Tallinna Ülikool
Informaatika Instituut

Probleemsed osalaused tekstide esitamisel lausearvutusvalemite abil

Bakalaureusetöö

Autor: Indrek Ruubel
Juhendaja: Erika Matsak

Autor: .....”.....”.....2011 a.
Juhendaja: .....”.....”.....2011 a.
Instituudi direktor: .....” .....” .....2011 a.

Tallinn 2011

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud.

.....	.....
(kuupäev)	(bakalaureusetöö kaitsja allkiri)

Sisukord

Sissejuhatus

Käesoleva bakalaureusetöö teemaks on probleemsed osalaused tekstide esitamisel lausearvutusvalemite abil.

Aastate jooksul on eesti filoloogid ja arvutiteadlased teinud koostööd eesti keele uurimiseks. Sellest sümbioosist filoloogia ja arvutiteaduse vahel arenes välja keeletehnoloogia. Antud teadusvaldkonna eesmärk on muuta keele uurimist lihtsamaks ja automatiseeritumaks. Näiteks kui varem oli filoloog kohustatud lausest sõnade morfoloogilist (sõnavormi) tähendust käsitsi välja kirjutama, siis nüüdseks on see väga suurel määral juba automatiseeritud. Just sellise analüsaatori olemasolu oli väga oluline käesoleva bakalaureusetöö valmimisel. Paraku leidub veel juhtumeid, kus arvuti ei suuda analüüsi tulemusena õiget vastet leida. Selle probleemi lahendamisega tegeletakse samm-sammult, tehes uurimistööd ja täiendades vastavat tarkvara.

Tänu morfoloogilisele analüüsile võib igast sõnast lauses tuletada süntaktilise rolli ja lähtudes sõnade grammatilistest vormidest leida loogikavalemite põhikomponente – indiviide ja predikaate.

Eelnevalt kirjeldatud tarkvara arendusega on tegelenud Erika Matsak. Tarkvara kannab nime dialoogisüsteem DST. Lühidalt kokku võttes suudab antud tarkvara lausete morfsüntaktilise analüüsi tulemusest saada predikaatloogika valemeid. Aga paraku leidub ka siin juhtumeid, kus arvuti ei tee õigeid otsuseid ja seda just osalause tasandil. Seega on vaja tarkvara õpetada. Aga selleks, et õpetada on vajalik eelnevalt palju teksti analüüsida. Et leida olukorrad, kus ja miks arvuti eksib. Lõppkokkuvõttes peaks DST olema praegusega võrreldes iseseisvam ja nõudma vähem inimese sekkumist, see tähendab, et on tarvis liikuda dialoogrežiimist üle automaatsele režiimile.

Selle bakalaureusetöö eesmärgid on:

- Luua tarkvara, mis toetab probleemsete osalausete struktuuri uurimist.
- Klassifitseerida probleeme, mis on seotud mitme predikaadi kasutamisega ühes osalauses.
- Klassifitseerida probleeme, mis on seotud mitme sõna kasutamisega ühe indiviidina.

Antud uurimustöö sihtrühmad on eesti keele filoloogid ja teadlased, kes uurivad keele ja loogikaga seotud aspekte.
Käesolevas töös uuriti ilukirjanduslikke tekste ning tekstid on võetud ilukirjanduse korpusest. „*Arvutiajastul on korpusena hakatud mõistma peamiselt polüfunktsionaalseid elektroonilisel kujul olevaid tekstikogusid. Seega mõeldakse tänapäeva arvutiajastul korpuse all peamiselt polüfunktsionaalseid elektroonilisel kujul olevaid tekstikogusid, millesse kuuluvad tekstid on valitud eesmärgipäraselt, nii et nendest koosnev tervik annaks tõepärase pildi kogu keelest.*" (http://www.cl.ut.ee/kursused/korp_ling01, Kadri Muischnek, 13.04.2011)

Uurimismeetodid, mida töös kasutatakse on rakendust loov uurimus (*Design research*) ja andmekaeve (*Data mining*).

1 Keele, morfo-süntaktilise analüüsi ja loogika vahelised seosed

Antud peatükis käsitletakse keele ja loogika vahelist seost. Selgitatakse lauseehituses olevate sõnade rolle ja koostatakse nendest valemeid. Selgitatakse morfoloogilist analüüsi, süntaktilist analüüsi, tekstide transformeerimist ja antakse ülevaade olemasolevast tarkvarast. Kirjeldatud aspektid on olulised mõistmaks DST toimimist.

1.1 Keel ja loogika

„Selleks et looduses saaks üks olend teisele infot anda, või et mingi olend (inimene, loom või masin) saaks infot vastu võtta, seda säilitada ja töödelda, läheb vaja füüsilise keskkonna eristatavaid seisundeid, näiteks helivõnkeid õhus või vees, erinevate sageduste ja amplituudidega laineid elektromagnetväljas, erineva tugevusega magneeditud alasid magnetmaterjalidel (lindid, kettad) jne. Ilma nende eristatavate seisunditeta pole võimalik midagi kuulda ega näha, ette lugeda ega üles kirjutada." (Lorents P. 2000)

Oma raamatus „*Keel ja loogika*" selgitab Lorents kuidas näha inimeste vahelises keeles loogikat. Ta seletab lahti, millised sõnad lauses kannavad erilist loogilist rolli ning kuidas neid leida. Lorents näitab, et erinevatel loogikavalemites kasutavatel sümbolitel on kindel tähendus. Sümbolite tähenduse aluseks on matemaatiline loogika, täpsemalt predikaatarvutuse loogika (Curry 1963). Antud harus tuuakse esile erinevate kokkulepitud sümbolite rollid ja tähendused. Tähtsamaid rolle on 6 ja autor toob need lühidalt välja koos kirjelduste ja näidetega:

Indiviidid- ehk objektid

Sõnad lauses, mis tähistavad üksikuid objekte või subjekte. Suurel määral on tegemist nimisõnadega.

Lihtne näide: „Auto on ilus" - Siin lauses on indiviidi rollis sõna „auto", sest räägitakse ühest konkreetsest objektist. Indiviidi andmeobjekte on võimalik esitada indiviidkonstantidena (fikseeritud väärtus), kui ka indiviidmuutujatena (muutuv väärtus).

Predikaadid- ehk seosed

Selles rollis olevad sõnad tähistavad indiviidide omadusi või loovad seoseid indiviidide vahel. Eesti keeles on predikaatide rollis peamiselt tegusõnad ja omadussõnad. Seejuures tuleb märkida, et predikaat võib olla esitatud ka keerulisema struktuurina: näiteks mitme tegusõnana, tegusõna „olema“ vormidena koos nimetavas käändes omadussõna või nimisõnaga (Matsak 2005)

Lihtne näide: „Auto on ilus” - Siin lauses saab predikaadi rolli „on ilus” sõnade kombinatsioon ehk „olema“ vorm pluss omadussõna.

Funktsionaalrollid – ehk tehted

Andmeobjektideks võivad olla funktsionaalkonstandid (nt. aritmeetikas +) ja funktsionaalmuutujad. Nii saabki lauses tuua näitena mingisuguse aritmeetilise tehte:

5 + 5 = 10.

Loogilised operatsioonid

Diskreetsest matemaatikast on tuntud 5 tähtsamat loogilist operatsiooni, mida saame tavakeelel rakendada. Antud operatsioonid toodavad vastavalt *Boole*'i loogikale kas Tõese (1) või Väära (0) vastuse. Nende loogiliste operatsioonide abil moodustatakse tarkvaras DST lausearvutusvalemeid.

Negatsioon ehk eitus (¬)	„*Mina ei taha süüa*” – siin lauses eitatakse soovi söömise vastu. Antud lause saab tõene olla, kui tõesti indiviid „*mina*” ei taha süüa. Näidislauses tähistab negatsiooni „*ei*”.
Konjunktsioon (&)	„*Mina võtsin päikest ja Peeter sõi suppi*” – kaks eraldi sündmust, mida seob konjunktsioon (&). Konjunktsiooni tähistab lauses „ja” ning mõlemad väited peavad vastama tõele, et loogilise operatsiooni väärtus oleks tõene.
Disjunktsioon (∨)	„*Peeter vaatas televiisorit või tegi kodusid töid*” – kaks tegevust, millest vähemalt üks väide peab vastama tõele, et loogilise operatsiooni väärtus oleks tõene. Näidislauses tähistab disjunktsiooni sõna „või”.
Implikatsioon (⊃)	„*Kui ilm on ilus, siis võib päikest võtta*” – Loogilise operatsiooni väärtus saab olla väär ainult sellisel juhul, kui tõesest väitest

osutub väär väide, ülejäänud juhtudel on loogilise operatsiooni väärtus tõene. Näide väärast loogilise operatsiooni väärtusest: „*Kui ilm on soe, siis ilm on külm*”. Näidislauses tähistab implikatsiooni „kui...,siis...”.

Ekvivalents (⇔) „*kaks pluss kaks võrdub neli on samaväärne kaks korda kaks võrdub neljaga*” – Loogilise operatsiooni väärtus on tõene juhul, kui mõlema väite väärtus on võrdne, ehk kas mõlemad väited on tõesed või väärad. Näidislauses tähistab ekvivalentsi „on samaväärne”.

Samuti mängivad lausearvutusvalemite moodustamisel tähtsat rolli 2 **kvantorit**:

Üldsuskvantor (∀) „*Iga mees sureb*” – Absoluutne fakt, mis on tõene iga kord. Mingi lõpliku hulga iga element omab ühist omadust. Üldsuskvantorit tähistab antud lauses „iga”.

Eksistentsikvantor (∃) „*Leidub juhtumeid, kus õpilane jätab õpetaja nõu kuulda võtmata*” – Võib leiduda juhtumeid, kus juhtub teisiti. Mingis lõplikus hulgas leidub elemente, millel on ühised omadused. Eksistentsikvantorit tähistab näidislauses „leidub”.

Punktuatsioonisümbolid ehk abisümbolid

Abisümbolid on kõik kirjavahemärgid, sulud jne, mis esinevad kindlasti pikkades tekstides. Nende abil eristatakse paljudel juhtudel lauseid üksteisest, aga mitte iga kord, näiteks kuupäevi esitatakse vahel kujul „15. aprill”. Selline esitusviis tekitas antud tarkvaraarendusel mõningaid probleeme. Tekkinud probleemidest räägitakse täpsemalt peatükis „2.3 Tarkvaraarendusel esinenud probleemid”.

Modaalsused

Modaalsused on olukordi iseloomustavad lauseosad, mis võivad olla esitatud kahel kujul.

1) Ilmtingimata, igal juhul, alati, ...

2) Võib-olla, võimalik, pole välistatud, ...

Eelnimetatud loogilisi rolle saame rakendada predikaatloogika valemi moodustamisel. Lauseid, mis on esitatud loogikavalemitega, on võimalik analüüsida matemaatiliste

meetoditega. Näiteks kui on mingi arutlus esitatud loogikavalemite abil, siis saame kiiresti selgusele jõuda, kas antud arutlus on loogiliselt korrektne või mitte.

1.2 Loogikavalemid

Eelnevate loogiliste rollidega saab esitada lauseid matemaatiliste valemitena. Kui alustada indiviididest, siis nende andmeobjektid võivad esineda indiviidmuutujatena/konstantidena, predikaadid võivad esineda predikaatmuutujatena/konstantidena. Näide: „*Auto on ilus*." – selles lauses saame teada, et indiviid on „*auto*". Määrame muutujale a väärtuseks „*auto*", ning predikaatmuutujaks P saab „*on ilus*". Valemi kujul näeb see lause välja P(a). Siin on tegemist ühekohalise predikaadiga, ehk predikaat P võtab vastu ainult ühe argumendi. Analoogselt on üles ehitatud ka kahekohalised, kolmekohalised, n-kohalised predikaadid. Muudame näiteks toodud lauset, kasutades eitust : „*Auto ei ole ilus*", sellele lausele vastab valem ¬P(a). Üritame erinevaid rolle kombineerida, illustreeriv näide: „*Kui Iga auto ei ole ilus, siis kõik inimesed on erinevad*". Valemina: $\neg(\forall aP_1(a)) \supset(\forall iP_2(i))$.

Siin valemis on järgmised rollid:

a – „*auto*"

$\forall$a – „*Iga auto*"

$P_1()$ – „*ilus olema*"

$\neg P_1()$ – „*ei ole ilus*"

$\supset$ - „*siis*"

i – „*inimene*"

$\forall$i – „*kõik inimesed*"

$P_2()$ – „*erinev olema*"

Muidugi võib veel palju näiteid tuua, kuid piirdume siinkohal lihtsate näidetega, kuidas loogilisi rolle tekstidel rakendada ja tekste valemite abil esitada.

1.3 Teksti transformeerimine

Kuna laused võivad eesti keeles väga erineval moel üles ehitatud olla ja esineb palju sõnu, mille ärajätmine või asukoht ei muuda lause tähendust, siis saab lauseid muuta, et nendest arusaamist oleks masinale lihtsam õpetada. Vaatleme protseduuri, mille käigus on võimalik tekste transformeerida, ilma et esialgne tekst oma algset tähendust kaotaks. Käsitletavat

protseduuri kirjeldas P. Lorents oma raamatus (Lorents P. 2000) Protseduur jaguneb seitsmeks vaheetapiks:

- ümberpaigutamine (ümberpaigutuste tegemine tekstis)
- täiendamine (uute osade lisamine)
- taandamine (olemasolevate osade eemaldamine)
- asendamine (olemasoleva osa asendamine uuega)
- sümbolite väljaeraldamine (nende tekstiosade väljatoomine, millel on mõne loogika alfabeedi sümboli roll)
- sümbolite liigitamine (sümbolitena väljaeraldatud tekstiosade rolli täpsustamine – näiteks kas on tegu mõnd objekti tähistava sümboliga, objektide omadust või objektidevahelist seost või hoopis loogilist operatsiooni tähistava sümboliga vms)
- sümbolite positsioneerimine (sümbolite paigutamine sinna, kus nad loogilistes konstruktsioonides peaksid korrakohaselt paiknema).

Näited:

"*Lõpuks ometi on kiisupoeg leidnud sõbra*" → Täiendamine → "*Enne kiisupojal ei olnud sõpra ja lõpuks ometi on kiisupoeg leidnud sõbra*" → Asendamine → "*Enne kiisupojal ei olnud sõpra ja nüüd on kiisupoeg leidnud sõbra*" → Ümberpaigutamine → "*Enne kiisupojal ei olnud sõpra ja nüüd kiisupoeg on leidnud sõbra*" → Asendamine → "*Enne kiisupojal ei olnud sõpra ja nüüd kiisupojal on olemas sõber*" → Taandamine → "*Enne kiisupojal ei olnud sõpra ja nüüd kiisupojal on sõber*" → $\neg A_1 (x_1, x_2, t_0) \& A_1(x_1, x_2, t_1)$, kus A_1 − olema,
x_1 – kiisupoeg, x_2 – sõber, t_0 − esimesel ajahetkel(esialgselt), t_1 − praegusel ajahetkel (nüüd). (Matsak E. 2010)

Nagu näidetest näha, siis kaotati algsest tekstist osa sõnu ära, toodi osa asemele ja tõsteti lauset ümber. Viimases lõplikus valemis esines isegi kolmekohaline predikaat. Tänu transformeerimisele on lausele lihtsam matemaatilist loogikat rakendada ja arvuti suudab seda paremini enda jaoks tõlgendada. Samuti vähendab transformeerimisprotsess ka veaprotsenti lausete tõlgendamises. Näited ei katnud kõiki transformeerimisprotsessi etappe, aga nendega on võimalik lähemalt tutvuda Matsak E. publikatsioonis.

1.4 Morfoloogiline analüüs

Morfoloogilise analüüsi ülesandeks on uurida sõnavorme. Leitakse sõna tüvi, sõnaliik, kääne või pööre jms. See saab olema aluseks ka süntaktilisele analüüsile ja samuti aluseks kogu lause transformeerimisele. Kuna lauset saab eesti keeles üles ehitada mitmel viisil ja sama sõna võib erinevates kontekstides erinevaid tähendusi omada, siis on see ülesanne raskendatud. Muidugi saab tööd alati käsitsi teha, aga see on väga kulukas ja vaevarikas, kui sisendlauseid on rohkem kui näiteks 2000. Kui ühel sõnal on mitu tähendust erinevates kontekstides, siis peab miski või keegi valima nendest olemasolevatest võimalustest selle ainukese õige, mis on samuti morfo-analüsaatori ülesanne.

Näide:
„Talle”
"tall" L0 S com sg adit cap @ADVL #1->3
- "tema" Llle P pers ps3 sg all cap @ADVL #1->3
„ei”
- "ei" L0 V aux neg cap @NEG #2->3
„meeldinud”
- "meeldi" Lnud V main indic impf ps neg cap <FinV> <Intr> <All> @FMV #3->3
„see”
- "see" L0 P dem sg nom cap @NN> #4->5
„mõte”
- "mõte" L0 S com sg nom cap @SUBJ #5->3

Sisendlauseks on „*Talle ei meeldinud see mõte*”, igale sõnale on järele kirjutatud võimalikud morfoloogilised vasted ja kriips on autori poolt hiljem ette tõmmatud õigele. Esimene sõna on hea näide sellest, kuidas sõna tähendus võib erinevates kontekstides täiesti erinev olla. „*tall*” – ehk lambatall (nominatiiv) ja „*talle*” ehk temale (allatiiv). Samuti sisaldab „*lambatalle*” morfoloogiline vaste substantiiv-apellatiivi „S com sg adit”. Morfoloogiliste vastete märgendite tähendused võib leida K. Müürisepa kodulehelt (http://www.cs.ut.ee/~kaili/parser/demo/morftags.html). Morfsüntaktiline analüsaator ei suutnud siin valida sõna „*talle*” puhul ainukest õiget vastet. Selle konkreetse analüsaatori eesmärk on leida üks õige vaste, leidub juhtumeid, kus analüsaator seda ei suuda ja väljastatakse kõik vasted, nagu näites juhtus.

1.5 Süntaktiline analüüs

„Loomuliku keele süntaksianalüsaator on programm, mis saab sisendiks morfoloogiliselt analüüsitud teksti (s.t on leitud iga sõna tüvi, lõpud, sõnaliik, kääne või pööre jms) ning väljastab süntaktiliselt analüüsitud teksti (leitud on igas lauses alus, öeldis, sihitis jt lauseliikmed). Enamasti esitatakse süntaktiline kirjeldus märgendite abil, s.t iga sõnavormi juurde kirjutatakse selle sõnavormi morfoloogilisi ja süntaktilisi omadusi kirjeldav märgend või märgendite kombinatsioon.” (Müürisep 2002)

Seega on morfoloogiline analüüs ja süntaktiline analüüs väga tihedalt seotud. Kui morfoloogilist analüüsijat ei oleks olemas või oleks ta manuaalne, siis süntaktiline analüüs ei saaks eksisteerida või tema manuaalne teostamine võtaks kaua aega. Kui vaadata eelmist näidet, mis toodi morfoloogilise analüüsi peatükis:

„Talle”
"tall" L0 S com sg adit cap @ADVL #1->3
- "tema" Llle P pers ps3 sg all cap @ADVL #1->3
„ei”
- "ei" L0 V aux neg cap @NEG #2->3
„meeldinud”
- "meeldi" Lnud V main indic impf ps neg cap <FinV> <Intr> <All> @FMV #3->3
„see”
- "see" L0 P dem sg nom cap @NN> #4->5
„mõte”
- "mõte" L0 S com sg nom cap @SUBJ #5->3

SiiS on näha, et igale morfoloogilisele väljundile järgneb süntaktiline roll, mis on kujul „@A-Z” (A-Z alfabeet). Esimese sõna mõlemad süntaktilised väljundid on määrused ehk adverbiaalid (märgend @ADVL). Teine sõna on eitus, seega süntaktiline roll @NEG. Kolmanda sõna märgend @FMV tähistab finiitset öeldist. Niiviisi on võimalik tuvastada eesti keelest lauseehitusmustreid, kui töödelda suurtes kogustes andmeid. Käesolevas töös arendatav tarkvara otsib mustreid osalausete tasemel. Autor toob välja erinevate märgendite tähendused süntaktilisest analüüsist:

Öeldise märgendid:

@+FMV - finiitne öeldis

@-FMV - infiniitne öeldis

@+FCV - olema liitaegades ning modaalverbid ahelverbides, finiitne vorm

@-FCV - olema liitaegades ning modaalverbid ahelverbides, infiniitne vorm

@NEG - verbi eitus

Põhja märgendid:

@SUBJ - alus ehk subjekt

@OBJ - sihitis ehk objekt

@PRD - öeldistäide ehk predikatiiv

@ADVL - määrus ehk adverbiaal, ka fraasiadverbiaal

Laiendite märgendid:

@AN> - omadus- ja järgarvsõna eestäiendina

@<AN - omadus- ja järgarvsõna järeltäiendina

@AD> - määrsõna eestäiendina

@<AD - määrsõna järeltäiendina

@PN> - kaassõna eestäiendina

@<PN - kaassõna järeltäiendina

@NN> - nimi-, ase- ja põhiarvsõna eestäiendina

@<NN - nimi-, ase- ja põhiarvsõna järeltäiendina

@VN> - partitsiip eestäiendina

@<VN - partitsiip järeltäiendina

@INF_N> - verbi infinitiitne vorm eestäiendina

@<INF_N - verbi infinitiitne vorm järeltäiendina

@<P - eessõna laiend,

@P> - tagasõna laiend

@<Q - kvantori jä rellaiend (kaks meest @<Q),

@Q> - kvantori eeslaiend (inimesi (@Q>) tulvil)

Muud:

@J - sidend

@I - hüüatus

(http://www.cs.ut.ee/~kaili/parser/demo/synttags.html, Müürisep K.)

1.6 DST

Tegemist on Erika Matsaku poolt arendatava tarkvaraga, mis toetub morfsüntaktilisele analüüsile ja transformeerib nn. loomuliku keele teksti matemaatiliseks loogikavalemiks. Uurimustöö sai alguse 2005 aastal ja on täiustunud sellest ajast saati.

1.7 Probleemsed osalaused

Probleemseid aspekte, mida ei suudeta õigesti valemiks transformeerida leidub just osalausetes ja tihti just sellisel juhul, kui tegemist on mitme sama rolli esindajaga. Kas mitu predikaati ühes osalauses või mitu järjestikust indiviidi. Näide predikaadi kohta:

„*Mina sõidan uue autoga*"

Probleemne koht, 2 predikaati.

Ehk transformeeritud lause peaks olema „*Mina sõidan autoga ja auto on uus*".

Soovitud tulemus peaks olema:

P1(q1, q2) & P2(q2)

q1 – „mina"

q2 – „auto"

P1() – „*q1* sõidab *q2*"

& - „ja"

P2() – „*q2* on uus"

Indiviidide puhul tekib probleem, et vahest viitab mitu indiviidi ühele objektile, seega on mõttekas nad üheks indiviidiks kokku liita, sest lause tähendus sellest ei muutu, eriti just valemi tasemel. Näide indiviidi kohta:

„*Proua Bauer oli tore inimene*"

Siin lauses esineb kaks indiviidi, mis viitavad ühele objektile: „*Proua*" ja „*Bauer*". Kuna mõlemad viitavad samale isikule, siis ei ole mõtet neid hoida valemis kahe erineva muutujana. Indiviidide ühendamise automatiseerimisega tegeleb hetkel doktorant Avar Pentel oma doktoritöös („*Üldise kontekstinfo ja loogiliste tuletussammude eristamine loomuliku keele tekstidest automaatses otsustussüsteemis*") ja käesolevas bakalaureusetöös arendatava tarkvara väljund on osa tema uurimustööst.

E. Matsak avastas DST tarkvara loomise algfaasis (Matsak 2004), et laused, mille morfoloogiline ehitus on sama, toodavad ka samasugused loogikavalemid. Sai selgeks, et kui leida kõikvõimalikud probleemsed mustrid, ning määrata neile vastavad õiged loogikavalemid, siis suudaks tarkvara iseseisvalt probleemsetest lausetest korrektseid valemeid moodustada.

1.8 Olemasolev tarkvara morfoloogiliseks analüüsiks

Aastate jooksul on antud valdkonnas tegelenud mitu meeskonda ja on valminud kaks põhilist tarkvara. Autor toob neid välja, kuna nad on olulised selle uurimustöö jaoks.

Kaili Müürisep:

EstCG Parser 1.0a, 2000 TÜ – Antud tarkvara on kohandatud Windows keskkonnale ja teeb sisendlausetega morfsüntaktilist analüüsi. Tarkvarast on tehtud ka edasiarendus Linuxi keskkonnale, mis on antud kasutusele Tallinna Ülikoolile.

Heiki-Jaan Kaalep:

Korpusepäring keeleveebis (Filosoft OÜ) – veebiaadressil www.filosoft.ee asuv eesti keele morfo-analüsaator ja süntesaator. Mugav veebipõhine kasutajaliides, mis võimaldab tekstile morfoanalüüsi, kui ka süntaktilist analüüsi teostada.

2 Tarkvaraarendus

Tallinna Ülikooli serveris on installeeritud Erika Matsaku poolt väljatöötatud dialoogisüsteem DST. Kahjuks leidub veel juhtumeid, kus DST töö ebaõnnestub mõne lause puhul, valemit ei koostata õigesti. Üheks probleemseks allikaks on mitu predikaati ühes osalauses ja mitu järjestikust indiviidi. Autori töö eesmärk on otsida osalausetest mitut predikaati ja vähemalt kahte kõrvuti asetsevat indiviidi, analüüsida ja klasterdada probleeme. Kui need juhtumid on leitud, siis saab uurida neid lauseid põhjalikumalt ja õpetada DST'd neid aksepteerima. See täiustaks transformeerimisprotsessi. Meil on vaja teada saada, miks ja milliste lausete puhul transformeerimine valemiteks DST abil ebaõnnestub, et lõppkokkuvõttes vähendada inimese sekkumist DST töösse.

Ühisel kokkuleppel juhendaja Erika Matsakuga leiti, et antud ülesannet oleks kõige parem Python programmeerimiskeeles teostada. Python keele süntaks on lihtne, kuid mõne jaoks võib-olla ebamugav range koodi treppimine. Samuti on elementaarsemad funktsioonid põhimoodulis juba sisestatud ja on võimalik ühe-rea koodiga teha ära mitu asja. Pythonile on ka väga palju lisamooduleid aktiivse arendajakommuuni poolt tekitatud ja nende liitmine ja kasutamine enda koodis on väga lihtne. Python omab tugevat Windowsi ning Linuxi tuge. Autori Windowsi keskkonnas arendatav kood on mõlemas keskkonnas töötav. Autor kasutas ka vabavaraliselt levitatavaid mooduleid oma eesmärgi saavutamisel.
Autori ajakava nägi välja järgmine:

1. aprill – Teooria kirjutamine valmis
8. aprill – Tarkvara arendus valmis
15. aprill – Klastrianalüüs ja kokkuvõte

2.1 Algoritmi kirjeldus

Loodaval tarkvaral on mitu alaplokki, mida peab läbima. Autor toob välja samm-sammulise algoritmi kirjelduse.

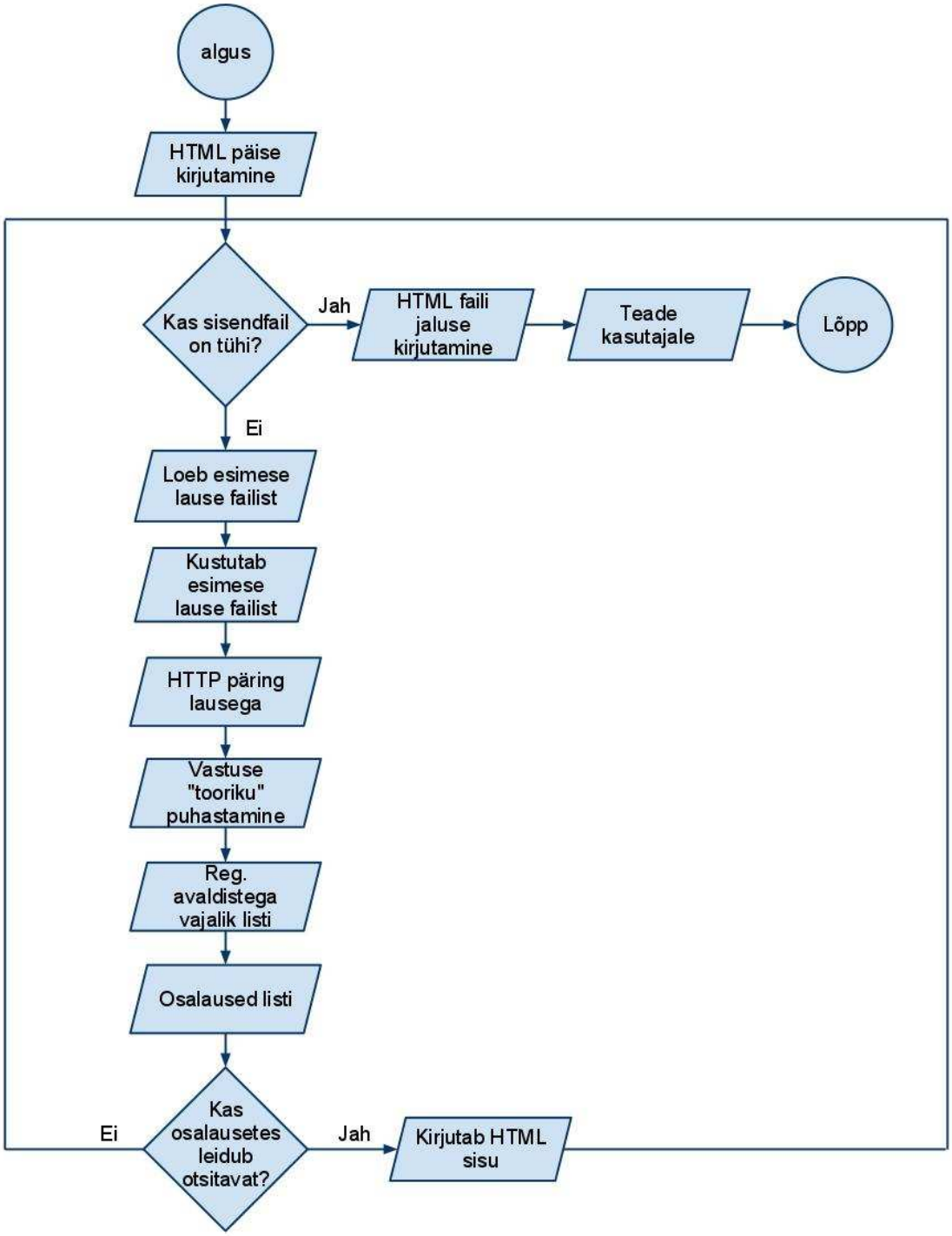
1. Analüüsitavate lausete sisendfaili (txt) esimese lause mällu lugemine ning selle kustutamine failist. Sisendfaili puudumisel või sisu olemasolu puudumisel annab programm veateate või lõpetab tegevuse.

2. Esimese lausega veebipäringu teostamine. Saadetakse lause päringuna veebikeskkonnale evkk.tlu.ee ja saadakse morfsüntaktiliselt analüüsitud vaste.[1]
3. „Tooriku" puhastamine. Ehk veebilehelt saadud vastest eemaldatakse kõik reavahetused, erilised märgid, jutumärgid jne.
4. Regulaaravaldistega lausete eraldamine listi (Kuna hetkel tehakse päring ainult ühe lausega, siis antud programm võimaldab ka mitme lausega seda teha. Algne versioon tegigi mitme lausega päringut, aga see lahendus osutus problemaatiliseks)
5. Regulaaravaldistega eraldatakse lause(te)st kõik sõnad ja tekitatakse alamhulkadega list, nii et indeksid kattuvad.
6. Regulaaravaldistega eraldatakse morfoloogilised vasted alamhulkadena, nii et indeksid kattuvad.
7. Regulaaravaldistega eraldatakse süntaktilised rollid alamhulkadena listi, nii et indeksid kattuvad.
8. Otsitakse lausetest osalauseid. Otsitakse osalausete piiri tähistavat märgistust, neid on kokku kolm („CLB" – sidesõna, „Com CLB" – komana esitatud osalause piir, „CLBC" – oletatav osalause piir). Kui piiri ei leita, siis loetakse terve lause osalausena.
9. Igast osalausest otsitakse kas kahe ja enama predikaadi olemasolu või kahe ja enama järjestikuse indiviidi olemasolu. Kummagi tingimuse tõele vastavusel, kirjutatakse see osaluse või terve lause HTML faili tabeli kujul.
10. Programm läheb algusesse ja teeb kõike uuesti.

[1] evkk.tlu.ee serveris on installitud K. Müürisepa poolt väljatöötatud süntaksanalüsaatori Linuxi versioon

2.2 Plokkskeem

Illustreeriv plokkskeem loodava tarkvara algoritmi kohta. (Joonis 1)



Joonis 1 Rakenduse algoritmi plokkskeem

2.3 Tarkvaraarendusel esinenud probleemid

Programmi loomisel peab väga tähelepanelik olema, sest vead tulevad väga lihtsalt sisse, see on inimlik. Võib juhtuda, et sisendlause saadetakse valel kujul serverile või ei osata dekodeerida tagasi saadud vastust õigesti või tekib erandjuhtumeid, mida esineb väga harva ja nendega on alguses võimatu arvestada, kui isegi nende olemasolu pole veel teada. Need olid mõned näited tekkinud probleemidest, mis selle rakenduse loomisel esinesid. Selles peatükis annab autor ülevaate arendusprotsessist ja raskematest probleemidest.

Autor alustas esmalt projekti hoopis uuema versiooninumbriga Pythoni keskkonnas, nimelt oli selleks versioon 3.1. Tol hetkel oli autor täiesti mitte teadlik, mida see tema jaoks tähendab. Autor luges erinevatest allikatest, et nimelt uuemas versioonis on Unicode transleerimine rohkem automatiseeritud ja rakenduse arendaja ei peagi sellele tähelepanu pöörama [http://docs.python.org/release/3.0.1/whatsnew/3.0.html]. See tulenes varasemast ebameeldivast kogemusest UTF-8'ga. Tundus väga hea võimalus protsessi lihtsustamiseks. Selgus, et see arvamus oli väga petlik. Nimelt, sisendfaili lugedes oli vaja regulaaravaldisega välja eraldada esimene lause. See tähendab seda et, stringi algusest kuni märgenduseni „*.!? A-Z, 0-9*" (punkt/hüüumärk/küsimärk, tühik, Suur algustäht või number). Probleem oli selles, et need tingimused ei pruugi alati kehtida. Mõni lause võib lõppeda punktiga, aga ära võib jääda näiteks tühik ja avaldis ei eristaks enam lause lõppu, näiteks „*U.S.A*". Või ei alga uus lause suure tähega, vaid väiksega, mis võib tuleneda näpuveast. Muidugi saab regulaaravaldises tingimusi OR operaatoriga hulganisti määrata, aga alati võib esineda mingisugune erand, mille puhul avaldis ei kehti. Siin kohal arvab autor, et ülesanne on siiski tehtav Python 3.1 keskkonnas, aga arendusega oli kiire ja ei saanud vigu lubada. Tekkis vajadus alternatiivse lahenduse vastu. Ja autor leidis sellise lisamooduli Pythoni keskkonnale nimega NLTK (Natural language toolkit), tegemist on inglise keeles kasutatava korpuse mooduliga, mis sisaldab kasulikku funktsiooni lause eraldamiseks (*Tokenizer*). Kuna antud funktsioon oli inglise keelele kohandatud, ei saanud päris kindel olla, et ta eestikeelseid lauseid suudab eristada. Selles veendumiseks prooviti erinevaid täpitähtedega lauseid, jutumärke, kuupäevi jne. Lause eraldamine oli edukas ja võis kindel olla, et funktsioon ei eksi. Näited kodulehel tundusid väga lihtsad ja tundus, et moodul hõlpsustab lausete eraldamist. Selgus tõsisasi, et antud lisamoodul toetab maksimum Python keskkonda 2.6. Üle selle versiooni puudus Pythonile tugi. Kuna tolle hetkene programm töötas keskkonnas 3.1, siis oli autor sunnitud oma programmi ümber kohandama 2.6 versioonile. Enamus tööd oli tehtud, aga teatud

käsklused erinesid ja tuli hakata arvestama Unicode transleerimisega. Nimelt peab Python 2.x keskkonnas väga paljudel juhtudel deklareerima, kas string on UTF-8 või ASCII kodeeringus. ASCII kodeeringus tundmatud sümbolid nagu õ, ö, ä, ü esitati baidi kujul ja neid kirjutati HTML'i baitidena. UTF-8 deklareerimine võttis kauem aega, sest tarkvara oli selleks hetkeks väga mahukas.

Järgmine oluline faktor oli see ,et programm luges terve sisendfaili endale mällu. Kuna autor katsetas enda rakendust suhteliselt väikeste sisendfailidega, siis ei tekkinud programmil probleeme ülesande täitmisega. Kui selgus, et programm peab olema suuteline töötlema 200000 lauset, oli selline meetod välistatud. Sellele probleemile leiti lahendus kiiresti. Nimelt oli vaja lugeda sisendfailist algul mingi kindel arv sümboleid, mis ei kasuta liiga palju mälu ja üks lause mahub selle sümbolite arvu piiridesse, sealt eraldada esimene lause, lugeda kokku mitu sümbolit on selles lauses, kustutada faili algusest nii palju sümboleid, kui pikk see lause on. Eraldatud lause saadetakse veebipäringusse ja töötlemisse, kuni lõpuks kirjutatakse ta HTML faili, kui ta rahuldab teatud tingimusi.

Samuti kujunes mäluprobleemiks olemasoleva faili pidev mällu lugemine. HTML fail loeti iga kord mällu, lisati uus tabeli rida ja kirjutati taas faili. Probleem lahenes „append" failikirjutamis-režiimi kasutamisega.

Kuna morfsüntaktiline analüsaator tagastab vahel ka vasteid kujul „<Vigased>" ,millel morfoloogilised vasted ja süntaktilised rollid puuduvad, siis programm ebaõnnestus. Ajutine lahendus oli „try" ja „catch" meetodi kasutamine, mis tõi välja probleemsed kohad ja uus lahendus eemaldas toorikust „vigased" vasted. Samuti oli vaja lisada koodile osa, mis kontrollib, et kui ühte vastet ei leita, siis samale indeksile ei tekitata teise listi elementi. Näide: Kui puudub morfoloogiline vaste, siis ei lisata süntaktiliste rollide listi elementi. Viimane juhtum ei tohiks tegelikult võimalik olla ja indeksite nihkumist pole siiamaani tekkinud. Aga erandite vältimiseks oli see vajalik.

2.4 Testimine

Testimise jaotus kaheks: Autori enda testimine ja tulemuste näitamine juhendajale. Arenduse käigus ilmnes möödarääkimisi kahe osapoole vahel ja programmi pidi ümber kohandama. Üldiselt oli autor õiges suunas ja suuremaid muudatusi ei pidanud tegema. Kokkuvõtvalt võib

öelda, et testimine toimus samm haaval, üritati iga järgneva sammuga saada alati õiget tulemust. Esimesed testimised algasid sellega, kui autor õppis Python 3.x keskkonna süntaksit tundma ja tuletas meelde, mis Pythoni keele omapärad on. Pikaajaline kogemus oli olemas, aga autor oli hiljuti teiste keeltega tegelenud.
Regulaaravaldiste põhjalik tundma õppimine võttis aega ja palju proovimist. Kuigi siin juhul oli teksti eraldamine lihtsam, sest süntaks, kust andmeid eraldati, oli rangelt ühesugune, samas leidus erandeid. Lõpptulemusena sai valmis enamusjuhtusid aksepteeriv avaldis, samas ei ole välistatud, et ta kõiki olukordi katab.
UTF-8 kodeerimine kujunes tülikaks, raske oli aru saada mis kodeeringus sisse loetakse ja mis kodeeringus faili kirjutatakse. Python 2.x puhul peab kas sisselugemisel või faili kirjutamisel kodeerimis-režiimi ära määrama. Samuti ei saa listi UTF-8 kodeeringuga otse määrata, iga listi element vajab eraldi määramist.

Saabus aeg esimene ilukirjanduslik tekst loodud programmiga töödelda. Programm kirjutas 2 MB suurusest txt failist ~30 MB HTML tabeli faili, programm töötas üle 10 tunni. Nii otsis rakendus osalauses vähemalt kahte esinevat predikaati. Juhendaja tegi klastrianalüüsi, kui avastas et HTML fail on mitte aksepteeritav. Andmed olid tabelis valesti esitatud. Nimelt pidid olema ühe sõna morfoloogilised andmed ühes tulbas (ehk kui on morfoanalüüsist tuli mitu vastet ühele sõnale, siis tuli need paigutada ühte lahtrisse), süntaktilised andmed samuti ühte tulpa. Esialgne programm oli asetanud morfoloogilised või süntaktilised analüüsi tulemused kõik eraldi lahtritesse, kui esines mitu morfoloogilist, süntaktilist vastet. Viga esines, sest aset oli leidnud osapoolte mööda rääkimine. Samuti avastas juhendaja, et puudus üks predikaati eristav märgend ja fail oleks pidanud tegelikult veel suurem olema. Kusjuures selgus, et selle märgendi pidi hoopis morfoloogilisest vastest eristama, tegemist oli _A_ märgendiga ehk omadussõna. Kõik ülejäänud rollid olid siiamaani süntaktilistest vastetest saadud. Osalause tasandil kuulub ka omadussõna predikaatide hulka ning filoloogilises aspektis süntaktiline predikaat ei sisalda omadussõna. Esialgse klastrianalüüsi võis ära kustutada. Ja jäädi programmi uuendust ning uut tulemust ootama.
Indiviidide tabeli puhul kirjutati kogemata terve lause faili, kui ühes osalauses esines 2 indiviidi järjest. Programm vajas muutust, õnneks ei olnud tegemist väga mastaapse muudatusega ja programm kirjutas peagi juba õiget väljundit.

2.5 Tulemused

Autor otsustas rakendusele ka graafilise kasutajaliidese luua ja selleks oli olemas pythonil lisamoodul nimega wxPython. Tänu sellele sai loodud lihtne ja praktiline kasutajaliides. (Joonis 4) Iga kord kui programm tööd alustas kirjutati esimeses seisundis HTML faili algusesse *DOCTYPE HTML 4.01* standard. Seejärel lisati lehele CSS vormistus, mis määras elemendile *<td>* *font-family*'i väärtusega „*Verdana, Arial, Helvetica, sans-serif; font-size: 8pt*". Seejärel alustati tabelit ennast *<table>* märgendiga. Peale seda liikus programm järgmisse tsüklilisse seisundisse (Joonis 1). Kui uuritavas osalauses esinesid otsitavad rollid, siis lisati HTML faili üks rida *<tr></tr>* märgendite vahele ning ritta nii palju veerge, kui lauses sõnu oli *<td></td>* märgendite vahele. Peale ridade ja veergude lisamist liikus programm järgmisse seisundisse, mis sulges kogu HTML faili märgenditega *</table>*, *</html>*.

Osalausete analüüsi tulemused

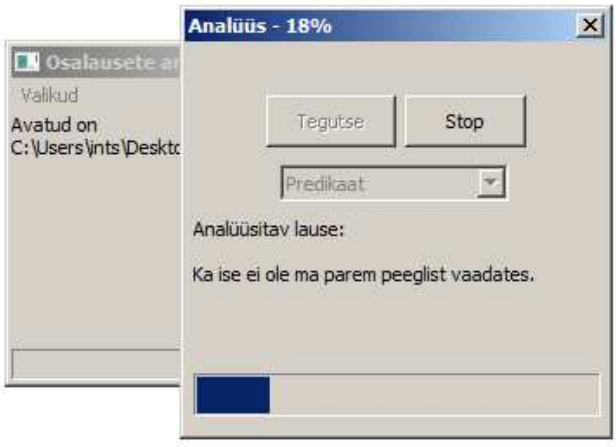
osalause	Ta ropendas lõõpis punkt	Ta	"tema" L0 P pers ps3 sg nom cap	@SUBJ	Indiviid	ropendas	"ropenda" Ls V main indic impf ps3 sg ps af cap	@FMV	**Predikaat**	lõõpis	"lõõpi" Ls V main indic impf ps3 sg ps af cap	@FMV	**Predikaat**	punkt	"punkt" Z	Fst	Kirjavahemark

Joonis 2 Predikaatide HTML tulemusfail

Osalausete analüüsi tulemused

osalause	Tunnen silmapilklist alanemist oma tundmustes	Tunnen	"tund" Ln V main indic pres ps1 sg ps af cap "tund" Ln V mod indic pres ps1 sg ps af cap	@FMV @FCV	Predikaat Predikaat	silmapilklist	"silmapilkl" Lst S com sg el cap "silmapilkli" Lst S com sg el cap	@NN> @ADVL @NN> @ADVL	**Indiviid** Indiviid
osalause	, kes tütarlastekoolis õpetust andis , Anna meelest Q inetu	,	"," Z	Com CLB	Roll puudub	kes	"kes" L0 P inter rel sg nom cap	@SUBJ	**Indiviid**
osalause	ja kord kui vend teda ühe Q	ja	"ja" L0 J crd cap	CLBC	Konjunktsioon	kord	"kord" L0 S com sg nom cap	@SUBJ @ADVL	Indiviid
osalause	Ma olen tüdinenud , oma elukorrast ja elust maal väikestes linnades	Ma	"mina" L0 P pers ps1 sg nom cap	@SUBJ	Indiviid	olen	"ole" Ln V aux indic pres ps1 sg ps af cap	@FCV	Oigsus Predikaat
osalause	Pakub end võimalus väljamaale reisida	Pakub	"pakku" Lb V main indic pres ps3 sg ps af cap	@FMV	Predikaat	end	"ise" L0 P pos det refl sg part cap	@OBJ	**Indiviid**
osalause	, mis provintsi üksinduses tekib , väikelinnade kaugetes kolgastes	,	"," Z	Com CLB	Roll puudub	mis	"mis" L0 P inter rel sg nom cap	@NN> @SUBJ	**Indiviid**

Joonis 3 Indiviidide HTML tulemusfail



Joonis 4 Kaks dialoogiakent (graafiline kasutajaliides)

2.6 Klastrianalüüs

Kui valminud programm oli otsinud sisendfailist predikaate ja indiviide, oli vaja tulemusfailid veel Microsoft Exceliga töödelda. Esialgses sisendfailis (txt) oli 28148 lauset.

2.6.1 Predikaatide analüüs

Peale sisendfailist mitme predikaadiga osalausete otsimist, sisaldas HTML fail umbes 20000 osalauset. Sellest HTML failist hakati välja eraldama omakorda huvipakkuvaid osalauseid ehk algas puhastamine. Selleks kasutati kahte makrot. Esimene makro otsis välja ainult ühese või üheste süntaktiliste märgenditega osalaused ja kopeeris need eraldi faili. Puhastamise tagajärjel jäi neid alles 14749. Puhastamisel asendati mitu ühesugust märgendit ühega. Miks sellised juhused esinesid tulenes sellest, et vahest oli ühele sõnale mitu vastet ja neile oli ühesugune süntaktiline roll. Kui sõnale vastas mitu erinevat süntaktilist märgendit, siis need jäeti hetkel välja, see on morfsüntaktilise analüsaatori probleem ja seda praegune uurimustöö ei kajasta. Nüüd rakendati teist makrot, mis luges kokku erinevate mustrite esinemissagedused. Autor toob välja neli kõige sagedamini esinenud mustrit ühes osalauses asetsevate predikaatide kohta:

1) Kõige sagedasem muster, mida esines 5001 korda, oli @FMV ja _A_. @FMV tähistas finiitset verbi ja _A_ tähistas omadussõna. Näited sellise mustriga osalausetest:

* „*Ta on surnud*” – „*on*” märgistas @FMV ja „*surnud*” märgistas _A_.
* „*kaotavad hea nime*” – „*kaotavad*” märgistas @FMV ja „*hea*” märgistas _A_.
* „*Tal oli hale meel*” – „*oli*” märgistas @FMV ja „*hale*” märgistas _A_.
* „*Kas ta oli haige*” – „*oli*” märgistas @FMV ja „*haige*” märgistas _A_.

2) Teine muster, esines 1861 korda, oli @FCV ja @IMV. @FCV tähistas finiitset vormi ja @IMV tähistas infiniitset verbi. Näited:

* „*Me oleme sellest vahel rääkinud*” – „*oleme*” märgistas @FCV ja „*rääkinud*” märgistas @IMV.
* „*Kas te arstile olete helistanud*” – „*olete*” märgistas @FCV ja „*helistanud*” märgistas @IMV.
* „*Ta on lõpuks murdunud*” – „*on*” märgistas @FCV ja „*murdunud*” märgistas @IMV.
* „*Põhjus on seega kõrvaldatud*” – „*on*” märgistas @FCV ja „*kõrvaldatud*” märgistas @IMV.

3) Kolmas muster, esines 1421 korda, oli _A_ ja @FMV, ehk esimene muster tagurpidises järjekorras. Näited:

* „*kuid peen pori laotub igale poole*” – „*peen*” märgistas _A_ ja „*laotub*” märgistas @FMV.
* „*Kus kogu maailm kargleb*” – „*kogu*” märgistas _A_ ja „*kargleb*” märgistas @FMV.

* „*et ma pahaks ei pane*” – „*pahaks*” märgistas _A_ ja „*pane*” märgistas @FMV.

* „*Mis ma seal õige teen*” – „*õige*” märgistas _A_ ja „*teen*” märgistas @FMV.

4) Neljas muster, esines 1047 korda, oli @FMV, _A_ ja _A_. Ehk üks finiitne verb ja kaks omadussõna. Näited:

* „*Algab lame tsiviliseeritud vestlus*” – „*Algab*” märgistas @FMV, „*lame*” märgistas _A_, „*tsiviliseeritud*” märgistas _A_.

* „*Kohvreid antakse edasi rõõmsate ja sõbralike hõigetega*” – „*antakse*” märgistas @FMV, „*rõõmsate*” märgistas _A_ ja „*sõbralike*” märgistas _A_.

* „*Televiisor näitab musta mehe keevalist armastust tühjusele*” – „*näitab*” märgistas @FMV, „*musta*” märgistas _A_ ja „*keevalist*” märgistas _A_.

* „*on väike ilus aas*” – „*on*” märgistas @FMV, „*väike*” märgistas _A_ ja „*ilus*” märgistas _A_.

Need esimesed neli mustrit esinesid kõik vähemalt üle tuhande korra ja viies muster esines juba 656 korda, mis on üpriski suur vahe neljandast. Kõige sagedasem muster esines 5001 korda ja ületas teist kohta 3140 võrra. Siit on näha, et eesti keeles on lause ehituses üks lauseehituse muster väga tugevalt kasutusel. Samas erinevaid mustreid kokku oli 307. Nendest 148 mustrit esinesid ainult ühe korra. 2 korda esines 34 erinevat mustrit. Üle 10 korra esines 58 mustrit. Neljandast mustrist võib näha, et paljudel juhtudel on tegemist pigem ühe objekti omaduste järjestamisega. Esimestest neljast sagedamast mustrist omavad kolm omadussõna mustris, kui see oleks välja jäetud oleksime teistsuguse tabeli saanud.

2.6.2 Indiviidide analüüs

Samuti oli vaja ka indiviidide tabel Microsoft Excelis töödelda. Siin juhul oli vaja välja tuua vähemalt 2 kõrvuti asetsevat indiviidi. Selle põhjal saab Avar Pentel oma doktoritöös uurida indiviidide ühendamist üheks. Ehk kui mitu indiviidi viitavad ühele samale objektile, siis kuidas neid tuvastada ja millal neid tohib üheks liita. Esialgses HTML tabelis, millesse oli sisendfailist (28148 lauset) eraldatud osalauseid, sisaldas 10604 osalauset. Samuti tehti ka siin puhastust, nimelt pidi taas Makro 2 eraldama HTML failist ainult need osalaused, mis sisaldasid üheseid süntaktilisi rolle, lõpuks oli neid kokku 4394. Sagedasem muster oli @NN> (nimi-, ase- ja põhiarvsõna eestäiendina) ja @SUBJ (alus ehk subjekt), see esines 568 korda ja autori esimesel hinnangul olid neist ühendatavad 273 indiviidi, ehk ~48%. Umbes pooled neist saab üheks indiviidiks liita, aga see protsent võib muutuda ja ei pruugi täpne olla.

Siit saab järeldada, et kui osalausetes on eesti keeles muster @NN> ja @SUBJ, siis umbes ½ juhtumil on tegu ühele samale objektile viitavate indiviididega.

Analüüsi kokkuvõte

Sisendlauseid oli 28148, kuid esialgu oli plaanitud 200 000. Kahjuks tekkis ajapuudus ja ei olnud kohe nii suurt sisendfaili saadaval, kui tegelikult vaja oleks olnud. Loodud tarkvaraga on plaanis rohkem lauseid läbi töödelda ja autor kavatseb abistada koodi muutmisel ja tõlgendamisel, kui peaks mingisuguseid probleeme tekkima. Rohkemate mustrite leidmiseks ja sageduste kinnistamiseks on plaanis 200000 lauset läbi töödelda. Aga juba nii väikeses koguses oli predikaatide puhul näha, kuidas üks muster hakkas teistest eristuma ja ületas teisi 3140 võrra, kui lauseid on rohkem, läheb see vahe veel suuremaks. Aga kuna tegemist oli ilukirjandusliku tekstiga, siis esindavad antud andmed ilukirjanduse korpuse baasil saadud andmeid. Mustrid võivad erineda, kui tegemist oleks näiteks ajakirjanduse korpusega.

Indiviidide puhul selgus, et umbes pooled järjestikused indiviidid mustris @NN> ja @SUBJ saab üheks indiviidiks liita, samas ei ole see jaotuvus lõplik.

Summary: Problematic Clauses in Presenting Texts by Logical Formulas

The initial data that was processed consisted of 28148 sentences. Fictional texts were studied and fictional text corpus was used. The application sliced all the sentences into clauses and the ones that were to be written into an HTML file, were decided by the user input. In this research predicates and individuals were studied, therefore two HTML files were generated. One for the predicates and the other one for the individuals. Two HTML files were also sorted in Microsoft Excel by the help of two macros. First macro filtered out all the clauses that consisted of only unique syntactical roles. If there were many of one kind, they were replaced with one. If one word consisted different syntactical roles, it was ignored, because that is the error of the morphsyntactical analyzer, which was not handled in this paper. After that the second macro had to count all the different patterns in a clause structure and how many occurences were there to every pattern. Predicates had 307 different patterns and individuals had 568 patterns. Finally the patterns were sorted incrementally by the count number. With these patterns we can study why dialog system DST fails in some cases at the clause level. We are also able to study in what cases individuals can be alligated. Also a lot more sentences are planned to be processed by the software, so it would give us a better look at the patterns and their occurences in fictional texts.

Kasutatud kirjandus

1. Kadri Muischnek, Korpuslingvistika. URL
http://www.cl.ut.ee/kursused/korpuslingvistika (viimati vaadatud 14.04.2011)

2. Müürisep Kaili. „*Developing a Syntactic Analyser for Estonian*". 1999

3. Müürisep Kaili, ESTKG süntaktilised märgendid. URL
http://www.cs.ut.ee/~kaili/parser/demo/synttags.html (viimati vaadatud 14.04.2011)

4. Müürisep Kaili, Morfoloogilised märgendid. URL
http://www.cs.ut.ee/~kaili/parser/demo/morftags.html (viimati vaadatud 14.04.2011)

5. Lorents Peeter, „*Keel ja loogika*". 2000

6. Matsak Erika, Loomuliku keele tekstides sisalduvate loogiliste konstruktsioonide väljaeraldamise dialoogsüsteem, 2005

7. Matsak Erika, Eestikeelsetes tekstides sisalduvate loogiliste konstruktsioonide väljaeraldamise süsteem, 2004

8. Matsak Erika, Discovering Logical Constructs from Estonian Children Language. URL
http://digi.lib.ttu.ee/i/?465 (viimati vaadati 14.04.2011)

9. Regular expressions operations. URL http://docs.python.org/library/re.html (viimati vaadatud 14.04.2011)

10. Python - Regular Expressions. URL
http://www.tutorialspoint.com/python/python_reg_expressions.htm (viimati vaadatud 14.04.2011)

11. Morfoloogilise analüsaatori ESTMORF kasutamine. URL
http://www.filosoft.ee/html_morf_et/morfoutinfo.html (viimati vaadatud 14.04.2011)