Tallinna Ülikool
Informaatika Instituut

Tarkvara statistiliste kollokatsioonide eraldamiseks ning selle rakendus morfosüntaktilises analüüsis

Seminaritöö

Autor: Sander Ots
Juhendaja: Erika Matsak

Autor: ..... „ ..... „2011

Juhendaja: ..... „ ..... „2011

Tallinn 2011

Sisukord:

Sissejuhatus

Õppides Eesti keelt kui võõrkeelt, tuleb ette palju raskusi. Raskused võivad olla seotud erinevate reeglitega, mis puuduvad inimese emakeeles. Keele õppija võib samuti koostada kohmaka lauseehitusega lauseid ise sellest teadmata. Selles töös käsitletakse tarkvaradisaini, mis kergendaks morfosüntaktilist töötlemist erinevates keelealastes uuringutes. Loodava rakenduse tulemuste uurimisega saavad keeleteadlased esile tuua reegleid, mis annaks võimaluse uute metoodikate loomiseks.

Keele aspektist on need uuringud olulised. Vastavate abivahendite loomine muudaks uuringuid põhjalikumaks ning asendaks nn. käsitöö. Keeleteadlased teevad praegu tööd suurte andmekogudega ilma vastava tarkvarata. Käesoleva seminaritöö teema on kindlasti väga aktuaalne.

Tarkvara peab suutma morfosüntaktiliste sõnatüüpide abil leidma klastreid ja kollokatsioone ehk seaduspärasusi lausetes. Valmisolevat tarkvara uurides ei leidnud programmi, mis seda teha suudaks. Olemasolev tarkvara suudab leida sõnade klastrid ja kollokatsioonid, küll aga mitte sõnatüüpide omi.

Loodav programm hakkab asendama Erika Matsaku VBA[1]'s kirjutatud exceli makrosid, mis võimaldasid poolautomaatselt morfosüntaktiliste klastrite otsimist (Metslang & Matsak, 2010). Eraldiseisvat programmi on vaja täieliku automatiseerituse saavutamiseks. Lisandub ka rohkem funktsionaalsust.

Klastreid ja kollokatsioone tavaelus otsitakse, näiteks selleks, et selgitada välja standardseid fraase. Standardsete fraaside tundmine aitab õppida keelt (neile kelle jaoks see keel ei ole emakeel), samuti on hea standardfraase kasutada automaatsetes tõlgetes. Kui aga me uurime morfosüntaktilisi klastreid või kollakatsioone, siis me saame kätte grammatilised mallid. Grammatiliste mallide abil on jällegi kergem keelt õppida.

Mitte eestlastele, näiteks venelastele, on päris keeruline aru saada käänetest mis vastavad küsimusele mille, mida ja mis. Vene keeles sellist käänet nagu "mille" ei ole. Inimesele on aga suureks abiks standardsed mallid nagu: pane (mis) uks kinni, ära pane (mida) ust kinni, ma panen (mille) ukse kinni. Grammatiliste seoste loomine lähtudes sõna

[1]Visual Basic for Applications

eespoolt seisvast vormist kergendab keele kasutust sel juhul kui on raske aru saada miks just nii või naa mingis keeles asi on esitatud.

Töö eesmärgiks on kindlaks teha mida meie rakendus peab kõike sisaldama, et saavutaks oma eesmärgi: otsida tekstidest morfosüntaktiliste märgendite järgi klastreid ja kollokatsioone.

Antud seminaritöö oleks aluseks tarkvara programmeerimisel. See hõlmab programmi funktsionaalsust, mäluhaldust, keeruliste algoritmide vooskeeme (*flowchart*) ja disaini.

Töö sisaldab viite peatükki. Esimeses peatükis seletan tähtsate mõistete tähenduse ja määran kindlaks, mis loodav programm peab kõike võimaldama. Teises peatükis on ülevaade kolmest olemasolevast tarkvarast: WordSmith Tools 5, kfNgram ja Xaira. Kolmandas peatükis on täpsem ülevaade programmi funktsionaalsusest. Neljandas peatükis on lahti seletatud rakenduse tehniline osa. Viiendas peatükis on graafilisest kasutajaliidesest ja süžeetahvlist (*Storyboard*) juttu.

1. Mõistete seletus ja tarkvara eesmärk

Antud peatükis selgitan tähtsamad mõisted lahti. Samuti teen kindlaks loodava tarkvara eesmärgid.

Mõistete seletus

Klaster - grupp sarnaseid elemente, mis on lähestikku. Antud töös on selle all mõeldud järjest paiknevate sõnade märgendid.

Kollokatsioon - on samuti klaster, mis on saavutatud kindla sõna märgendi otsingu tulemusena, milles üks element on otsitav sõna, mis paigutatakse klastri keskele ning tuuakse esile tema vasakpoolne ning paremapoolne ümbrus. (Smadja F. A & McKeown, 1990)

Süntaktiline analüüs - lause esitamine formaalsete märgendite abil. Märgendid tähistavad erivevaid lause rolle, näiteks öeldis, põhja märgendeid: subjekti ja objekti erinevaid laiendeid jne. Kokku on 27 märgendit. (Müürisep, ESTKG süntaktilised märgendid)

Morfosüntaktiline analüüs - ühendab endas kaks analüüsi: morfoloogilist ning süntaktilist. (Kaalep & Muischnek)

Morfosüntaktiline märgend - sõna kõikvõimalik kirjeldus. Formaalne märgend, mis esitab kirjeldust sõna vormi kohta ning selle rolli lauses. (Müürisep, Morfoloogilised märgendid)

N-gram - on järjest paiknevate elementide jada. Tavalises kontekstis on elementideks sõnad või üksikud tähed. Selle abil saab vaadata millised osad esinevad koos teistega ja kui sageli.

Tarkvara eesmärk

Loodav tarkvara peab võimaldama kasutajal pikkadest tekstidest leida morfosüntaktiliste märgendite järgi klastrid või kollokatsioonid.

2. Ülevaade olemasolevast tarkvarast

Antud peatükis heidame pilgu keeleuurimis programmidele. Valikus on kolm programmi, WordSmith Tools 5, kfNgramja Xaira. Valituks osutusid need, sest WordSmith Tools 5 on väga populaarne ja kfNgram klassikaline N-gram meetodil töötav ning samuti seetõttu, et just nende programmide kohta on juba olemas põhjalik võrdlev analüüs (Ari, 2006). Igas programmis kasutan näitelauseid „Kolm sõna järjest. Neli sõna järjest. Viis sõna järjest.“. Sisestan igasse programmi näitelaused ja ka näitelaused morfosüntaktiliste märgenditega. Et saada morfosüntaktilisi märgendeid kasutan morfoanalüsaatorit EstCGParser (mis on välja töötatud ning testitud Tartu Ülikoolis). Lause „Kolm sõna järjest“ morfosüntaktiliste märgenditega näeks välja selline:

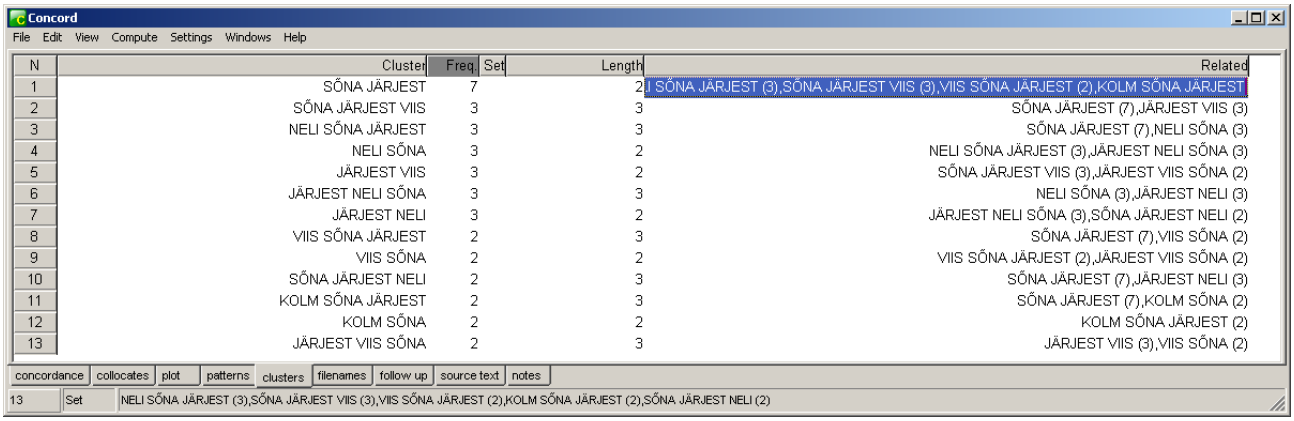
LA
*#### **CLB @???*
Kolm
*kolm+0 //_N_ card sg nom #cap l // **CLB @SUBJ @ADVL*
sõna
sõna+0 //_S_ com sg part // @<Q
järjest
järjest+0 //_D_ // @ADVL
järg+st //_S_ com sg el // @ADVL @<NN
LL
@???

Tulemus 1 - "Kolm sõna järjest" morfosüntaktiline märgend

WordSmith Tools 5

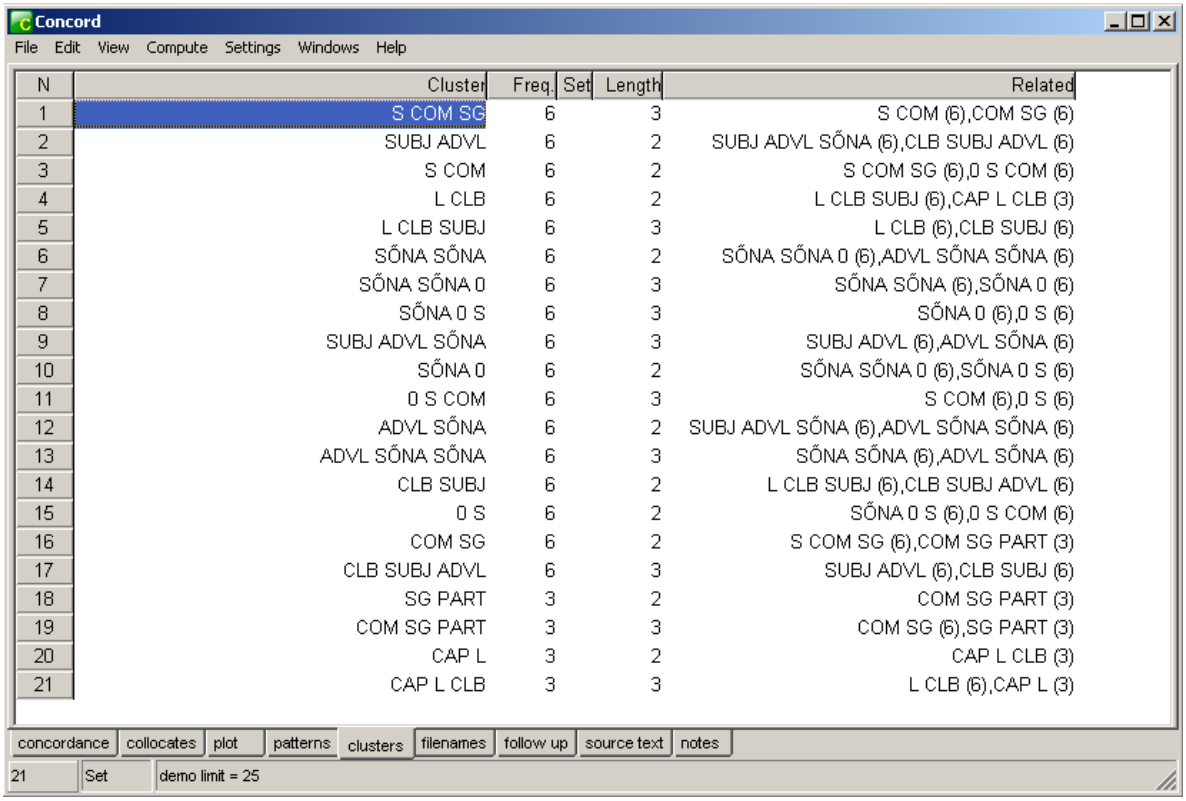
Tegu on tasulise programmiga, mis pakub väga laia funktsionaalsust. Laia funktsionaalsusega kaasneb kohe keeruline kasutajaliides, mis võib kasutaja alguses ära ehmatada. (Ari, 2006) Kui lähemalt hakata uurima, saab ka kollokatsioone selle programmiga leida. Programm leiab kollokatsioonid hõlpsalt. Küllaga teeb ta seda sõnade otsimise järgi, mitte sõna morfosüntaktiliste märgendite järgi.

Sisestades näitelause ja näitelause morfoanalüsaatorist läbi töödelduna saame sellised tulemused[Tulemus 2](otsides kuni kahe sõnalisi klastreid ja miinimum korduste arvuks on valitud kaks).



N	Cluster	Freq.	Set	Length	Related
1	SÕNA JÄRJEST	7		2	I SÕNA JÄRJEST (3),SÕNA JÄRJEST VIIS (3),VIIS SÕNA JÄRJEST (2),KOLM SÕNA JÄRJEST
2	SÕNA JÄRJEST VIIS	3		3	SÕNA JÄRJEST (7),JÄRJEST VIIS (3)
3	NELI SÕNA JÄRJEST	3		3	SÕNA JÄRJEST (7),NELI SÕNA (3)
4	NELI SÕNA	3		2	NELI SÕNA JÄRJEST (3),JÄRJEST NELI SÕNA (3)
5	JÄRJEST VIIS	3		2	SÕNA JÄRJEST VIIS (3),JÄRJEST VIIS SÕNA (2)
6	JÄRJEST NELI SÕNA	3		3	NELI SÕNA (3),JÄRJEST NELI (3)
7	JÄRJEST NELI	3		2	JÄRJEST NELI SÕNA (3),SÕNA JÄRJEST NELI (2)
8	VIIS SÕNA JÄRJEST	2		3	SÕNA JÄRJEST (7),VIIS SÕNA (2)
9	VIIS SÕNA	2		2	VIIS SÕNA JÄRJEST (2),JÄRJEST VIIS SÕNA (2)
10	SÕNA JÄRJEST NELI	2		3	SÕNA JÄRJEST (7),JÄRJEST NELI (3)
11	KOLM SÕNA JÄRJEST	2		3	SÕNA JÄRJEST (7),KOLM SÕNA (2)
12	KOLM SÕNA	2		2	KOLM SÕNA JÄRJEST (2)
13	JÄRJEST VIIS SÕNA	2		3	JÄRJEST VIIS (3),VIIS SÕNA (2)

Tulemus 2 - Näitelause WST'iga läbi töödeldud



N	Cluster	Freq.	Set	Length	Related
1	S COM SG	6		3	S COM (6),COM SG (6)
2	SUBJ ADVL	6		2	SUBJ ADVL SÕNA (6),CLB SUBJ ADVL (6)
3	S COM	6		2	S COM SG (6),0 S COM (6)
4	L CLB	6		2	L CLB SUBJ (6),CAP L CLB (3)
5	L CLB SUBJ	6		3	L CLB (6),CLB SUBJ (6)
6	SÕNA SÕNA	6		2	SÕNA SÕNA 0 (6),ADVL SÕNA SÕNA (6)
7	SÕNA SÕNA 0	6		3	SÕNA SÕNA (6),SÕNA 0 (6)
8	SÕNA 0 S	6		3	SÕNA 0 (6),0 S (6)
9	SUBJ ADVL SÕNA	6		3	SUBJ ADVL (6),ADVL SÕNA (6)
10	SÕNA 0	6		2	SÕNA SÕNA 0 (6),SÕNA 0 S (6)
11	0 S COM	6		3	S COM (6),0 S (6)
12	ADVL SÕNA	6		2	SUBJ ADVL SÕNA (6),ADVL SÕNA SÕNA (6)
13	ADVL SÕNA SÕNA	6		3	SÕNA SÕNA (6),ADVL SÕNA (6)
14	CLB SUBJ	6		2	L CLB SUBJ (6),CLB SUBJ ADVL (6)
15	0 S	6		2	SÕNA 0 S (6),0 S COM (6)
16	COM SG	6		2	S COM SG (6),COM SG PART (3)
17	CLB SUBJ ADVL	6		3	SUBJ ADVL (6),CLB SUBJ (6)
18	SG PART	3		2	COM SG PART (3)
19	COM SG PART	3		3	COM SG (6),SG PART (3)
20	CAP L	3		2	CAP L CLB (3)
21	CAP L CLB	3		3	L CLB (6),CAP L (3)

Tulemus 3 - Morfoanalüsaatoriga läbi töödeldud näitelause tulemus

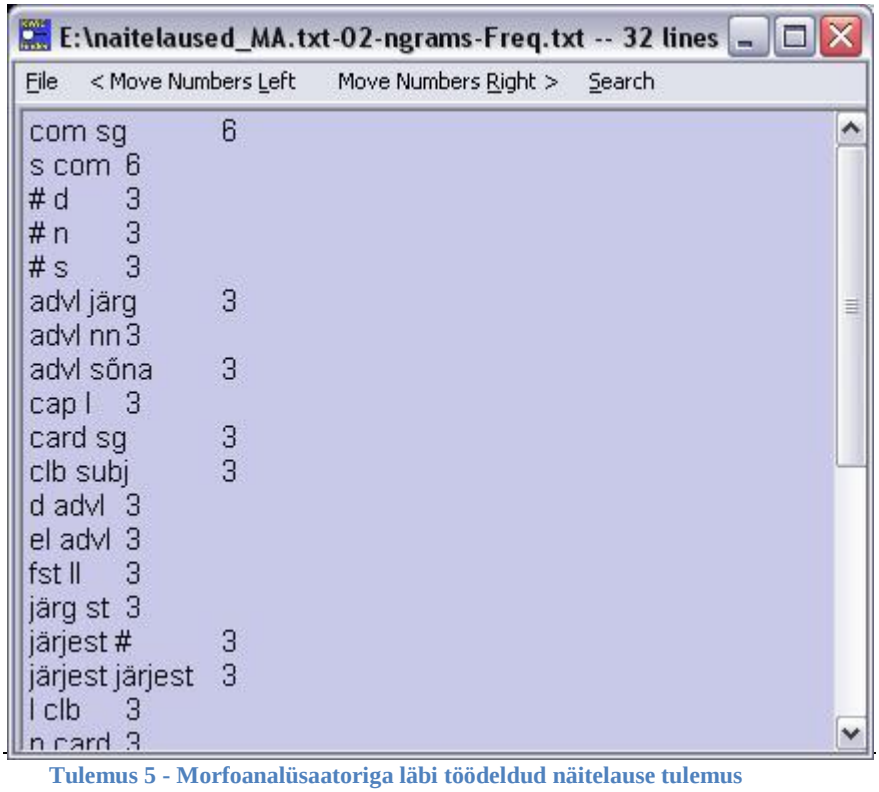
Kui tulemusi uurima hakata, selgub et programm käsitleb morfosüntaktilise märgendi osi erinevate sõnadena ja seetõttu ei ole tulemus see, mida vaja on[Tulemus 3]. Wordsmith Tools 5 ei täida kõiki meie vajadusi.

kfNgram

Teine programm, mida uurime on kfNgram, mis on tasuta allatiritav ja kasutab N-gram meetodit. Programm ei võimalda niipalju, kui WordSmith Tools 5 võimaldab. Kuid siin on jällegi võlu. Kuna programmil on vähem funktsioone, on teda lihtsam õppida ja mõista. Rakendus võimaldab kasutajal hõlpsalt leida klastreid[Tulemus 4].

Tulemus 4 - Näitelause kfNgram'iga läbi töödeldud

Nüüd vaatame olukorda, kus sisendiks on sama lause morfoanalüsaatoriga läbi töölded.[Tulemus 5]



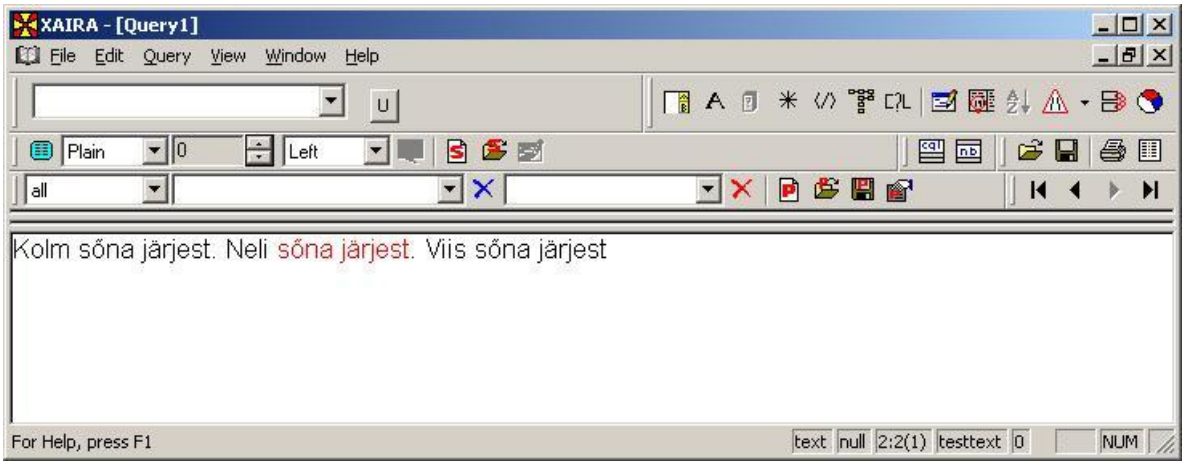
Tulemus 5 - Morfoanalüsaatoriga läbi töödeldud näitelause tulemus

Isegi teksti õigele kujule viies leiab antud programm ka morfosüntaktiliste märgendite järgi klastrid. Küll aga see on pika töö järel. Kuna eesmärgiks on saada täisautomaatne programm, ei sobi kfNgram meile.

Xaira

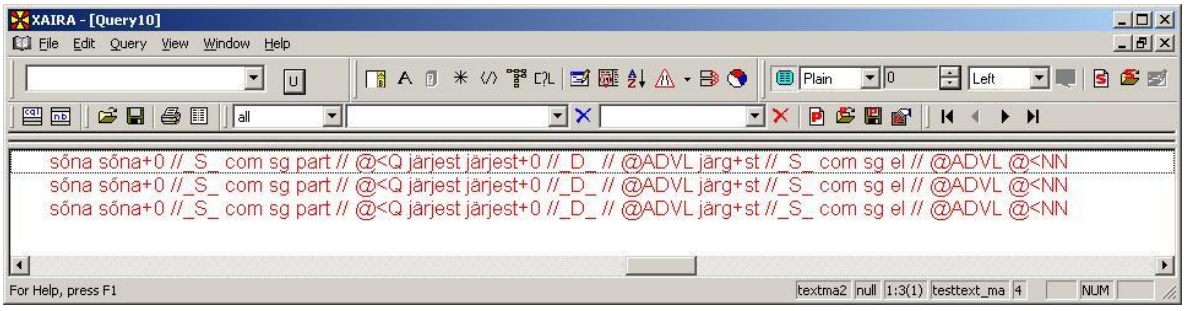
Kolmas programm, mida vaatame on Xaira. Tegemist on XML otsingumootoriga, mida on võimalik ka kasutada klastrite ja kollokatsioonide otsimiseks. Et programmi kasutada, peab alguses oma tekstid indekseerima Xaira'ga kaasas oleva programmiga, mis teeb selle kasutamise ebamugavaks. Pärast indekseerimist on võimalik otsida programmist sõnade või fraaside korduste arvu. Kõiki klastreid automaatselt ta leida ei oska. (Ari, 2006)

Indekseerides näite lause sai järgmise tulemuse. Antud juhul otsisin „sõna järjest" tekstist[Tulemus 6].



Tulemus 6 - Näitelause programmis Xaira

Otsides morfosüntaktilisi märgendite abil esines raskusi. Kuna Xaira oskab ainut fraasi järgi otsida, et kasutaja peab teadma, mida ta otsib. Otsides sõnu "sõna sõna+0 //_S_ com sg part // @<Q järjes järg+s //_S_ com sg in // @ADVL @<NN". Seda otsides näeme, et kasutajal ei ole mugav seda sisestada.



Tulemus 7 - Näitelause märgendite otsing Xaira programmis

Xaira oskab küll fraaside järgi korduseid otsida, küllaga mitte kõiki korraga. Et teada saada tekstis kõige enim levinumad klastrid, läheks kasutajal tohutult aega.

3. Programmist

Selles peatükis tooks täpsemalt välja mida programm peaks kõike teha suutma.

Programm peab sisaldama

- Teksti lisamise võimalust
- Funktsiooni, mis leiab tekstist morfoloogilised/süntaktilised klastrid
- Võimalus otsida tekstist morfoloogilisi/süntaktilisi kollokatsioone
- Suudab kuvada leitud tulemusi
- Salvestamise ja uuestiavamise võimalus
- Tulemuste eksportimine tekstifaili/exeli faili
- Graafilist kasutajaliidest

Kasuks tuleb

- Operatsioonisüsteemist sõltumatu
- Programmi töökiirus

Järgmistes peatükkides kirjutan programmi tehnilisest küljest ja programmi disainist.

4. Programmi tehniline külg

Selles peatükis kirjutan programmeerimiskeele valikust. Samuti kirjeldan erinevaid programmeerimisvõtteid ja põhi algoritmi.

Programmeerimiskeele valik

Programmeerimiskeele valikul tuleks silmas pidada erinevate keelte omadusi. Kuna töökiirus on meil oluline, oleks hea kasutada programmeerimiskeelt C. Kiiremaks mõõdeti kolm programmeerimiskeelt, milleks olid FORTRAN, Ada ja C. (Corlan, 2003) (Language benchmark - Results). Kuid teisest küljest vaadates on oluline ka praktilisus. Ei ole mõtet ratast leiutama hakata. Kuna Java suudab palju enamat kui C, on tugev soov just seda keelt kasutada. Siin silmas pidades graafilist kasutajaliidest ja tabelite genereerimisevõimalust. Kui töökiirus selle all kannatab, võib teatud funktsioonid C-s kirjutada.

Idee poolest sobiks ka keel C++, kuid siiani pole tollega kokku puutunud. Mõistlik on jääda keelte juurde, mida juba mõistan.

Valituks osutub programmeerimiskeel Java. Kui Java hätta jääb, võtab C appi. Java töötab ka enamustel operatsioonisüsteemidel.

Teksti lisamine

Et tekstidest kollokatsioone üldse leida, peab tekst olema sisestatud. Sisestatav tekst peaks olema morfoanalüsaatoriga läbi töödeldud, et sõnade märgendid oleksid olemas. Nupp väärtusega „Sisesta tekst“ avab faililehitseja, kust kasutaja saab faili avada. Samuti peaks olema võimalus teksti lisamine eelneva teksti lõppu.

Teksti mälus hoidmine

Teksti hoida mälus kahemõõtmelise massiivina, kus esimesel kohal on sõna string ja teisel kohal massiiv sõna märgendiga. Siin tuleb küll kitsendus. Kuna Java maksimaalne massiivi pikkus 32bit süsteemis on $2^{31}-1$. Kahtlen küll, et selleni välja jõutakse. Kui ka jõutakse, saab kasutada teist massiivi lisaks esimesele.

Tekstist klastrite leidmise algoritm

Programm peab suutma pikkadest tekstidest morfoloogilised/süntaktilised klastrid hõlpsalt leidma. Seda tagamaks peaks seda tegev algoritm mitmeid kordi läbi mõeldud olema. Algoritm on üles ehitatud N-gram mudeli põhjal.

Algoritmi esimene osa. Klastriminsuurus ja klastrimaxsuurus on kasutaja poolt sisestatud numbrid, mis näitavad kui suures vahemikus klastreid otsitakse. Muutuja k on number, mis tähistab klastri pikkust. Muutuja n on number, mida kasutatakse lugejana terve teksti läbimises (kogu teksti sõnade arv). Muutuja m on ka number, mis vastab tulemuste massiivide suurusega.

Tulemuste massiiv on kahemõõtmeline. Esimeses mõõtmes on klastri järjenumber. Teine mõõde on keerulisem. Teise mõõtme esimesel seitsmel kohal asetsevad sõna märgendid. Kaheksandal kohal asub klastri korduste arv ja pärast seda arvu on sõnalised näited klastri kohta vastavalt korduste arvule. Tulemuste massiiv oleks selline:

- tulemused[102][0] = sõna 1 märgend
- tulemused[102][1] = sõna 2 märgend
- tulemused[102][2] = sõna 3 märgend
- tulemused[102][3] = tühi
- tulemused[102][4] = tühi
- tulemused[102][5] = tühi
- tulemused[102][6] = tühi
- tulemused[102][7] = 2
- tulemused[102][8] = kolm sõna järjest
- tulemused[102][9] = kolm sõna järjest

Selgitus: tegu on klastriga, mille järjenumber on 102. Klaster on kolme sõnaline ja esineb kaks korda. Mõlemal korral oli selleks klastriks „kolm sõna järjest“.

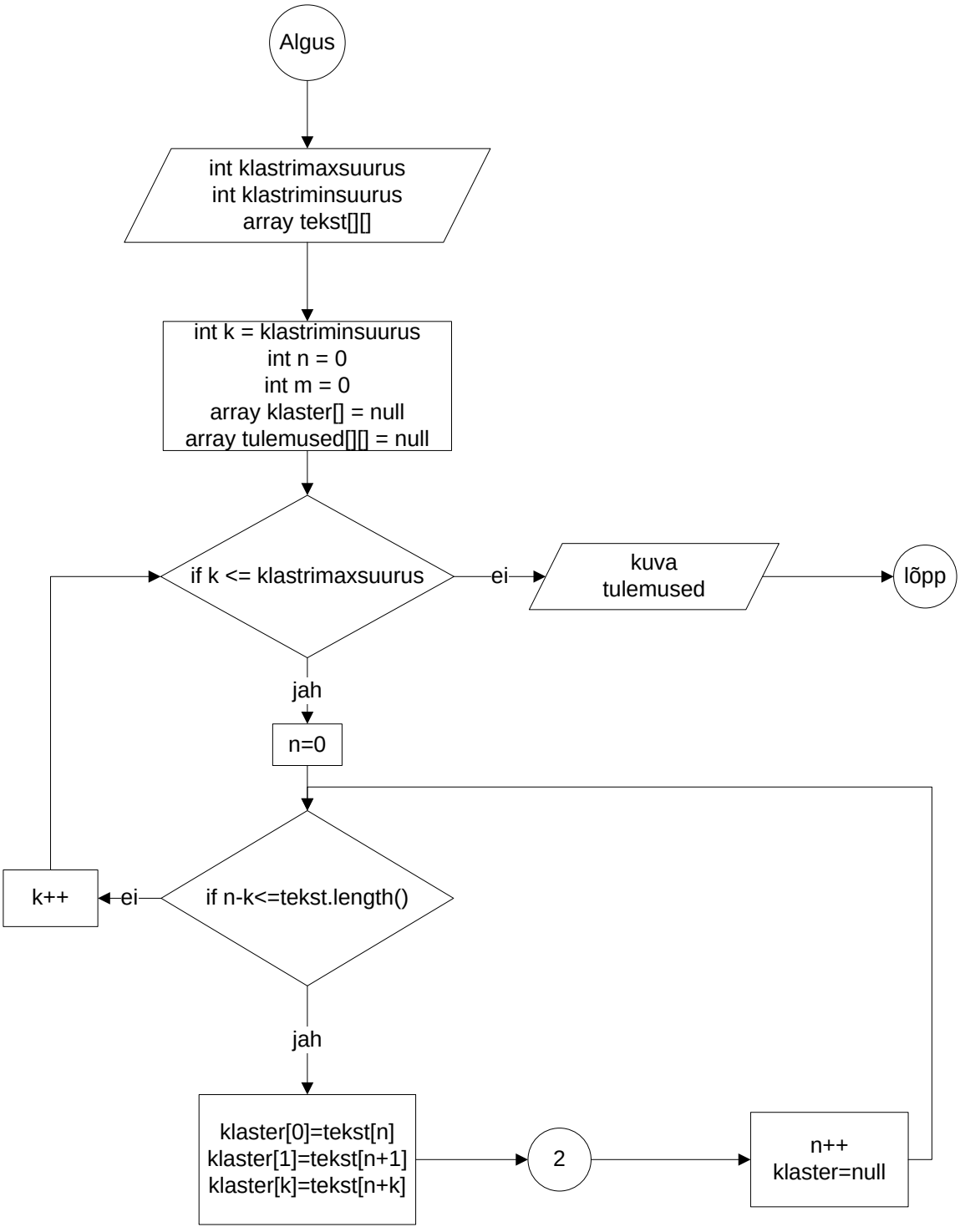
Klaster on samuti massiiv, kuhu kogutakse kokku ajutiselt hetkel käsil olev klaster. Klaster oleks näiteks [sõna 1 märgendid; sõna 2 märgendid; sõna 3 märgendid].

Algoritmi teises osas on kaks abimuutujat. numbrid i ja j on abiks tsüklites.

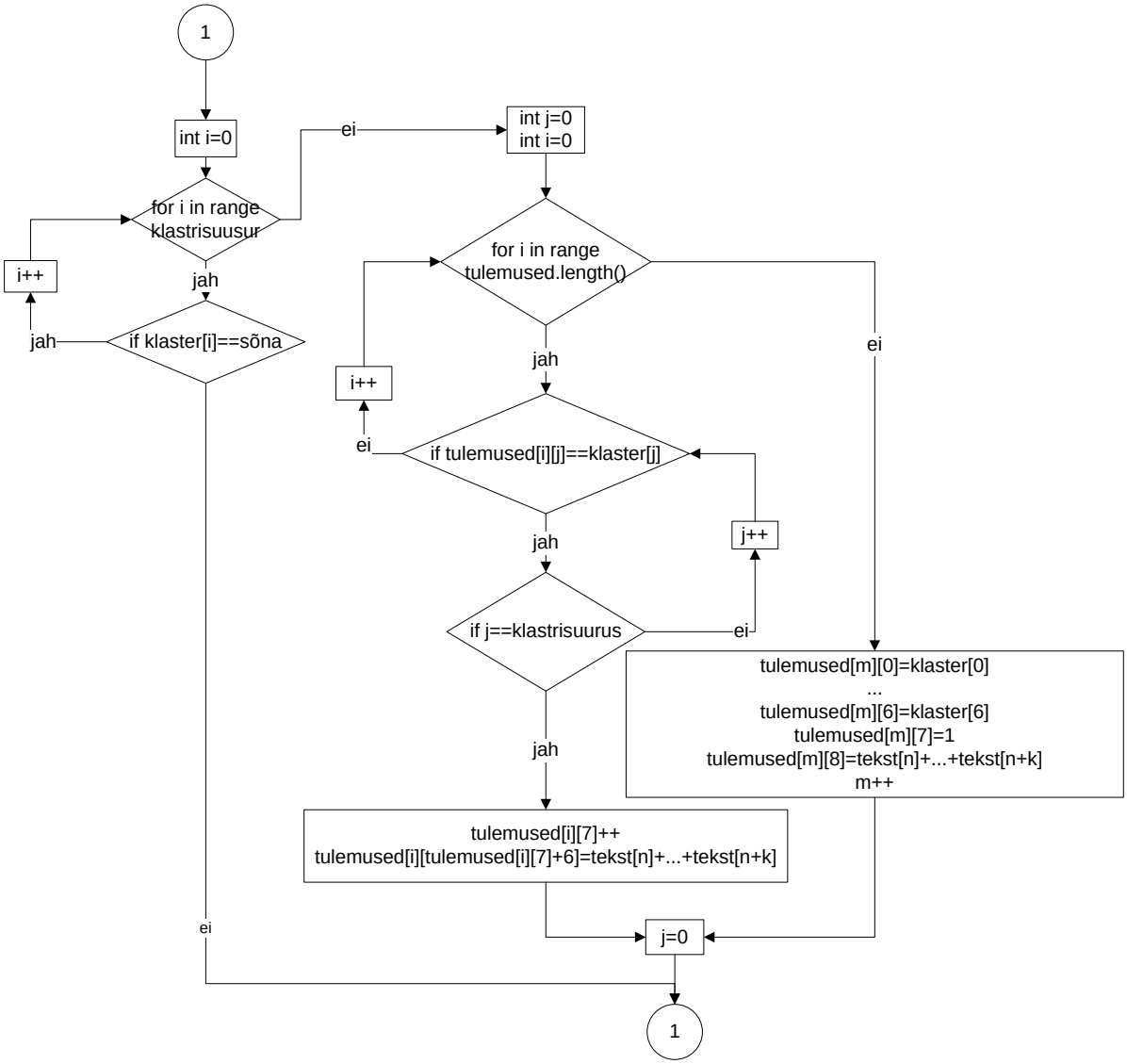
Selgitus

Loetakse sisse kasutaja poolt andmed. Algväärtustatakse vajaminevad väljad. Algab esimene tsükkel. k väärtuseks on kasutaja poolt sisestatud klastri minimaalne suurus. selle

suurusega algab teine tsükkel. Niikaua, kuni n on tekstis olevate sõnade arvust väiksem, tehakse järgmist: Lisatakse klastri massiivi n kohale olevate sõnade märgendid, arvestades muutuja k väärtust. Algab klastri kontroll. Kontrollitakse, kas klastri sisse on sattunud ikka sõnade märgendid. Järgnev tsükkel on tulemuste massiivi kontrollimiseks. Kontrollitakse, ega samade märgenditega tulemust juba olemas ei ole. Kui ei ole, siis lisatakse uus rida tulemuste massiivi. Kui on, siis lisatakse vastavate märgenditega tulemuste massiivi sagedusele üks juurde. Samuti lisatakse tulemuste massiivi ka sõnad, et kasutajal oleks mugavam hiljem vaadata, mis reaalsed sõnad kordusid. Suurendatakse vastavalt loendureid ja tühjendatakse klastri massiiv. Tsüklite lõpus kuvatakse kasutajale tulemused tabelisse.


Algus
int klastrimaxsuurus
int klastriminsuurus
array tekst[][]
int k = klastriminsuurus
int n = 0
int m = 0
array klaster[] = null
array tulemused[][] = null
if k <= klastrimaxsuurus
ei
kuva
tulemused
lõpp
jah
n=0
k++
ei
if n-k<=tekst.length()
jah
klaster[0]=tekst[n]
klaster[1]=tekst[n+1]
klaster[k]=tekst[n+k]
2
n++
klaster=null

Algoritm 1



Algoritm 2

Tulemuste kuvamine

Graafilises Java kasutajaliideses on lihtne luua tabeleid. Tabeli kujul on hea viis algteksti ja ka tulemusi kuvada. Oluline on kasutaja mugavus. Selletõttu oleks vaja kahte erinevat vaadet. Esimeses vaates oleks tekst algsel kujul koos iga sõna märgenditega. Teine vaade oleks kollokatsioonide/või klastrite vaatamiseks. Esimestes lahtrites oleks kollokatsioonid sõnade märgenditega. Tabeli viimane lahter näitaks mitu korda tekstis esines antud kollokatsioon. Need oleks sorteeritud korduste põhjal.

Salvestamine

Salvestamine toimub tavalisse tekstifaili. Faili salvestamise juures tuleb alguses kindlaks teha, mis salvestamist üldse vajab. Tekst, mis sisestatud, peab alles jääma. Samuti ka klastrid, kollokatsioonid ja nende kordused. Seetõttu salvestatud fail koosneks kahest osast. Esiteks tekst koos oma märgenditega kujul „sõna“ tühik „sõna märgendid“ reavahe. Eraldajaks oleksid tühikud ja iga sõna lõppeks reavahega. Teine osa salvestusfailist oleksid klastrid või kollokatsioonid. Kuju võiks olla selline: „Esimese sõna märgendid (eraldusmärk) ... (eraldusmärk) Viimase sõna märgendid (eraldusmärk) Korduste arv. (eraldusmärk) sõnalised näited klastri kohta (eraldusmärk) ... (eraldusmärk). Sarnaselt tulemuste massiivile klastrite leidmise algoritmi juures.

Tõrgete ärahoidmiseks tuleks salvestada ka teksti lisamisel faili. Tekitada ajutine fail, kus on tekst märgenditega olemas. Suurte tekstikoguste puhul hoiab see ebameeldivaid üllatusi ära.

Salvestatud faili avamisel laetakse sõnad koos märgenditega ja kollokatsioonidega mällu ja kuvatakse kasutajale.

Eksportimine

Tulemuste eksportimine Microsoft Office Excelisse käiks CSV[2] faililaiendi kaudu. Kuna selle laiendiga failid on lihtsa ülesehitusega, on nendesse lihtne salvestada. CSV failis on igas lahtris olev element semikooloniga eraldatud. Sellist faili on lihtne luua ja hõlpsalt avatav exceliga.

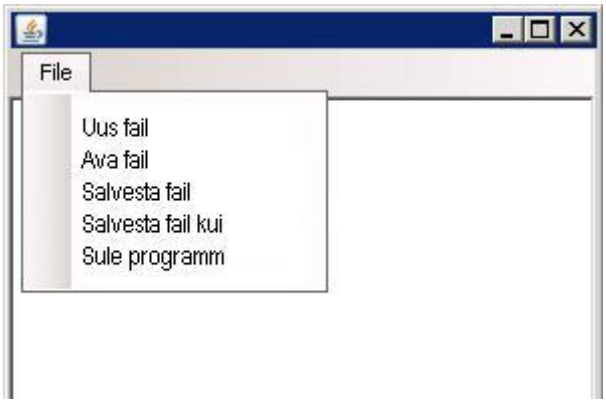
[2] Comma-Separated Values

5. Programmi disain ja süžeetahvel

Kuna tarkvara hakkavad kasutama väheste arvutikasutuskogemustega inimesed, on vajadus graafilise kasutajaliidese järele. Selles peatükis kirjeldan graafilist kasutajaliidest ja toon välja ideid, kuidas oleks kasutajal mugav loodavat programmi kasutada.

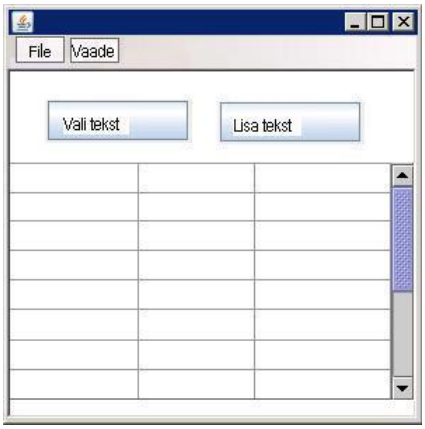
Graafiline kasutajaliides

Kasutaja mugavust silmas pidades tuleb programmile graafiline kasutajaliides. Programmi käivitades tuleb ekraanile raam millel on rippmenüü ja sulgemisvõimalus. Rippmenüüs asuksid: „Uus fail“, „Ava fail“, „Salvesta fail“, „Salvesta fail kui“ ja „Sule programm“.



Pilt 1- Programm käivitades

Uus fail avab uue raam’i. Sellel kasutajaliidese elemendil asuvad nupud: „vali tekst“ ja „lisa tekst“. Samuti on siin tühi tabel. Kui tekst lisatakse, kuvab antud tabel seda.



Pilt 2 - uus fail

Tabelis oleks sisestatud tekst sellisel kujul:

Tabel 1 - Sisestatud teksti tabel

Sõna 1	Sõna 2	Sõna 3	Sõna 4	Sõna 5	**Kerimisriba**
Sõna 1 märgend	Sõna 2 märgend	Sõna 3 märgend	Sõna 4 märgend	Sõna 5 märgend	
Sõna 6	Sõna 7	Sõna 8	Sõna 9	Sõna 10	
Sõna 6 märgend	Sõna 7 märgend	Sõna 8 märgend	Sõna 9 märgend	Sõna 10 märgend	
...	...	...	...	...	

Samuti asetseb sellel kaadril võimalus klastrite/kollokatsioonide raami kallale asuda. Rippmenüü „vaade“ all saab valida kas algteksti või klastrite vaate vahel. Klastrite lehel on kaks tekstilahtrit, kus saab määrata kui suuri klastreid otsida ja nupp selle praktiseerimiseks. Samuti on sellel lehel ka tabel.

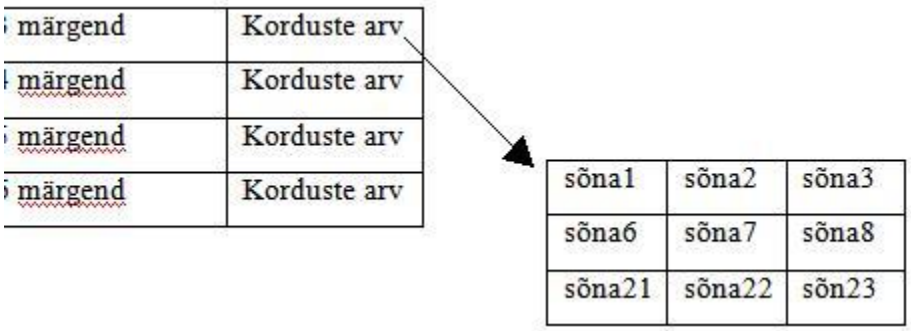
Pilt 3 - klastrite vaade

Antud tabelit kasutades oleks kasutajal mugav näha klastrite korduste arvu ja samuti igat klastrit (morfosüntaktiliste märgenditega) näha:

Tabel 2 - Kollokatsioonide tabel

Sõna 1 märgend	Sõna 2 märgend	Sõna 3 märgend	Korduste arv	**Kerimisriba**
Sõna 2 märgend	Sõna 3 märgend	Sõna 4 märgend	Korduste arv	
Sõna 3 märgend	Sõna 4 märgend	Sõna 5 märgend	Korduste arv	
Sõna 4 märgend	Sõna 5 märgend	Sõna 6 märgend	Korduste arv	
…	…	…	…	

Kollokatsioonid oleksid korduste alusel sorteeritud. Kusjuures, kui korduste arvu peal klõpsata, peaks ta tooma uue tabeli, kus on sõnadega iga kordus välja toodud [Pilt 4].



Pilt 4 - Hetkepilt

Kokkuvõte

Antud Seminaritöö on aluseks tarkvara programmeerimisel. Üheks töö eesmärgiks oli võtta kõige tuntumad antud valdkonna programmid ning testida, kas nende abil oleks võimalik otsida morfoloogiliste märgendite klastreid. Töö sisaldab ülevaadet kolmest programmist ja nende puudustest morfoloogiliste klastrite otsimisel. Nii Wordsmith tools 5, kfNgram kui Xaira leidsid väga hõlpsalt tekstidest kollokatsioonid, kuid seda ainult sõnade abil. Lähtudes sellest, et sõnade klastrite ja kollakotsioonide otsingu põhimõtted on ka võimalikes teistes selleks otstarbeks loodud tarkvarades samad, on antud seminaritöös tehtud järeldus, et selleks et leida just morfoloogilised struktuursed kooslused, peame looma eraldiseisva programmi.

Töös käsitletakse loodava tarkvara funktsionaalsust. Uurides potentsiaalsete kasutajate vajadusi sai tähtsamaid võimalusi uuritud ja kirjeldatud. Samuti sai loodud kollokatsioonide ja klastrite leidmiseks algoritm.

Disaini poole pealt oli ilmselge vajadus graafilise kasutajaliides järele. Sai välja toodud ideid, milline programm välja peaks nägema ja kuidas andmeid kuvataks.

Valminud töö on detailseks aluseks tarkvara loomisel ja selle põhjal on võimalik alustada programmeerimist. On läbi mõeldud võimalikud probleemid, mis kindlasti kergendavad edaspidist tööd. Loomulikult ei saa välistada probleemsete kohtade või raskuste teket bakalaureuse töös.

Kokkuvõtteks saab öelda, et seminaritöös püstitatud eesmärgid on edukalt saavutatud. Antud tulemused on heaks aluseks bakalaureusetöös planeeritavaks arendusuuringuks.

Kasutatud kirjandus

Ari, O. (2006, Jaanuar). *REVIEW OF THREE SOFTWARE PROGRAMS*. Retrieved Veebruar 24, 2010, from http://llt.msu.edu/vol10num1/review3/default.html

Corlan, A. (2003). *Programming language benchmarks*. Retrieved Veebruar 22, 2010, from http://dan.corlan.net/bench.html

Kaalep, H. J., & Muischnek, K. Eesti keele püsiühendid arvutilingvistikas: Miks ja kuidas.

Language benchmark - Results. Retrieved Veebruar 22, 2010, from Language Benchmark: http://www.bioinformatics.org/benchmark/results.html

Metslang, H., & Matsak, E. (2010). Kesksete lausekomponentide järjestus õppijakeeles: arvutianalüüsi katse. Eesti Rakenduslingvistika Ühingu aastaraamat (175 - 193). Tallinn.

Müürisep, K. *ESTKG süntaktilised märgendid*. Retrieved Veebruar 26, 2010, from http://www.cs.ut.ee/~kaili/parser/demo/synttags.html

Müürisep, K. *ESTKG süntaktilised märgendid*. Retrieved November 4, 2011, from http://www.cs.ut.ee/~kaili/parser/demo/morftags.html

Müürisep, K. *Morfoloogilised märgendid*. Retrieved November 4, 2011, from http://www.cs.ut.ee/~kaili/parser/demo/morftags.html

Smadja F. A & McKeown, K. R. (1990). Automatically extracting and representing collocations for language generation.