','').replace('-','')
131         tykid = ['verb.ok', 'suf.ok', 'pref.ok', 'nomm.ok', 'mmm.ok', 'lopud.ok']
132         s_tykid = []                ### Kõik sobilikud tykid
133         for sonastik in tykid:
134             with open(sonastik) as f:
135                 s_tykid.append(sobivad(f, sisend))   ### Sobilikud tükid
136         sonu = 1
137         sonadd = sonad(sisend, s_tykid)
138 
139         for sona in anagrammid(sisend, list(sonadd.keys()), min_pikkus, sonadd):
140             print(str(sonu) + '. ' + sona)
141             sonu += 1
142 
143     print("\nSisesta väljumiseks 'q'")
144     sisend = input('või sisesta sõna(d): ')
145     while sisend != 'q':
146         main(sisend)
147         sisend = input('\n Sisesta sõnad: ')
```

IV. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, MAIKO PLINTE

(sünnikuupäev: 17.03.1993)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

Anagrammide genereerija eesti keele jaoks

mille juhendaja on
Heiki-Jaan Kaalep,

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. 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, **14.05.2015**