TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Karl Erik Karindi

Lõputöö teksti analüsaator

Bakalaureusetöö (9 EAP)

Juhendaja: Raimond-Hendrik Tunnel, MSc

Tartu 2020

Lõputöö teksti analüsaator

Lühikokkuvõte:

Käesolevas lõputöös loodi programm, mis keeletehnoloogilisi lahendusi kasutades annab lõputöö kirjutajale automaatset tagasisidet lõputöö kohta. Programmi eesmärk oli katsetada EstNLTK funktsionaalsusi ja oma loodud analüsaatorite reegleid. Selle kasutamiseks loodi API ja kasutajaliides. Lõpuks paluti mitme iteratsiooni jooksul tudengitel testida programmi ja anda sellele tagasisidet. Tulemuste analüüsis selgus, et kasutajad parandasid oma tekstides analüsaatori välja toodud vead ja kokkuvõtlikult hinnati analüsaatorit positiivselt.

Võtmesõnad:

Keeletehnoloogia, loomuliku keele analüüs, API, keelereeglite katsetused, kasutajatestimine, kantseliit, pikad laused, komavead, umbisikulisus

CERCS:

P170, Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

P175: Informaatika, süsteemiteooria

Thesis Text Analyzer

Abstract:

In this thesis, a program that uses language technology and natural language processing to give automatic feedback about a thesis text was created. The purpose of this program was to experiment with EstNLTK functions and various self-made rules. To use the program, an API and web-service were created. Finally, the program was tested by students over multiple iterations. Analyzing the results, it seemed that the users fixed the mistakes pointed out by the analyzer. It was also found that the analyzer was generally rated positively.

Keywords:

Language Technology, natural language processing, API, language rule experiments, user testing, officialese, long sentences, comma mistakes, impersonality

CERCS:

P170, Computer science, numerical analysis, systems, Control

P175: Informaatika, süsteemiteooria

Sisukord

1. Sissejuhatus

Igal aastal lõpetab Tartu Ülikooli üle 2000 tudengi, kellest igaüks peab lõpetamiseks kirjutama lõputöö[1]. Lõputöö on demonstratsioon teadmistest ja oskustest, mida tudeng on oma õpingute vältel omandanud. Tartu Ülikooli arvutiteaduse instituudis on võimalik valida kas rakendusliku või teoreetilise töö vahel [1]. Kuigi eesmärgid ja metoodika võivad kahe liigi vahel erineda, on neil mõlemal siiski kirjalikul osal oluline rõhk. Lõputööd kirjutades on oluline kasutada korrektset ja teadustekstile omast keelekasutust ning -stiili (vt peatükk 2).

Tudengil on teadusteksti norme ja tavasid tihti raske järgida, kuna selle kirjutamisega on tavaliselt vähe kogemusi. Lõputöö kirjutamise protsessi käigus tekib kirjutades nii stiili- kui ka grammatikavigu, mille parandamisele kulub palju aega ja energiat. Veelgi probleemsem on olukord, kus vead jäävad märkamata ja lõputöö kvaliteet seepärast kannatab.

Korrektne kirjastiil toetab tööd ja aitab lugejal keskenduda teksti sisule. Seaduspärasustest hoidumine raskendab info vastuvõtmist, kuna lugeja tähelepanu koondub hoopis keelekasutusele [2].

Käesolevas lõputöös katsetati automaatsete analüüside loomiseks erinevaid EstNLTK algoritme ja funktsionaalsusi. Kasutati ja arendati ka olemasolevaid lahendusi, mis olid leitud kas internetist või varasematest lõputöödest. Erinevate algoritmide ning analüüside koos katsetamiseks loodi analüsaatorist programm, mis toetab tudengit ja juhendajat eestikeelse lõputöö kirjutamisel.

Programmi arendamisel katsetati reeglitega, mis aitaks järgida head lõputööle omast keelekasutust, -stiili ja -nõudeid. Programm kasutab keeletehnoloogilisi lahendusi ja loomuliku keele töötlust, et tuvastada problemaatilised tekstilõigud, laused või sõnad. Programmi on loodud umbisikulisuse, lausepikkuse ja -keerulisuse, korduvate sõnade, puuduvate komade ja kantseliidi analüsaatorid (vt peatükk 5). Mõnedel juhtudel, näiteks puuduvate komade analüüsis, pakutakse vigade parandamiseks soovitusi.

Programmist peaks kasu saama tudengid ja juhendajad nii töö kirjutamise käigus kui ka lõpliku töö ülevaatusel.

[1] Kõrghariduse esimese astme ja integreeritud õppe lõpetanud (alates 1985) Tartu Ülikoolis http://www.ut.ee/lopetanute-nimekiri/

Töö idee tuli veebipõhise õpikeskkonna CGLearn lõputööde moodulist[2]. CGLearn on Raimond-Hendrik Tunneli loodud veebirakendus, mille põhiline eesmärk on ainekursuste materjalide jagamine ja haldamine.

Lõputööde moodul võimaldab lõputöö kirjutajaid efektiivsemalt juhendada [3]. Näiteks saab moodulis nii juhendaja kui ka juhendatav lihtsasti jälgida lõputöö kirjutamise protsessi. Juhendatava jaoks luuakse automaatselt graafik, kus on toodud välja ülesanded ja olulisemad tähtajad. Juhendatav saab ka logida lõputööle kulutatud aega ja jälgida graafikust kinni pidamist. CGLearn õpikeskkonda saavad kasutada vaid arvutigraafika ja virtuaalreaalsuse labori[3] juhendajad ja nende juhendatavad.

CGLearni registreeritud tudengid saavad iga nädala alguses raporti oma lõputöö kirjaliku osa mustandist. Õpikeskkonna tudengid kirjutavad mustandit Google Docs[4] failina. Raportis on välja toodud statistika ja tagasiside lõputöö kohta. Näiteks on raportis automaatne tagasiside selle kohta, kas tudeng püsib ajagraafikus ning kas mustandis on piisavalt pilte ja sõnu. Käesoleva lõputöö tulemusena lisandus raportisse ka eestikeelsete lõputööde tekstianalüüs.

Enne lõputöö analüsaatori loomist analüüsis CGLearn tekstianalüüsi poolelt ainult ingliskeelsete lõputööde tekstikeerukust. Tekstikeerukuse arvutamiseks kasutati Flesch-Kincaid lugemiskeerukuse algoritmi [4]. CGLearni autor leidis see-eest, et tudengitel ei olnud tekstikeerukuse analüüsist kasu. Seda seetõttu, et meetrik on oluline pigem lasteraamatute puhul, kus tekst peab lastele kergesti arusaadav olema.

Sooviti, et iganädalastest raportitest oleks juhendatavale võimalikult palju kasu. Seetõttu loodi käesoleva lõputöö raames erinevaid analüsaatoreid, mis annavad tagasisidet levinumate tekstivigade kohta.

Iga nädala pühapäeva öösel esitab CGLearn õpikeskkond lõputööde analüsaatori programmi API-le päringu, mille vastuses on tekstianalüüside tulemused. API on rakendusliides, millega eraldiseisvad programmid saavad üksteisega infot vahetada[5]. Analüüside tulemused

[2] CGLearn koduleht https://cglearn.eu/

[3] Arvutigraafika ja virtuaalreaalsuse labori koduleht https://cgvr.cs.ut.ee/
[4] Google Docs https://www.google.com/docs/about/
[5] API https://www.freecodecamp.org/news/what-is-an-api-in-english-please-b880a3214a82/

lisatakse raportisse, kust juhendatavad saavad näha automaatset tagasisidet oma lõputööde tekstidele (vt Lisa IV).

Selleks, et lõputööde analüsaator oleks lihtsasti kättesaadav ka kirjutajatele, kes pole CGLearn õpikeskkonda registreeritud, loodi programmi jaoks veebiliides. Veebiliidesest on rohkem kirjutatud peatükis 4.2.

Kuigi rakendus on veebiliidese kaudu kasutatav kõikidele lõputöö kirjutajatele, on analüsaatorid teataval määral subjektiivsed ja ei pruugi vastata kõikide erialade ja instituutide nõuetele. Näiteks toob rakendus välja mina- või meie-vormis sõnad, ent mõnes instituudis võib nende kasutamine lõputööd kirjutades olla täiesti aktsepteeritav. Samamoodi on lausepikkuse ja -keerulisuse kontrolliga. Rakendus soovitab kasutada lihtsaid ja konkreetseid lauseid, ent erinevate juhendajate ja instituutide vahel võib soovitud ja eeldatav lause pikkus erineda. Rakendust saavad kasutada kõik tudengid, aga see on põhiliselt konfigureeritud CGLearn tudengite jaoks.

Esialgu sooviti teha programm, mis suudaks analüüsida nii ingliskeelset kui ka eestikeelset teksti, ent lõpuks otsustati keskenduda ainult eestikeelsetele tekstidele. Seda seetõttu, et bakalaureusetöö maht on liiga väike, et keskenduda kahe keele analüüside loomisele. See-eest on käesoleva töö pealt lihtne teha edasiarendusi teiste keelte analüüside jaoks (vt peatükk 7.3) Lisaks sooviti katsetada EstNLTK funktsionaalsusi ja võimalusi automaatseks tekstianalüüsiks.

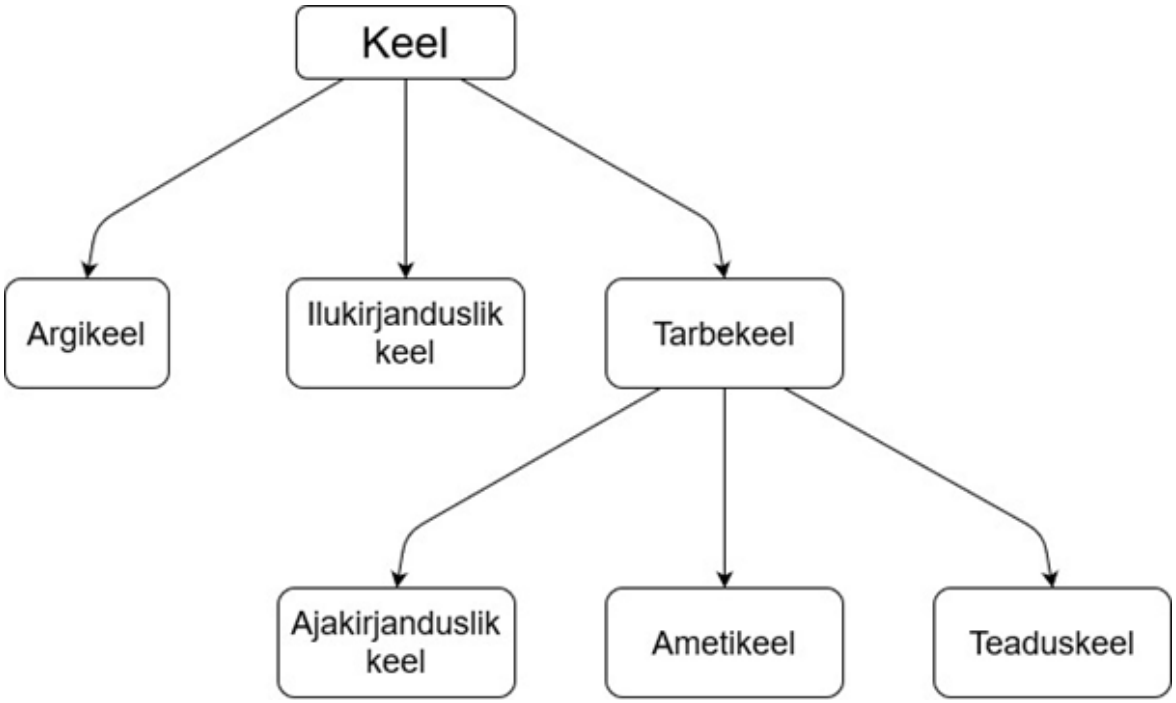
Lõputöö viimases faasis uuriti, kas loodud analüüsid olid eestikeelsete lõputööde kirjutamisel tudengite jaoks kasulikud (vt peatükk 6). Kasutajatel paluti mitme nädala jooksul anda tagasisidet programmi erinevatele versioonidele. Erinevate versioonide vahel tehti programmis edasiarendusi. Seejärel võrreldi, kas analüüsid olid ajapikku kasutajate jaoks kasulikumaks läinud.

Lõputöö teksti analüsaatori programm on kättesaadav lingil
https://thesisanalyzer.cloud.ut.ee

Programmi lähtekood on kättesaadav lingil https://gitlab.com/karlkarindi/thesisanalyzer

2. Teadustekst

Keelt saab kasutuse järgi jaotada argikeeleks, ilukirjanduslikuks keeleks ja tarbekeeleks, millest viimast on võimalik omakorda jaotada ajakirjanduslikuks keeleks, ametikeeleks ja teaduskeeleks (vt Joonis 1) [2]. Selles lõputöös keskendutakse just teaduskeele analüüsile.



Joonis 1. Keele kasutamise jaotus

Teadusstiili eristuvad tunnused on korrektne keelekasutus, vormistuse ja kompositsiooni nõuded, spetsiifilised ja üheselt mõistetavad sõnad, impersonaalsus, neutraalne sõnavara ning oskussõnavara ja terminite kasutamine [2]. Üks levinumaid ekslikke arusaamasid on see, et teadustekst peab olema keeruline, ent tegelikkuses peaks see olema selgesti arusaadav ja sujuvalt loetav ilma, et see muutuks liiga primitiivseks [2].

On oluline, et kõik välja toodud faktid ning väited oleksid põhjendatud kas näidete või allikaviidetega ja kõik mõisted ja terminid selgitatakse esimesel kasutusel lahti. Välditakse liigset sõnakasutust ja on pööratud tähelepanu teksti liigendusele ja struktuurile [2].

Neutraalsuse poole pealt peaks tulemused olema esitatud erapooletult. Allikad peab valima ja tõlgendama objektiivselt ja tuleb vältida argikeelt, kujundlikke väljendeid ning ülemääraseid omadus- ja määrsõnu [2].

Arvutiteaduse instituudis soovitatakse jätta autori isik ja tema isiklikud arvamused kõrvale ehk tekstile soovitatakse läheneda impersonaalselt [1]. Erinevates valdkondades on see-eest

erinevad reeglid. Näiteks on kohati lubatud ja isegi soovituslik kasutada mina-vormi või ümberütlevaid väljendeid [2]. Näide ümberütlevast väljendist oleks see, kui *Arvan, et* asemel kasutada hoopis *Autor on seisukohal*.

Juhendajatel on impersonaalsusega seoses tihti isiklikud eelistused. Näiteks CGLearni kasutava arvutigraafika lõputööde juhendaja soovitab kasutada personaalset lähenemist vaid kokkuvõttes, kus reflekteeritakse tehtud tööle.

Teaduskeele järgimiseks on tehtud mitmeid teeke ja programme, mis lihtsustavad reeglitest kinnipidamist. Järgnevas peatükis võetakse vaatluse alla kaks olemasolevat programmi, et näha, millistest olemasolevatest rakendustest saaks tekstianalüüsi juures eeskuju võtta.

3. Sarnased programmid

Eesti- ning ingliskeelsete tekstide puhul pole olemas levinud programmi, mille põhieesmärgiks on lõputööde ehk akadeemilise teksti automaatne analüüs ja tagasisidestamine. Küll on see-eest olemas programmid, mille eesmärk on aidata autorit kirjutamisega, pakkudes välja grammatika- või stiiliparandusi. Mõned neist on järgnevalt välja toodud.

3.1 Microsoft Word

Microsoft Word[6] on üks enim kasutatud tekstitöötluse programme, mis pakub võimalusi teksti automaatseks parandamiseks [5]. Eesti keele jaoks pakub Word ainult õigekirja kontrolli, ent teiste keelte jaoks on lisaks õigekirjale võimalik kontrollida grammatikat ning korrektset kirjastiili [5]. Kuna Word ei paku grammatika ja kirjastiili kontrolli eesti keelele, vaadeldakse järgmistes näidetes programmi analüüse ning soovitusi ingliskeelsete näidete puhul.

Sarnaselt lõputöö teksti analüsaatoriga, pakub Word puuduvate komade analüüsi. Näiteks lause *The man walked with his dog but the weather was terrible* puhul leiab Word, et sõna *but* ees peab olema koma. Analoogselt tuvastab lõputöö teksti analüsaator, et lauses *Mees kõndis oma koeraga aga ilm oli kohutav* on sõna *aga* ees koma puudu. Kuna eesti keele jaoks grammatika kontroll puudub, siis Word ei tuvasta eestikeelses lauses komaviga.

Word suudab tuvastada tekstiosasid, milles on kasutatud halba kirjastiili. Näiteks tuvastatab Word lauses *This program was made by students*, et seda saab paremini kirjutada kui *Students made this program*.

Word pakub ingliskeelse teksti jaoks umbisikulisuse kontrolli. Näiteks lause *I made this* puhul tuvastatakse, et lauses on kasutatud mina-vormi.

Ingliskeelse teksti jaoks pakub Word lausepikkuse ja -keerulisuse kontrolli. Word ütleb, et lause *For all intents and purposes, the reason Mr. Henderson arrived late for work was due to the fact that he stopped at very many traffic lights that were red in colour* on liiga pikk[7]. Selle asemel pakub Word hoopis lauset *The reason Mr. Henderson arrived late for work was because he stopped at very many traffic lights that were red in color*. Word pakub pika

[6] Microsoft Word koduleht https://www.microsoft.com/en/microsoft-365/word
[7] Common Grammatical Errors: Wordiness https://bethune.yorku.ca/writing/wordiness/

lause parandamiseks lahendust, et muudetakse lause sõnastust. See-eest võiks hea analüsaator pakkuda ka seda, et teha ühest pikast lausest mitu lühemat.

Microsoft Word annab kasulikke soovitusi lausete ja stiilivigade parandamiseks, ent selle suurim miinus on see, et eesti keele jaoks stiilivigade kontrolli pole. Rakenduse suurim pluss on selle õigekirjakontroll, mis suudab edukalt leida ja esile tuua kirjavigu.

Kuna Word tuleb õigekirja parandamisega hästi toime, ent stiiliparandusi see eesti keeles ei paku, otsustati käesolevas lõputöös luua analüsaatorid, mis tuvastaksid ka levinumaid stiilivigu. Näiteks suudab lõputöö teksti analüsaator tuvastada stiilivigade hulgast korduvaid sõnu, pikki lauseid, halba kirjastiili ja mina- või meie-vormi (vt peatükk 5).

3.2 Grammarly

Grammarly[8] pakub veebirakendust, mis tuvastab kasutaja kirjutatud tekstis kirja- ning stiilivigu. Rakenduse põhiline eesmärk on aidata inimestel paremini ning efektiivsemalt üksteisega suhelda[9]. Grammarly suudab analüüsida inimeste sotsiaalmeedia postitusi, sõnumeid ning dokumente[9]. Kasutatakse tehisintellekti, et tuua välja grammatikavigu ja teha soovitusi kirjastiili parandamiseks[9]. Grammarly toetab ainult inglise keelt.

Järgnev info kaasa arvatud lausenäidetega on võetud Grammarly kodulehelt siis, kui kasutaja on sisse logitud[10].

Grammarly pakub nii tasuta kui tasulist versiooni oma rakendusest. Tasuta versioon parandab vaid õigekirja, grammatikat ja punktuatsiooni. Näiteks suudab tasuta versioon tuvastada puuduvaid komasid. Lauses *I like to travel in the winter not the summer* leiab Grammarly, et sõna *not* ees on puuduv koma.

Tasulisel versioonil on see-eest rohkem funktsionaalsusi. Näiteks tuvastab tasulises versioonis Grammarly ülekasutatud sõnad. Grammarly leiab, et lauses *An algorithm change can be an opportunity for your company to explore a change of strategy* on liiga palju kasutatud sõna *change*. Rakendus soovitab asendada teine *change* sünonüümiga *shift*.

[8] Grammarly koduleht https://app.grammarly.com/
[9] Grammarly tutvustus https://support.grammarly.com/hc/en-us/articles/115000090792-What-is-Grammarly-
[10] Grammarly koduleht sisselogituna. https://www.grammarly.com/plan?breadcrumbs=true&firefoxDataControl=true&install=true&page=install&profile=true

Lisaks suudab tasuline versioon tuvastada stiilivigu. Näiteks lause *The late fees should be paid by Jesse* saab Grammarly sõnul kirjutada paremini kui *Jesse should pay the late fees*.

Kuigi tundub, et Grammarly tekstianalüüs töötab hästi ja on väga arenenud, ei suuda see analüüsida eestikeelset teksti. Nagu Microsoft Word puhulgi, on see üks rakenduse suurimaid miinuseid.

Grammarly ei paku lahtist API teenust[11]. See tähendab, et iseseisvad arendajad ei saa integreerida oma rakendusi Grammarly teenustega selleks, et kasutada oma rakendustes Grammarly tekstianalüüsi. Näiteks ei saa CGLearn pöörduda Grammarly API poole, et analüüsida ingliskeelsete lõputööde tekste. Teksti analüsaator võiks olla see-eest kättesaadav kõigile, kaasa arvatud erinevatele rakenduste arendajatele.

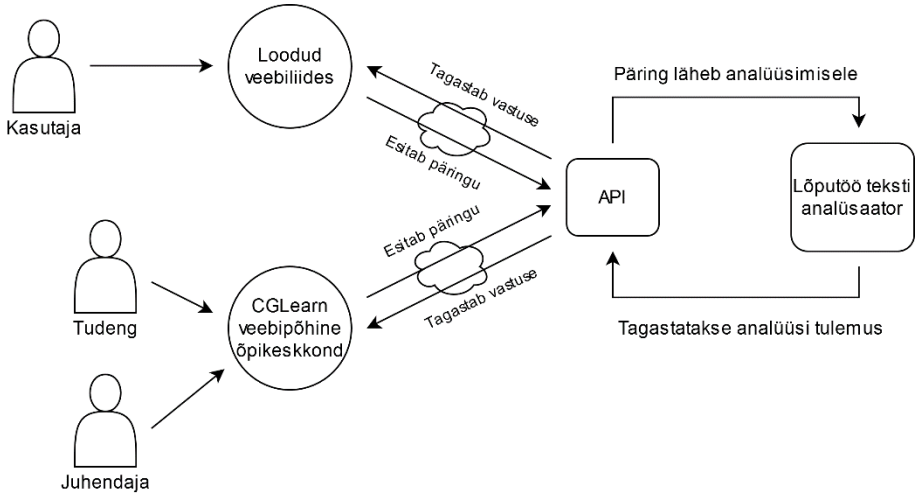
[11] Grammarly vastus seoses API küsimusega https://twitter.com/Grammarly/status/955810447910948864

4. Programmi ülevaade

Järgnevates alapeatükkides antakse ülevaade programmist. Selgitatakse programmi ülesehitust, suhtlust kasutajaliidesega ja kasutatud tehnoloogiaid. Kirjeldatakse jõudlust ja enim kasutatud teeke. Kuna programm on mõeldud erinevate reeglite ja EstNLTK funktsioonide katsetamiseks, on selle disain modulaarne ja kergesti muudetav edasiste arengute jaoks.

4.1 Ülesehitus

Lõputöö analüsaatori jaoks loodi lahtine *Application Programming Interface* (API) mikroteenus. Mikroteenusele esitab päringu veebipõhine õpikeskkond CGLearn või analüsaatori jaoks loodud veebiliides (vt Joonis 2). Mikroteenus on kirjutatud programmeerimiskeeles Python ja see kasutab Flask[12] veebiraamistikku (vt peatükk 4.5), mis on võrdlemisi kergekaaluline ja sellise projekti jaoks sobiv. Programmeerimiskeeleks valiti Python seetõttu, et selles on kättesaadav EstNLTK[13] teek (vt peatükk 4.6), millega kaasnevad sisseehitatud keeletöötluse ja -analüüsi funktsionaalsused. Programm jookseb Ubuntu[14] operatsioonisüsteemiga serveril.



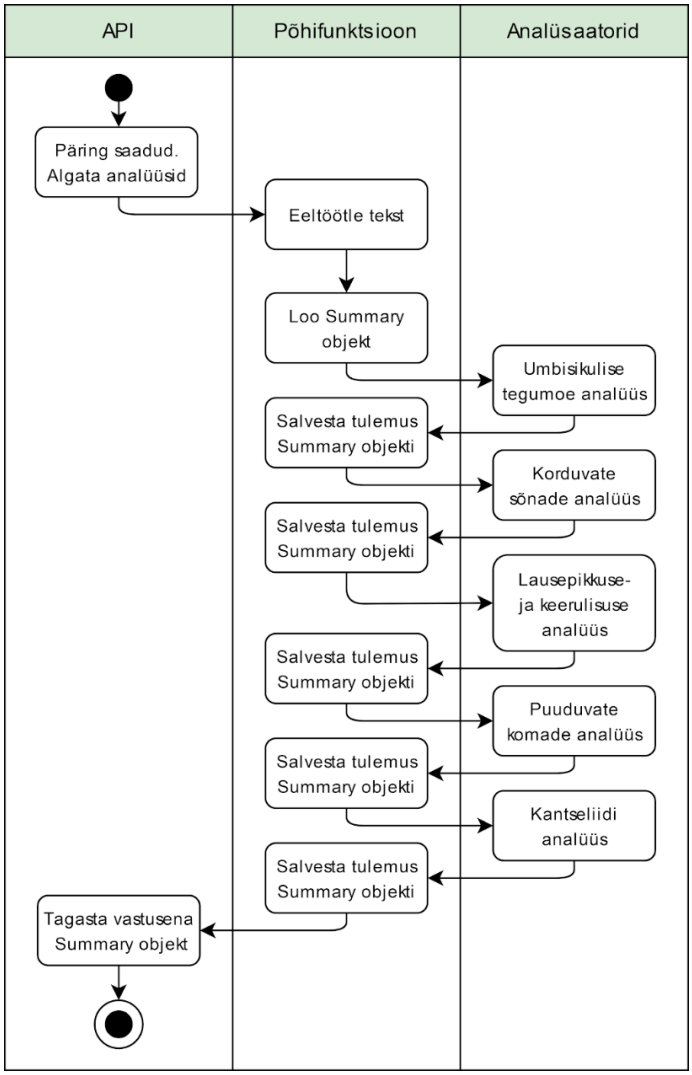
Joonis 2. Lõputöö analüsaatori ülesehitus

Lõputöö teksti analüsaator on jagatud erinevateks väiksemateks analüsaatoriteks, millest igaühel on oma eesmärk. Klient saadab API-le päringu, milles on JSON formaadis analüüsimisele minev sisendtekst. JSON dekodeeritakse, et saada puhtal sõnekujul tekst. Seejärel kutsutakse `analysis_main` moodulis välja funktsioon `analyze`.

[12] Flask koduleht. https://flask.palletsprojects.com/en/1.1.x/
[13] EstNLTK https://github.com/estnltk/estnltk
[14] Ubuntu koduleht. https://ubuntu.com/

Programmi ülesehitus on modulaarne ja paindlik. Analüsaatorid on üksteisest sõltumatud ja eraldi moodulites. Kuna iga analüsaator on eraldi moodulis, on neid võimalik lihtsasti muuta. Modulaarse ülesehituse üks suurimaid boonuseid on ka see, et programmi on soovi korral lihtne lisada uusi analüsaatoreid. Programmi töövoogu illustreerib Joonis 3.

Moodulis `analysis_main` luuakse esmaselt `Summary` objekt. Objektil on andmeväljad, kuhu hakatakse kõikide analüsaatorite tulemusi salvestama. Sisendtekst eeltöödeldakse ja algatatakse analüüsid. Analüsaatorite ja eeltöötluse kohta on rohkem juttu peatükis 5. Kui mingi spetsiifiline analüsaator on oma töö lõpetanud, salvestatakse selle tulemus eelmainitud `Summary` objektis vastavasse andmevälja.



Joonis 3. Lõputöö teksti analüsaatori töövoog

Kui kõik analüüsid on tehtud, tagastatakse kliendile API kaudu vastus, mis on JSON kujul `Summary` objekt. Kasutajatele näitab vastust arusaadaval kujul programmi jaoks loodud veebiliides või CGLearn õpikeskkond.

4.2 Veebiliides

Lõputöö analüsaatori jaoks loodi veebiliides (vt Joonis 4), mida saab kasutada koheseks tekstianalüüsiks. Testimise ajal salvestati kõik veebiliidese kaudu tehtud päringud andmebaasi. Sellest on rohkem kirjutatud peatükis 4.4.

Joonis 4. Lõputöö teksti analüsaatori veebiliides

Liides on mõeldud rakenduse lihtsasti kasutamiseks. Kasutaja peab vaid kopeerima oma lõputöö teksti vastavasse tekstilahtrisse ning vajutama nupule *Analüüsi*. Seejärel hakkab serveril jooksev programm sisendit analüüsima. Sõltuvalt teksti pikkusest võib see võtta aega mõne minuti. Pikemate tekstidega võib analüüsiga minna aega kuni 5 minutit. Rakenduse kiirusest ja jõudlusest tuleb rohkem juttu peatükis 4.3. Samal ajal, kui analüüs toimub, näidatakse kasutajale laadimisanimatsiooni.

Joonis 5. Analüüsi tulemuse vaade

Kui analüüs on valmis, näidatakse kasutajale tulemuste lehte (vt Joonis 5). Tuuakse välja lausete arv, sõnade arv ja analüüsiks kulunud aeg. Kasutaja näeb kokkupandavate lahtritena viit põhilist analüüsi: umbisikulise tegumoe, lausepikkuse, puuduvate komade, kantseliidi ja sõnakorduste analüüsi. Kui vajutada hiirega kindla analüüsi peale, siis avaneb lahter, kus saab näha spetsiifilise analüüsi tulemusi. Soovi korral on võimalus uus analüüs teha.

4.3 Jõudlus

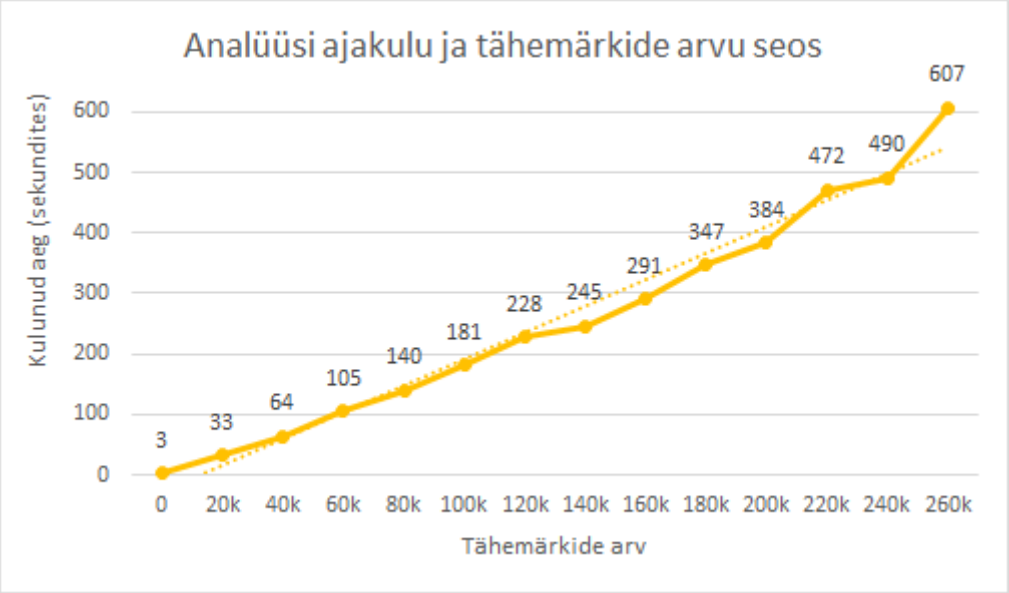
Tekstianalüüs on aeganõudev protsess. Selles peatükis on kirjeldatud, kuivõrd palju aega läheb tekstide analüüsimiseks ja vastuse tagastamiseks kliendile. Lisaks on välja toodud, kuidas riistvara mõjutab tekstianalüüsiks kuluvat aega.

Lõputöö analüsaator jookseb Ubuntu 18.04 operatsioonisüsteemiga virtuaalmasinal, millel on 4 tuuma ja 2 GB vahemälu. See on Estonian Scientific Computing Infrastructure[15] (ETAIS) veebiliidese kaudu saadud High Performance Computing (HPC) keskuse virtuaalmasin.

[15] Estonian Scientific Computing Infrastructure koduleht https://etais.ee/

Üks virtuaalmasin (edaspidi server) suudab korraga teenindada 4 erinevat klienti. Iga kliendi analüüsi jaoks antakse üks protsessori tuum seniks, kuni analüüs saab tehtud. Juhul, kui kliente on rohkem kui 4, peavad järgmised kliendid ootama, kuni üks hõivatud tuumadest vabaks saab.

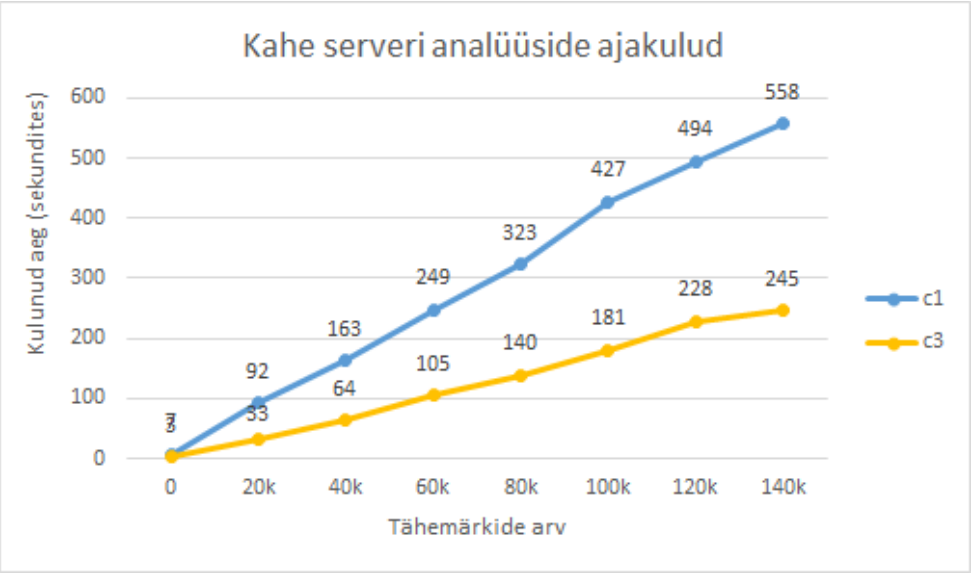
Katsetamisest selgus, et analüüsidele kuluv aeg sõltub kõige rohkem kahest faktorist: sisendteksti pikkusest ja serveri protsessorist. Katsetustes kasutati kahte erinevat serverit, mis olid mõlemad HPC keskusest saadud. Esimene server kasutas 2012. aasta Intel(R) Xeon(R) CPU E5-2650 protsessorit (edaspidi viidatuna kui c1) [6]. Teine kasutas see-eest uuemat ja võimsamat 2019. aasta AMD EPYC 7702 64-Core protsessorit (edaspidi viidatuna kui c3) [7]. Rakenduse *live*-server kasutab võimsamat c3 protsessorit.



Joonis 6. Analüüsi ajakulu ja tähemärkide arvu seos c3 protsessoriga

Joonis 6 näitab c3 protsessori analüüside ajakulu. Graafikust saab järeldada, et analüüsiks kuluva aja ja tähemärkide arvu vahel on võrdlemisi lineaarne seos. Iga kord, kui sisendtekst suurenes 20 000 tähemärgi ehk umbes 2450 sõna võrra, pikenes analüüsiks kuluv aeg üldiselt 30-50 sekundi võrra. Erandiks on viimane analüüs, kus 240 000 ja 260 000 tähemärgiliste tekstide analüüsi ajakulu pikenes 120 sekundi võrra. Tulemuste põhjal võiks arvestada, et 10 000-sõnalise lõputöö analüüs võtab aega umbes 160-180 sekundit. Pikemate tekstide puhul, kus on 260 000 tähemärki ehk umbes 31 000 sõna, võib analüüs võtta aega 607 sekundit ehk 10 minutit.

Järgnevalt selgitatakse, kuivõrd tugevalt mõjutab riistvara analüüsi jaoks kulunud aega.



Joonis 7. Kahe serveri analüüside ajakulu võrdlus

Serverite ajakulu võrdluse jooniselt (vt Joonis 7) on näha, et protsessor mõjutab tugevalt rakenduse ajakulu. Juba 140 000 tähemärgi juures läks nõrgema protsessoriga serveril analüüsiks 5 minutit kauem kui tugevama protsessoriga serveril. Katsetuste tulemustest saab järeldada, et rakenduse jõudlus on suuresti protsessoriga piiritletud (*CPU-bound*).

Selleks, et vältida veebiliidese üle koormamist ja päringu aegumist, on veebiliidesele lisatud tähemärkide piirang. Kui sisendtekstis on üle 200 000 tähemärgi, näidatakse kasutajale veateadet, et sisendtekst on liiga pikk. Niiviisi välditakse seda, et rakendus jooksutatakse liiga pikkade sisendtekstidega umbe.

Päringu aegumise vältimiseks on lisatud reegel, et ühe päringu jaoks kulutatakse maksimaalselt 15 minutit. See tähendab, et kui analüüs ei saa 15 minuti jooksul valmis, peatatakse päring.

Kui rakenduse poole pöördub API kaudu teine programm, siis tähemärgi piirangut ei ole. CGLearn pöördub API poole peatüki kaupa, ehk iga peatükk saadetakse analüüsimisele eraldi. Seetõttu on tõenäosus väike, et CGLearn saadetud analüüs rakenduse kokku jooksutab.

Programmi põhiline kasutusjuht on analüüsida lõputöö tekste, mis on automaatselt CGLearnist analüüsiks saadetud. Igal pühapäeva ööl saadetakse eestikeelsed lõputööd API-le, misjärel analüüsitakse kõik tööd. Nende analüüsimisega pole probleeme tekkinud.

4.4 Tulemuste salvestamine

Selleks, et jälgida veebiliidese kasutust ja analüüsida programmi tulemusi, on võimalik salvestada kõik veebiliidese kaudu tehtud analüüsid andmebaasi. Selleks on vaja juurkaustas olevas `env.py` failis märkida lipuke `LOG_TO_DATABASE` tõeseks. Sellisel juhul salvestatakse andmebaasi tabelitesse kõikide analüüsi tulemused, aga mitte originaalteksti ennast. Kasutajaid teavitatakse liideses tekstiga, et nende analüüside tulemusi logitakse.

Kuigi analüüside tulemused salvestatakse eraldi tabelitesse, on kõige mugavam tulemusi uurida HTML kujul ehk samal kujul, nagu kasutaja oma analüüsi tulemusi näeb. Seetõttu salvestatakse andmebaasi kasutaja tekstianalüüsi tulemus ka HTML kujul. Kasutades graafilist andmebaasiliidest nagu näiteks DBeaver Community[16], on võimalik eksportida kõik analüüsid ühte HTML faili. Seejärel saab kasutada loodud abimoodulit, et kõik analüüsid eraldada erinevateks HTML failideks. Selle kasutusjuhendi jaoks vaata Lisa III.

Tulemuste salvestamine oli kasulik, kui rakendust testiti kasutajatega. Niiviisi sai kokku viia kasutajate analüüside tulemused ja nende hinnangud rakendusele. Analüüside tulemuste nägemiseks vaata Lisa III.

4.5 Flask

Flask on vabavaraline Pythoni raamistik, millega saab lihtsasti luua veebirakendusi. Selle autor on Armin Ronacher ning see sai populaarseks seetõttu, et inimesed soovisid alternatiivi keerukaks peetud Django raamistikule[17] [8]. Tänapäeval on Flask üks enim kasutatud Pythoni raamistikke. Seda peetakse kergekaaluliseks, kuna ilma lisadeta põhiversioonil puuduvad andmebaasi abstraktsioonikiht, andmeväljade kontroll, autentimine ja muud funktsionaalsused, mida veebirakenduste loomises tihtipeale kasutatakse[18]. See-eest on võimalik lisada erinevaid kolmanda osapoole liideseid, mis võimaldavad neid funktsionaalsusi rakendada[18]. Lõputöös otsustati rakenduse *backendi* puhul Flaski kasuks, kuna selle kohta on palju dokumentatsiooni ning sellega on lihtne ja

[16] DBeaver Community koduleht https://dbeaver.io/
[17] Django koduleht https://www.djangoproject.com/
[18] Flask vikipeedia leht https://en.wikipedia.org/wiki/Flask_(web_framework)#cite_note-4

kiire luua REST arhitektuuril põhinevat arendusserverit. Kasutati kõige uuemat versiooni 1.1.1.

4.6 EstNLTK

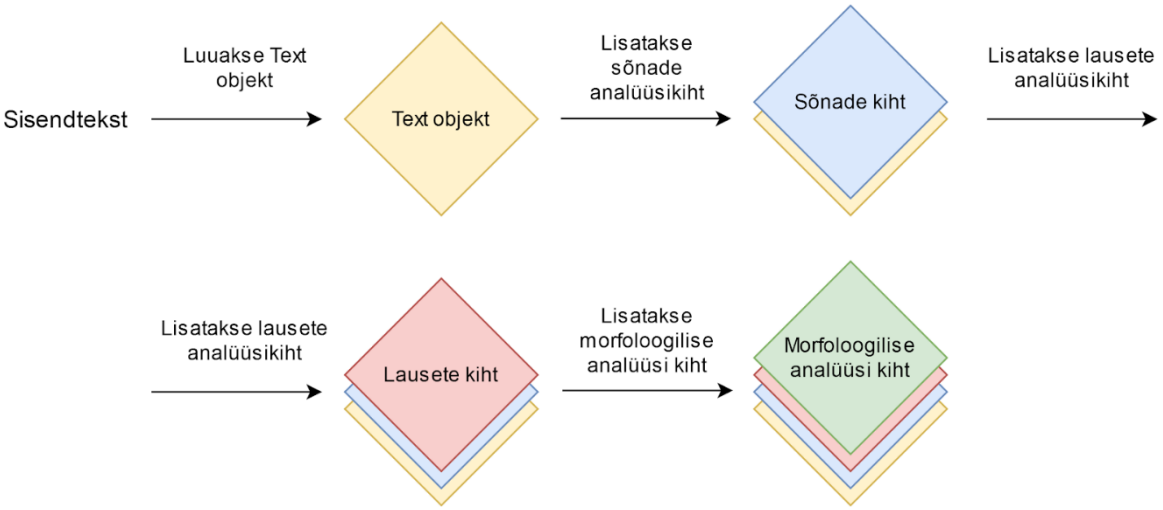
EstNLTK (*Estonian Natural Language Processing Toolkit*) on Pythonis kirjutatud teekide kogumik, mis pakub erinevaid võimalusi eesti keele töötluseks. Selle arendamist juhendab Tartu Ülikooli turvalise andmekaeve vanemteadur Sven Laur[19] ning seda finantseerib Eesti Haridus- ja teadusministeerium „Eesti keeletehnoloogia“ programmi raames [9]. Projekti eesmärk on ühendada ja parandada olemasolevaid keeletöötluse võimalusi selleks, et need oleksid kergesti kättesaadavad kõikidele soovijatele [9].

Teek võimaldab kasutada peamisi keeletehnoloogilisi operatsioone. Näiteks toetab EstNLTK osalausestamist, lausestamist ja sõnestamist, morfoloogilist analüüsi, ajaväljendite ja nimeolemite tuvastust, verbi- ja nimisõnafraaside tuvastust ja palju muud [9]. Antud lõputöös olid EstNLTK funktsionaalsustest suurima vaatluse all osalausestamine, sõnestamine, morfoloogiline analüüs ja lausestamine. Kõikide funktsionaalsuste kohta on olemas põhjalikud kasutusjuhendid ja dokumentatsioon.

Käesolevas lõputöös otsustati kasutada EstNLTK versiooni 1.6. Võimalik oli kasutada ka 2017. aastal lõplikult valmis saanud EstNLTK versiooni 1.4.1. Esialgu katsetati programmi EstNLTK 1.4.1 versiooniga, ent hiljem otsustati siiski hilisema versiooni kasuks. Otsuse tegemisel aitas kaasa EstNLTK arendaja Siim Orasmaa, kes tõi versioon 1.6 suurimateks eelisteks välja ühtlasema ja selgema liidese ning parema sõnestamise ja lausestamise. Lisaks saab 1.6 regulaarselt uuendusi, mistõttu on tulevikus rohkem võimalusi uuteks analüüsideks.

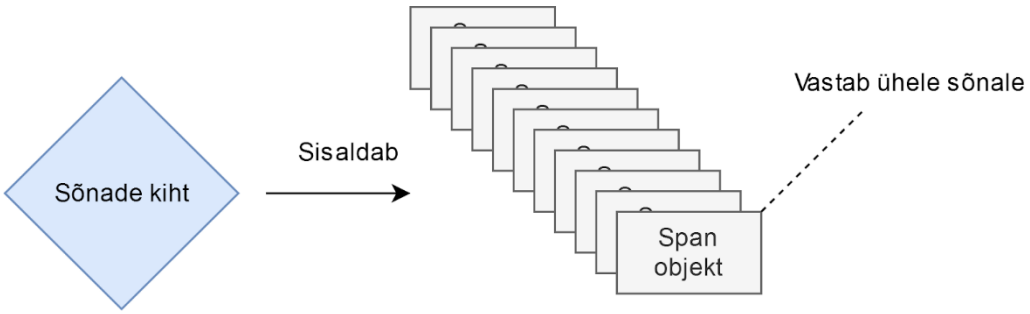
Versioon 1.6 teksti analüüsitakse ja töödeldakse kihilise struktuuri põhjal (vt Joonis 8). Originaalsest sisendtekstist luuakse `Text` objekt, millele saab erinevaid analüüsikihte lisada. Näiteks saab lisada sõnade analüüsikihi `words`, mis loob järjendi `Span` objektidest. Joonis 8 näitab, kuidas `Text` objektile kihte juurde lisatakse. Käsk `tag_layer` lisab `Text` objektile kõik põhilised analüüsikihid.

[19] EstNLTK autorid https://estnltk.github.io/estnltk/1.4.1/authors.html



Joonis 8. EstNLTK kihiline struktuur

Sõnade analüüsikihis kapseldab iga `Span` objekt ühte teksti sõna (vt Joonis 9). Igal `Span` objektil on omakorda olemas atribuudid, millega saab kätte näiteks sõnaalguste ja -lõppude asukohad (indeksid) või sõna originaalteksti.



Joonis 9. Sõnade kihi jagunemine `Span` objektideks

Erinevatel kihtidel on `Span` objektidel kihispetsiifilised atribuudid. Analoogselt sõnade kihile, saab `Text` objektile lisada ka näiteks `sentences` ja `morph_analysis` analüüsikihid. Esimene neist loob analüüsikihi lausete eraldamiseks ja analüüsimiseks, teine võimaldab uurida sõnade morfoloogilisi atribuute. Kihis `sentences` kapseldab `Span` objekt ühte lauset ja `morph_analysis` kihis ühe sõna morfoloogilist analüüsi. Kihi `morph_analysis` kaudu on võimalik saada kätte sõnade algvorm ja sõnaliik. `morph_analysis` kiht oli lõputöös korduvalt kasutatud sõnade analüüsi puhul võtmetähtsusega.

Neid kihte ja EstNLTK teeki kasutati kõigis viies analüsaatoris.

5. Analüsaatorid

Lõputööde analüsaatori programm jaguneb omakorda väiksemateks analüsaatoriteks, millest igaüks otsib sisendtekstis kindlat tüüpi murekohta. Kõik analüsaatorid asuvad programmi kaustas `Services/Analysis`. Järgnevates alapeatükkides tuleb juttu teksti eeltöötlusest, analüsaatoritest, nende loogikast ja tulemustest.

Katsetades anti analüsaatorile veebiliidese või API kaudu sisendtekst. Kui veebiliidest ei olnud veel loodud, pöörduti API poole programmiga Postman[20]. Katsetuste tulemuste põhjal tehti järeldused, kas analüsaatori tulemused on rahuldavad või mitte. Kui tulemus ei olnud rahuldav, muudeti analüsaatorite reegleid või lähenemisviisi. Protsessi korrati, kuni analüsaatorid andsid rahuldavaid tulemusi.

Kui analüsaatorid olid valmis, paluti tudengitel programmi testida. Testimisest on rohkem kirjutatud peatükis 6.

5.1 Sisendteksti eeltöötlus

Enne seda, kui analüsaatorid saavad tööle hakata, peab teksti eeltöötlema. Eeltöötlusega tegeleb moodul `preprocessor`. Eeltöötlus tähendab seda, et sisendtekstist luuakse `Text` objekt, misjärel lisatakse objektile kõik EstNLTK analüüsikihid (vt Joonis 8). Seda tehakse EstNLTK funktsiooniga `tag_layer`.

Seejärel kaardistatakse kõikide lausete ning sõnade algus- ja lõpuindeksid. Indeksid tähistavad sõna või lause asukohta tekstis. Lause algusindeks 140 tähistab seda, et lause algab originaaltekstis 140. tähemärgilt.

Luuakse eraldi objekt `PreprocessedText`, kuhu salvestatakse kõik tekstis leiduvad sõnad ja laused. Niiviisi on võimalik näiteks lause kaudu kätte saada kõik selle sõnad ilma, et peaks hiljem uuesti teksti analüüsima. See objekt antakse kõikide järgnevate analüsaatorite käivitusfunktsioonidele kaasa.

Eeltöötlus on oluline, kuna analüüsikihtide lisamine võib sõltuvalt sisendteksti pikkusest võtta palju aega. Kuna kõikidel analüsaatoritel on eeltöötlusest saadud infot vaja, on seda mõistlikum teha kohe alguses ainult ühe korra. Niiviisi on rakendus tunduvalt kiirem, sest enam ei kutsuta korduvalt välja aeganõudvaid EstNLTK funktsioone.

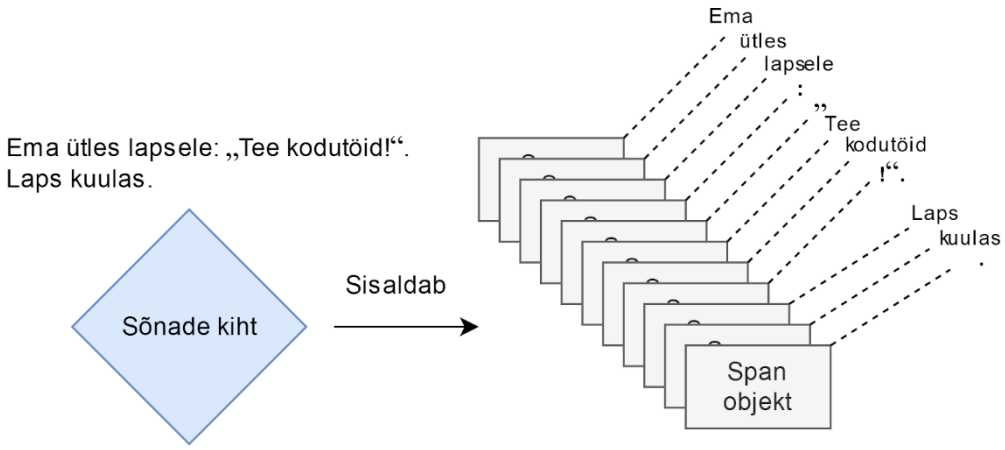
[20] Postman koduleht https://www.postman.com/

5.2 Mõiste- ja tsitaadituvastaja ning -eemaldaja

Programmi analüsaatorites on vaja eristada, kas tekstilõik on mõiste või tsitaat. Kuna EstNLTK teegis ei leidu nende tuvastamiseks sääraseid funktsionaalsusi, otsustati lõputöös luua klassid `QuoteAnalyzer` ja `QuoteRemover`. Need klassid on abianalüsaatorid analüsaatorite jaoks, millest on rohkem kirjutatud edasistes alapeatükkides.

`QuoteAnalyzer` on klass, mis tuvastab, millised sõnad on tekstis tsitaadis ning millised ei ole. Programmi põhiline ja olulisim funktsioon on `is_word_in_quotes`, mis saab argumendiks ühe EstNLTK sõnade kihi `Span` objekti (vt Joonis 9).

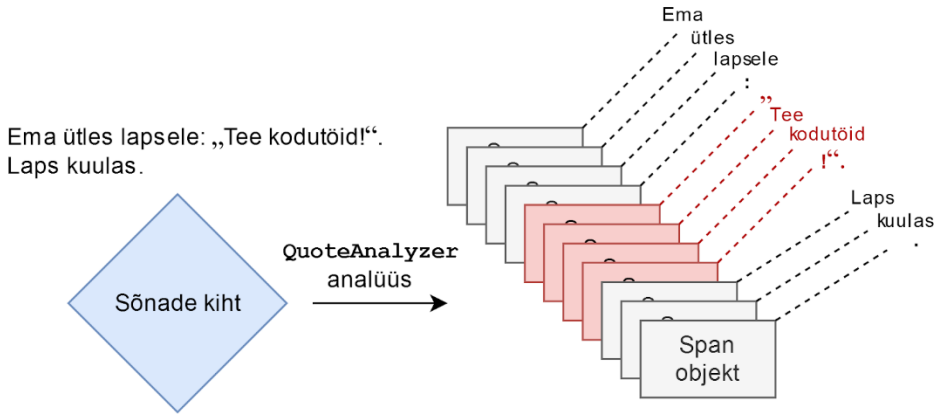
On oluline meeles pidada, et EstNLTK lisab sõnade kihti `Span` objektideks kõik sõnad, aga ka tekstis leiduvad sümbolid. Näiteks on sõnade kihis eraldi sümbolitena jutumärgid, punkt, küsimärk ja hüüumärk (vt Joonis 10).



Joonis 10. Sõnade kiht tekstiga *Ema ütles lapsele: „Tee kodutöid!”. Laps kuulas.*

Oluline on teada, et sõnade kiht võib eraldada sümboleid erinevalt. Eraldamine sõltub sümbolitest, mis on sisendtekstis üksteisega kõrvuti. Näiteks näitab Joonis 10, et otsekõnet alustav jutumärk ja koolon on kihis üksteisest eraldi. See-eest on hüüumärk, lõpetav jutumärk ja punkt arvatud üheks „sõnaks“. Selleks, et `QuoteAnalyzer` õigesti töötaks, peab klassi loogika arvestama säärase käitumisega.

`QuoteAnalyzer` klassi kasutamiseks vaadatakse tsükliga läbi kõik sõnade kihi elemendid ehk `Span` objektid. Iga element antakse argumendiks funktsioonile `is_word_in_quotes`, mis seejärel tagastab väärtuse `True` või `False` vastavalt sellele, kas sõna oli jutumärkide vahel või mitte.



Joonis 11. `QuoteAnalyzer` analüüsi tulemus

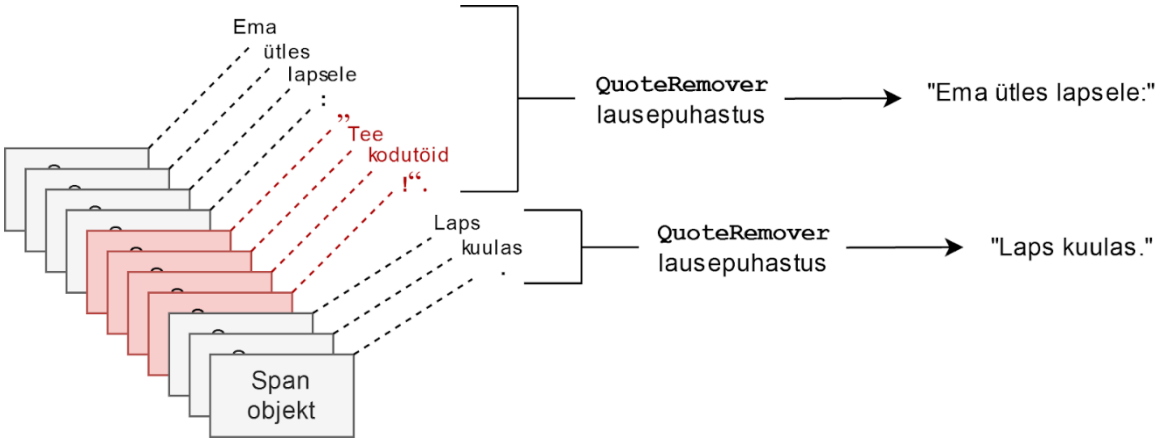
Joonis 11 illustreerib analüsaatori lõpptulemust, kus punasega on märgitud jutumärkide vahel olevad sõnad.

`QuoteAnalyzer` võimekus ei ole piiratud ainult ühe lausega, vaid see suudab analüüsida tervet sisendteksti. Kui tsitaat on pikem kui üks lause, märgitakse ära, et kõik jutumärkide vahel olevad laused on osa tsitaadist.

Mõnes analüsaatoris piisab sellest, kui tuvastatakse ainult sõnad, mis on jutumärkide vahel. Näiteks on üks selline analüüs umbisikulise tegumoe analüüs (vt peatükk 5.4). Selles on vaja vaid veenduda, et mina- või meie-vormis sõna ei oleks mõiste või osa tsitaadist. Sellistel puhkudel pole vaja edasist tekstitöötlust ehk jutumärkide vahel oleva teksti kustutamist.

Sellegipoolest on mõnes analüsaatoris, näiteks lausepikkuse- ja keerulisuse analüsaatoris, vaja jutumärkide vahel olev tekst eemaldada. Selle jaoks on loodud klass `QuoteRemover`.

Klassi põhiline funktsioon on `remove_quoted_parts_from_sentence`. Funktsioon võtab argumendiks vaadeldava lause ja indeksid sõnade jaoks, mis ei ole jutumärkide vahel. Need indeksid on saadud `QuoteAnalyzer` analüüsi tulemusena. Seejärel luuakse indekseid ja sisendteksti sõnade kihti kasutades uus sõne, mis vastab esialgsele lausele, ent millest on jutumärkide vahel olev tekst eemaldatud. Joonis 12 demonstreerib funktsiooni lõpptulemust.



Joonis 12. `QuoteRemover` lausepuhastuse lõpptulemus

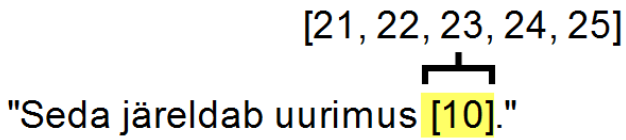
`QuoteRemover` on kõige olulisem lausepikkuse- ja keerulisuse analüsaatoris, kus on vaja, et jutumärkide vahel olev tekst ei mõjutaks kuidagi lause keerukust. Selle kohta on rohkem kirjutatud peatükis 5.6.

5.3 Viite-eemaldaja

Programm vajab analüsaatorit, mis tuvastab ja eemaldab lausest kõik viited. Selle vajalikkusest tuleb juttu peatükis 5.6. Kuna EstNLTK teegis ei leidu viidete eemaldamise funktsionaalsust, otsustati luua eraldi klass `CitationRemover`. Sarnaselt `QuoteAnalyzer` ja `QuoteRemover` klassidega, on `CitationRemover` teistele analüsaatoritele abiklassiks.

Kui `QuoteAnalyzer` ja `QuoteRemover` töötasid sõnapõhiselt, ehk vaatluse all oli iga sõna, siis `CitationRemover` töötab karakteripõhiselt. Kasutatakse regulaaravaldist, mis suudab leida levinumad teadusteksti viitamisstiilid. CitationRemover tuvastab numbrilise viitamise ja tekstisisese viitamise koos autori nime ja aastaarvuga.

Klassi tööd alustab funktsioon `get_sentence_without_citations`, millele antakse argumendina vaadeldav lause sõnekujul. Seejärel otsitakse regulaaravaldisega üles kõik viite sõneindeksid lauses (vt Joonis 13). Arvesse võetakse ka *white-space*'i, mis on kummalgi pool viidet.



Joonis 13. Viidete leidmine lausest

Kui viidete indeksid on leitud, luuakse uus tühi sõne. Esialgses lauses vaadatakse iga karakterit, pidades järge käesoleval indeksil. Kui käesolev indeks ei kuulu viidete indeksite järjendisse, siis lisatakse karakter uude lausesse. Tulemuseks on lause, millest on viide eemaldatud.

5.4 Umbisikulise tegumoe analüsaator

Teadustekst peaks üldise arusaama järgi olema objektiivne ja neutraalne nii keeleliselt kui ka sisult [10]. Mitmete teadustekstide autorid nagu näiteks Sirkka Hirsjärvi, Pirkko Remes ja Paula Sajavaara väidavad, et teadusteksti autor peab enda isikut tagaplaanil hoidma [11]. Lugeja tähelepanu peaks nende sõnul püsima käsitletaval teemal ning autori keelekasutus ja isik pärsivad tekstile keskendumist.

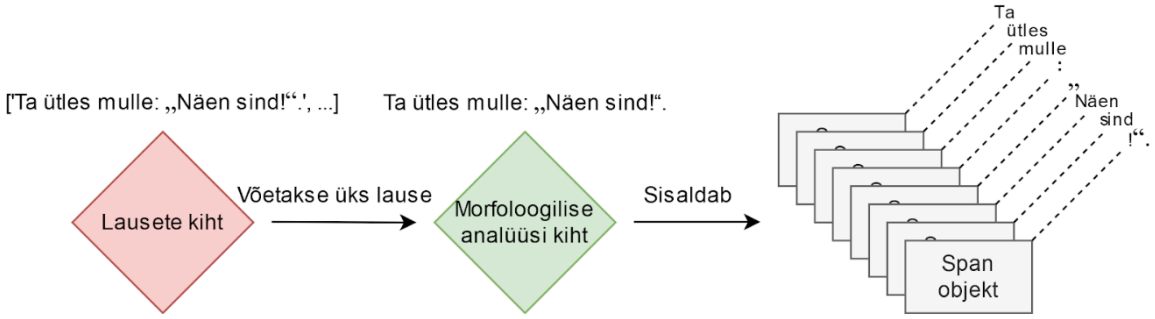
Eesti keeles on nii isikuline kui umbisikuline tegumood, millest viimast kasutatakse selleks, et kirjutaja isikut kõige paremini tagaplaanile jätta [10]. Erinevates instituutides võib lõputöö kirjutamisnõuetes tegumoe nõue varieeruda, aga arvutiteaduse instituudis on

soovituslik kasutada mina-vormi asemel umbisikulist tegumoodi. See tähendab, et iseendale viidates on lõputöös soovituslik kasutada näiteks varianti *arvatakse*, mitte *mina arvan*.

Lõputöö analüsaatori programmis loodi umbisikulisuse kontrollimiseks Pythoni moodul `impersonality_analyzer`, mis analüüsib ja otsustab, kas lõputöö sisendtekst on täielikult umbisikulises tegumoes kirjutatud.

Esmaselt kasutatakse EstNLTK funktsiooni `tag_layer`, et tekst lausestada ja teha tekstile morfoloogiline analüüs. Tulemuseks on `Text` objekt, millel on olemas lausete ja morfoloogilise analüüsi kihid. Umbisikulisuse analüüsi käivitab `analyze` funktsioon, mis saab argumentideks originaalteksti ja EstNLTK `Text` objekti lausete kihi. Igal lausel on omakorda olemas morfoloogilise analüüsi ja sõnade kiht.

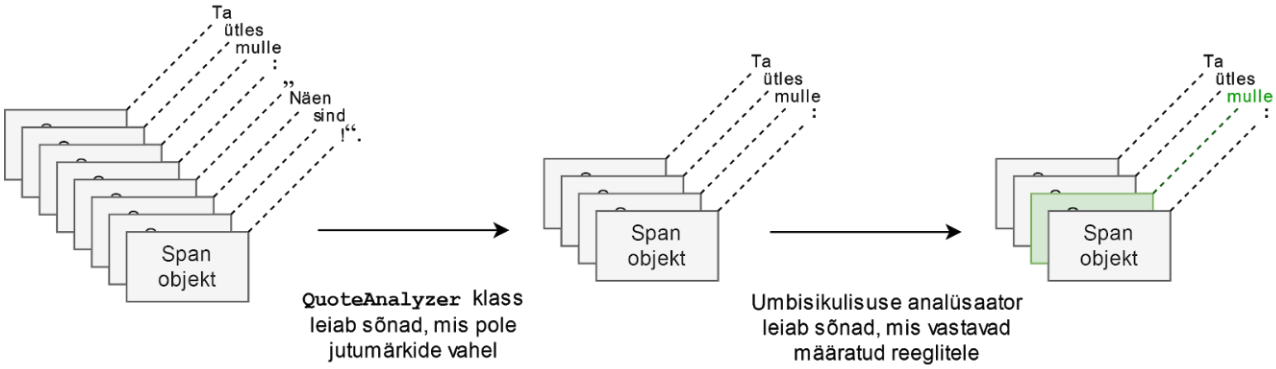
Seejärel vaadeldakse iga lauses sisalduva sõna morfoloogilist analüüsi. Analüüsi kihis on `Span` objektid, kus iga objekt vastab ühele sõnale (vt Joonis 14). Igal objektil on morfoloogilise analüüsi kihile omased atribuudid, nagu näiteks sõna lemma, sõnajuur ja sõnaliik.



Joonis 14. Morfoloogilise analüüsi kiht

Kontrollitakse morfoloogilise analüüsi kihist `Span` objektidest saadud sõnalõppu, -liiki ja -juurt. Uuritakse, kas sõna vastab reeglitele, mis on mina- või meie-vormi määramiseks loodud. Reeglite järgi on sõna mina- või meie-vormis siis, kui **sõna sõnajuur on *mina* või see on tegusõna ning selle lõpp on kas *-sin, -in, -n, -sime, -ime, -me***. Kui need reeglid on vaadeldava sõna jaoks täidetud, siis otsustatakse, et sõna on mina- või meie-vormis.

On võimalik, et kuigi sisendteksti autor ei kasutanud lõputöös mina- või meie-vormi töö sisupeatükkides, siis sellegipoolest leidub mina-vorm kas tsitaadis või mõistes. Selleks, et vältida valepositiivseid tulemusi, tehakse iga sõna puhul kontroll, kas see on mõiste või osa tsitaadist. Selle kontrollimiseks on loodud eraldi klass `QuoteAnalyzer` (vt peatükk 5.2). Kui `QuoteAnalyzer` tuvastab, et sõna on jutumärkide vahel, siis umbisikulise tegumoe analüsaator seda arvesse ei võta.



Joonis 15. Analüsaator leiab lausest mina- või meie-vormis sõnad

Umbisikulise tegumoe analüüs tagastab lõpuks kõik laused, kus eirati tsitaadi- või mõisteväliselt umbisikulist tegumoodi. Iga lause puhul tuuakse välja sõnad, mis ei ole mina- või meie-vormis (vt Joonis 15).

Analüsaatoril on see-eest üks suurem puudus. Nimelt ei suuda analüsaator sisendteksti puhul eristada, kas tekst on kaldkirjas või mitte. Niiviisi võib lihtsasti tekkida valepositiivseid tulemusi, kus teksti autor on mõiste või tsitaadi jaoks jutumärkide asemel kasutanud kaldkirja. Kuigi kaldkirja on lubatud kasutada nii tsitaadi kui ka mõiste jaoks, kohtleb analüsaator seda siiski tavalise tekstina. Selle lahenduse kohta on pikemalt juttu peatükis 7.2.

CGLearn saadab API-le teksti peatükkide kaupa analüüsimiseks. Olukorras, kus mina- või meie-vorm on sisupeatükist väljaspool, saab CGLearn ignoreerida vastava peatüki korral umbisikulisuse analüüsi tulemusi. Seetõttu ei märgita CGLearn õpikeskkonnas näiteks kokkuvõttes mina- või meie-vormi kasutamist veaks.

Isiklikest katsetustest ja kasutajatega testimisest selgus, et analüsaator töötab hästi. See leidis etteantud lõputööde tekstides üles kõik laused, kus esines mina- või meie-vorm.

5.5 Korduvate sõnade analüsaator

Üks levinumaid stiilivigasid on see, et tekstis kasutatakse teatud sõnu liiga palju. Hea stiil näeb ette, et välditakse sõnakordusi olukordades, kus see on võimalik.

Lõputöö analüsaatoris loodi moodul `overused_word_analyzer`, mis toob välja ülekasutatud ehk liiga palju korduvad sõnad. Ülekasutatud sõnade analüsaatori loogika põhineb suuresti GitHubi kasutaja omerdemirkan vabavaralisel programmil nimega Synonymy[21]. Synonymy programm on kirjutatud programmeerimiskeeles JavaScript[22] ning töötab ainult inglise keelega. Lõputöös loodud ülekasutatud sõnade analüsaator on see-eest kirjutatud programmeerimiskeeles Python ja analüüsib hetkel vaid eesti keelt. Inglise keelele laiendamise võimalusest on räägitud peatükis 7.3. Võrreldes programmiga Synonymy, on tehtud lisaks edasiarendusi ja kohandusi vastavalt lõputööde analüsaatori API vajadustele.

Selleks, et leida tekstist üles ülekasutatud sõnad, on vaja andmestikku lemmadest ja nende esinemiste sagedustest tekstides. Lõputöös kasutatakse Eesti Keeleressursside Keskuse[23] (EKRK) koostatud teaduskirjanduskorpuse lemmade sagedusloendit[24]. Kuni töö katsetamisfaasi viimase iteratsioonini kasutati EKRK tasakaalus korpuse lemmade sagedusloendit[25]. Tasakaalus korpus on segu ajakirjandus-, ilukirjandus- ja teaduskirjanduskorpuse lemmadest ja nende sagedustest, mistõttu leiti, et see ei esinda niivõrd hästi teaduskeelt. Seetõttu valiti andmeteks siiski teaduskirjanduskorpuse lemmad ja nende esinemissagedused tekstides.

Tagasiside peatükis on näidatud, kuidas kasutajad hindasid analüüsi tulemusi siis, kui andmebaasis kasutati tasakaalus korpuse andmestikku. Võrdluseks on toodud välja kasutajate hinnangud siis, kui kasutati teaduskirjanduse korpuste andmestikku.

[21] Synonymy programm https://github.com/omerdemirkan/synonymy
[22] Javascript koduleht https://www.javascript.com/
[23] EKRK koduleht https://keeleressursid.ee/et/
[24] EKRK teaduskirjanduskorpuse lemmade sagedusloend https://keeleressursid.ee/images/cl-ut-ee/sagedusloendid/lemma_tea_kahanevas.txt
[25] EKRK tasakaalus korpuse lemmade sagedusloend https://keeleressursid.ee/images/cl-ut-ee/sagedusloendid/lemma_kahanevas.txt

Üldiselt võib öelda, et loomulikes keeltes järgivad sõnade esinemissagedused universaalselt Zipf seaduspärasust [12]. Sellist fenomeni on tähele pandud isegi praeguseks väljasurnud ja senini täielikult tõlkimata keeltes, nagu näiteks Meroiti (ingl *Meroitic*) keeles [13]. Zipf seadus ütleb, et sõna sagedushinnangut võimalik leida järgneva valemiga:

$$f(r) \approx \frac{1}{r}$$

kus r on sõna sagedushinnang (*frequency rank*) ja $f(r)$ on selle sõna sagedus korpuses [12].

Iga sõna puhul, mis sisendtekstis esineb, arvutatakse selle sõna tekstisisene sagedushinnang. Seejärel leitakse sõna tegelik sagedushinnang teadustekstide korpusest. Tulemusena saab võrrelda, kas sõna esineb sisendtekstis märgatavalt rohkem kui on tavaliselt teadustekstides oodatud. Kui sõna esineb tõepoolest harilikust märgatavalt rohkem, siis märgistab analüsaator sõna ülekasutatuks.

Esmaselt on vaja sõnade sagedushinnangute leidmiseks tekst lemmatiseerida. Uurimusest on selgunud, et Zipf seaduspärasus kehtib ka lemmatiseeritud teksti puhul [14]. Lemmatiseeritud sõnad saab kätte EstNLTK morfoloogilise analüüsi kihist.

Ülekasutatud sõnade tuvastamisel leitakse sõnad, mille algvormi ehk lemmat on tekstis oodatust rohkem. See tähendab, et ühe sõna erinevaid sõnavorme loetakse siiski ühe sõnana. Näiteks on tähenduslikus mõistes võrdväärsed sõnad *autole* ja *autosse*, olgugi et nad on erinevates käänetes.

Korduvate sõnade analüsaator töötab järgnevalt:

1) Lemmatiseerimine

Kasutatakse EstNLTK morfoloogilise analüüsi kihti, et leida iga sõna lemma. Luuakse sõnastik, kus võtmeteks on lemmad ning väärtusteks kõik sõnad (koos algus- ja lõpuindeksitega), mis sellele lemmale tekstis vastavad (vt Joonis 16).

```
'objekt': {<Word (id, None, text: objekt, part_of_speech: S, position: [16906, 16912], sentence_index: 157, sentence_position: [16864, 16958])>,
           <Word (id, None, text: objekt, part_of_speech: S, position: [10882, 10888], sentence_index: 95, sentence_position: [10857, 10889])>,
           <Word (id, None, text: Objektil, part_of_speech: S, position: [10890, 10898], sentence_index: 96, sentence_position: [10890, 10973])>,
           <Word (id, None, text: objekti, part_of_speech: S, position: [11221, 11228], sentence_index: 99, sentence_position: [11115, 11235])>,
           <Word (id, None, text: objekt, part_of_speech: S, position: [11333, 11339], sentence_index: 100, sentence_position: [11236, 11340])>,
           <Word (id, None, text: objektidest, part_of_speech: S, position: [17029, 17040], sentence_index: 158, sentence_position: [16959, 17041])>,
           <Word (id, None, text: objektile, part_of_speech: S, position: [17080, 17089], sentence_index: 159, sentence_position: [17042, 17113])>,
           <Word (id, None, text: objektile, part_of_speech: S, position: [17142, 17151], sentence_index: 160, sentence_position: [17114, 17181])>,
           <Word (id, None, text: objekt, part_of_speech: S, position: [17260, 17266], sentence_index: 162, sentence_position: [17195, 17303])>,
```

Joonis 16. Sõnastik, kus lemmale vastavad kõik lemma esinemiskorrad algtekstis

2) **Filtreerimine**

Lemmade sõnastikust eemaldatakse lemmad, mis on kas stoppsõnad või esinevad tekstis vähem kui 7 korda. Seejärel eemaldatakse sõnad, mida EKRK sagedusloendis ei ole. Kuna EKRK loendis on üle 140 000 lemma, siis on selliseid sõnu vähe.

3) **Sagedushinnangu leidmine ja sõna ülekasutatuks märkimine**

Iga lemma jaoks leitakse selle sagedus tekstisiseselt. Võrreldakse lemma sagedushinnangut ja selle tegelikku sagedust tekstis. Jagatakse tekstisisene sagedus korpusesisese sagedushinnanguga. Kui tulemus on suurem kui 5 ehk tekstisiseselt on kasutatud lemmat 5 korda rohkem kui oodatud, märgitakse lemma ülekasutatuks.

4) **Sorteerimine**

Ülekasutatud lemmad sorteeritakse sagedushinnangu järgi kahanevalt. See tähendab, et esimene lemma on tekstis kõige rohkem ülekasutatud. Info liigsuse vältimiseks jäetakse järele ainult 8 ülekasutatud lemmat.

5) **Sõnaklastrite leidmine**

Iga ülekasutatud lemmaga kaasneb ka info sõnade ja nende positsioonide (indeksite) kohta, mis sellele lemmale vastavad. Vaadatakse iga sõna algus- ja lõpuindeksit ja tuvastatakse selle põhjal sõnaklastrid. Sõnaklastritest tuleb juttu järgnevates lõikudes.

6) **Tulemuse tagastamine**

Analüsaator tagastab analüüsi tulemusena kõik ülekasutatud sõnad.

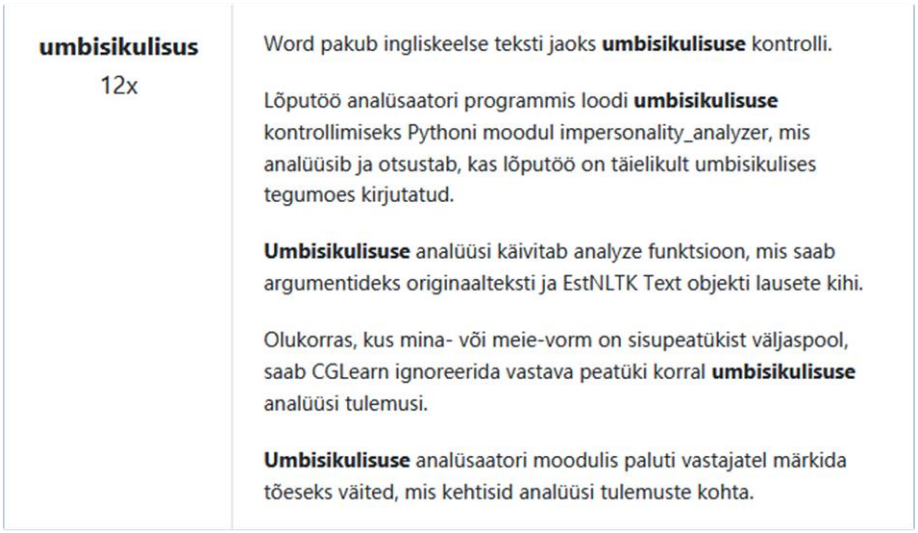
Korduvate sõnade analüüs tuvastab sõnad, mida on terve teksti siseselt kasutatud harilikust 5 korda rohkem. Synonymy põhiline eesmärk oli sama. See-eest märgati nii korduvate sõnade analüüsi kui Synonymy tulemusi uurides, et lugejat tegelikult ei häiri, kui korduvaid sõnu on kasutatud erinevates lõikudes. Kui sõnade omavaheline kaugus oli üksteisest suurem, ei mõjunud need lugejale enam niivõrd korduvatena. Lugejat häirib pigem see, kui korduv sõna esineb mitu korda ühes tekstilõigus või -osas. Seetõttu leitakse korduvate sõnade jaoks tekstisiseselt sõnaklastrid.

Sõnaklastriteks peetakse tekstiosasid, kus ühise lemmaga sõnad ei ole üksteisest kaugemal kui 300 karakterit. Aken 300 karakterit sai valitud empiiriliselt katsetades. Analüüsis

leitakse üles laused, mis kuuluvad ühte sõnaklastrisse. Kõik vaadeldavad sõnad, mis on üksteisele lähemal kui 300 karakterit, lähevad ühte kasti. Kui kastis on vähemalt 4 sõna, siis tuvastab analüüs klastri. Iga kastis oleva sõna puhul leitakse lause, kuhu sõna kuulub. Kui kõik laused on tekstis järjestikku, näidatakse kasutajale klastriks määratud lauseid järjest. Kui mõne lause puhul on vahelause, kus vaadeldavat sõna ei leidu, asendatakse tulemustes too lause märgiga „[...]".

Katsetades selgus, et arvesse ei tohiks võtta jutumärkide vahel esinevaid sõnu, sest need on kas tsitaadid või nimed. Näiteks oli ühes katsetamisel kasutatud arvutiteaduse instituudi lõputöös mainitud ainet „Programmeerimise alused" 31 korda, mistõttu märgiti ka lemma *programmeerimine* ülekasutatuks. Kuna teksti loonud autor ei saa aine nime muuta, ei tohiks sõna *programmeerimine* selle põhjal ülekasutatuks märkida. Seetõttu otsustati, et jutumärkide vahel olevaid sõnu ülekasutatud sõnade analüüsis arvesse ei võeta.

Lisaks selgus, et pikkade tekstide puhul võib mingit sõna esineda paljudes lausetes. Kui näidata kõiki lauseid kasutajale korraga, võib see tulemuse analüüsi muuta tülikaks ja segaseks. Seetõttu, kui üks lemma sisaldub enam kui 20 lauses, näidatakse kasutajale ainult esimest 20 lauset, milles sõna sisaldub. Seejärel saab kasutaja vajutada nupule „Näita rohkem lauseid", et näha kõiki lauseid, milles sõna sisaldub.



Joonis 17. Korduvate sõnade analüsaatori tulemus veebiliideses

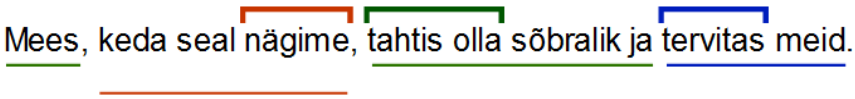
Joonis 17 näitab korduvate sõnade analüsaatori tulemust veebiliideses. Välja on toodud ülekasutatud sõna, selle kasutamiste arv ja laused, kus sõna esines.

5.6 Lausepikkuse ja -keerulisuse analüsaator

Emakeeleseltsi väitel peab selge tekst olema lihtsa ja loogilise lauseehitusega [15]. Kuna pika lause puhul on lause ülesehitus keeruline, siis ei pruugi tekkida üht mõttetervikut [15]. Ühte keerulisesse lausesse proovitakse siduda mitu erinevat mõtet. Tulemus on see, et lugejale jääb lause sisu arusaamatuks. Säärased laused pärsivad teksti loetavust ja selgust.

Lõputöö analüsaatoris kontrollib lausepikkust moodul `sentences_analyzer`. Moodul analüüsib igat lauset ja teeb selle põhjal ettemääratud reeglite järgi otsuse, kas lause on liiga pikk või mitte. Esiteks vaadeldakse lauseid, mis on saadud sisendteksti eeltöötlusest. Iga lause puhul leitakse osalaused ja lause verbiahelad (ingl *verb chains*).

Verbiahelad on lauseosad, mis koosnevad ühest või enamast abitegusõnast ja ühest põhilisest tegusõnast, mis annab edasi lause peamist mõtet[26]. Näiteks lauses *Ta hakkas süüa tegema* on verbiahel *hakkas tegema*.



Joonis 18. Osalausestamine ja verbiahelate leidmine

Osalausestamiseks kasutatakse EstNLTK `ClauseSegmenter` klassi ning verbiahelate leidmiseks EstNLTK `VerbChainDetector` klassi. Lauses *Mees, keda seal nägime, tahtis olla sõbralik ja tervitas meid* leidub `ClauseSegmenter` sõnul 3 osalauset. Iga verbiahela puhul kaardistatakse ära, millisesse osalausesse see kuulub. Nagu näitab Joonis 18, siis pärast osalausestamist ja verbiahelate kaardistamist on tulemuseks see, et esimeses osalauses *Mees tahtis olla sõbralik ja* on tuvastatud verbiahel *tahtis olla*. Teises osalauses *keda seal nägime* on leitud verbiahel *nägime* ja kolmandas osalauses *teretas meid* on verbiahel *teretas*.

Vastava info põhjal tehakse otsus, kas lause on liiga pikk või keeruline. Otsustamisel peab lause vastama kindlatele ettemääratud reeglitele. Programmi algfaasis kasutati otsuse tegemisel reeglit, et lause on liiga pikk või keeruline juhul, kui osalausete arv on 5 või rohkem ning vähemalt pooltes osalausetes on verbiahel.

[26] Easypacelearning lehekülg teemal verbid https://www.easypacelearning.com/all-lessons/grammar/78-verbs

Katsetades selgus, et sellise reegli puhul võivad pikad laused jääda pikaks märkimata. Näiteks kui lauses oli 10 osalauset, aga vaid neljas neist leidus verbiahel, siis lauset ei märgitud pikaks.

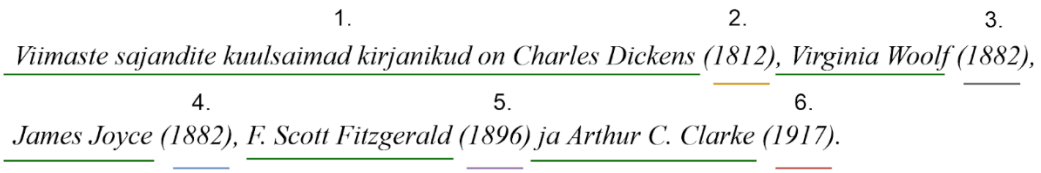
Seega otsustati uue reegli kasuks. Lause on programmi mõistes keeruline tingimusel, **kui osalausete arv on 5 või rohkem ning vähemalt kahes erinevas osalauses on verbiahel.**

Osalausetes verbiahelate sisaldumise kontroll täidab mitut eesmärki. Esiteks, verbiahel osalauses tähendab seda, et osalause tutvustab lausesse mingisugust uut mõtet. See muudab lause keerulisemaks ja seetõttu ka tõenäoliselt vähem loetavamaks. Teiseks, lauses võib olla palju osalauseid, aga ainult vähestes osalausetes leidub verbiahel. Sellisel juhul on suur tõenäosus, et tegemist on loeteluga. Kuna loetelust ei ole üldiselt lugejal keeruline aru saada, ei tohiks ka lausepikkuse ja -keerulisuse analüsaator seda keerulise lausena välja tuua.

Tuleb ka arvestada EstNLTK osalausestaja iseärasustega, mis võivad analüüsi tulemusi mõjutada. Näiteks ei tohiks keeruliseks märgistada lauset:

Viimaste sajandite kuulsaimad kirjanikud on Charles Dickens (1812), Virginia Woolf (1882), James Joyce (1882), F. Scott Fitzgerald (1896) ja Arthur C. Clarke (1917).

Lugejale võiks tunduda, et lause ei ole keeruline. See-eest leiab osalausestaja, et lauses on 6 osalauset (vt Joonis 19). Esimese osalausena tuvastatakse *Eelmise sajandi kuulsaimad kirjanikud on Charles Dickens, Virginia Woolf, James Joyce, F. Scott Fitzgerald ja Arthur C. Clarke*. Ülejäänud 5 osalauset on osalausestaja arvamusel aastaarvud, vastavalt *(1812), (1882), (1882), (1896), (1917).*



Joonis 19. Näidislause osalausestamise tulemus

Selleks, et vältida osalausestajast tingitud valepositiivseid tulemusi seoses keeruliste lausetega, on lisatud eraldi reegel sulgude vahel oleva teksti kohta. Nimelt, kui sulgude vahel olevas tekstis puudub verbiahel, siis ei võeta seda osalausestamisel arvesse. Seda seetõttu, et tehakse järgnev eeldus: kui sulgude vahel oleval tekstil puudub verbiahel, siis on see täpsustus, lühend või viide. Nagu eelmainitud lause puhul näha, siis täpsustus või

viide ei muuda lauset märgatavalt keerulisemaks. Küll võib aga öelda, et kui sulgude vahel leidub verbiahel, on lause läinud keerulisemaks. Seda seepärast, et lausesse on verbi tõttu tekkinud üks mõte rohkem.

Lisareegli tõttu võetakse eelneva lause puhul lausepikkuse ja -keerulisuse analüüsis arvesse vaid ühte osalauset *Eelmise sajandi kuulsaimad kirjanikud on Charles Dickens, Virginia Woolf, James Joyce, F. Scott Fitzgerald ja Arthur C. Clarke*, milles leidub verbiahel *on*.

Joonis 20 illustreerib lisareegli mõju. Vasakpoolsel pildil on näha, kuidas osalausestaja on aastaarvud märkinud osalauseteks. Parempoolsel joonisel kujutatud andmeid kasutatakse analüüsis otsuse tegemisel, kas lause on liiga pikk või keeruline. Andmetest on näha, et aastaarvud on lisareegli tõttu eemaldatud. Kuna lause ei vasta eelmainitud pika või keerulise lause reeglile, siis analüsaator ei too seda lauset välja.

Joonis 20. Vasakul lause, millel pole rakendatud lisareeglit. Paremal lisareegliga parandatud lause

Sulgude vahel võib olla verb, mis on ühesõnaline täpsustus ja ei mõjuta lause keerukust niivõrd palju. Näiteks lauses *Inimesed liiguvad (kõnnivad) hoone poole* on sulgude vahel üks sõna. Kuna sulgude vahel on verbiahel ning on keeruline eristada täpsustust ja uut lausesse tekkinud mõtet, siis võetakse see siiski analüüsis arvesse.

Üks analüsaatori olulisemaid omadusi on, et see ei võta arvesse jutumärkide vahel olevaid sõnu. Nende puhul tehakse eeldus, et need on kas mõisted või tsitaadid. Tsitaat on autori

tekstist sõltumatu ehk autor seda muuta ei saa, seetõttu ei ole ka vajadust tsitaati keerulise näitena välja tuua. Mõiste- ja tsitaadikontrolli ning eemaldamist teevad klassid `QuoteAnalyzer` ja `QuoteRemover`. Nende kohta on rohkem kirjutatud eelnenud peatükis 5.2.

`QuoteAnalyzer` ja `QuoteRemover` klasside vajalikkust demonstreerib järgnev näitelause:

Ta on kirjutanud: „Praegu on kogu meie ametnikkond ühtse riikliku süsteemi teenistuses ning kirjade ja teiste dokumentide sõnastus minetab järjest isikupära, mistõttu asutustevahelised sidemed on tihedad, sest ühes kantseleis keele kohta langetatud väärotsus jõuab dokumentide vahendusel peagi teistesse, kus keele asjus kriitikavõimetud ametnikud selle omaks võtavad ja seda omakorda levitama hakkavad.“ [16].

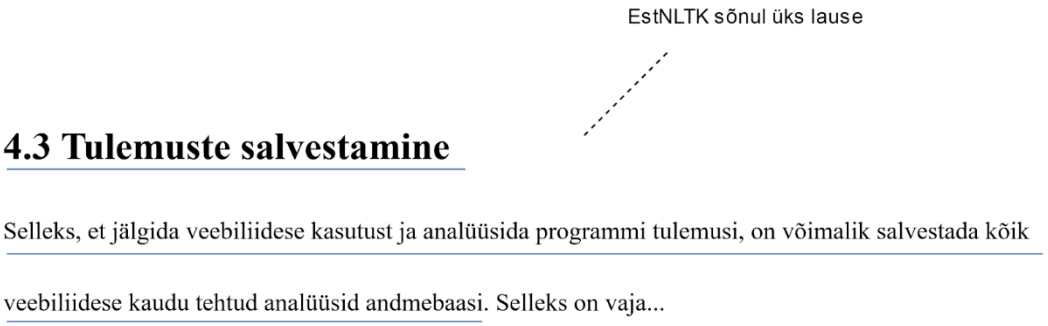
Näitelause on demonstratsiooni jaoks kohandatud Uno Liivaku kirjutis. Analüsaator märgistaks tsitaadi ilma lisakontrollita liiga pikaks. Kuna tekst on tsitaadis, siis mõiste- ja tsitaadituvastaja ning -eemaldaja töö tulemusena võetakse analüüsil arvesse ainult lauseosa *Ta on kirjutanud:*. Kuna see ei vasta liiga pika või keerulise lause tuvastamise reeglitele, ei too analüsaator seda lauset välja.

Katsetamisest selgus, et analüüsi tulemusi mõjutasid ka lausesisesed viited. Seetõttu ei võeta lauset analüüsides viiteid arvesse. EstNLTK `ClauseSegmenter` osalausestaja leiab, et viited on omaette osalaused. Viidete osalauseks märkimine mõjutab aga eelmainitud pika või keerulise lause tuvastamise reegleid. Seetõttu kasutatakse probleemi vältimiseks loodud viite-eemaldaja klassi `CitationRemover`, mis leiab ja eemaldab lausesisesed viited.

Joonis 21. Vasakul lause, kus viide pole eemaldatud. Paremal lause, kus viide on eemaldatud

Joonis 21 demonstreerib lause *See mõiste on Näidis Autori (2015: 10) õpikus välja toodud* puhul `CitationRemover` vajalikkust. Olgugi et seda lauset poleks ka enne `CitationRemover` tööd märgistatud liiga pikaks, on sellegipoolest näha, et viited mõjutavad tugevalt osalausestaja tulemusi.

Lisaks märgati kasutajatega testides, et analüüsi kvaliteeti mõjutasid pealkirjad, alapealkirjad ning jooniste pealkirjad. Kuna nende lõpus puudub lauselõpetaja, arvab EstNLTK lausestaja, et need on osa lausest. Probleemi olemust demonstreerib Joonis 22.



Joonis 22. Pealkirja mõju EstNLTK lausestajale

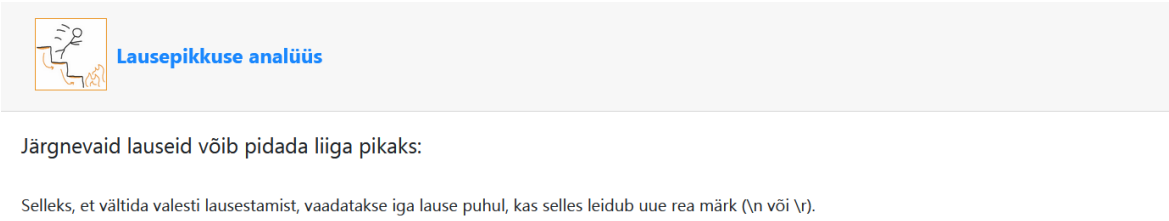
Nagu näitab Joonis 22, arvab EstNLTK lausestaja, et pealkiri on osa lausest. Seda seepärast, et lausestaja otsib lauselõpu jaoks punkti või muud lauselõpetajat. Kuna uue rea märk \n ei ole lauselõpetaja, siis loetakse üheks lauseks nii pealkiri kui ka lõigu esimene lause.

Selleks, et vältida valesti lausestamist, vaadatakse iga lause puhul, kas selles leidub uue rea märk (\n või \r). Kui leidub, eraldatakse lause uue rea märgi pealt kasutades Python funktsiooni `split`. Kui uue rea märgile eelneva teksti lõpus puudub punkt, eeldatakse, et tegu on pealkirjaga. Siis võetakse analüüsil arvesse vaid teksti, mis järgneb uue rea märgile.

Lausepikkuse- ja keerulisuse analüsaator töötab hästi. Katsetamisest ja tagasisidest on selgunud, et analüsaator toob välja laused, mis võivad lugeja jaoks olla liiga pikad või keerulised. Teksti autor peab see-eest ise otsustama, kas analüsaatori välja toodud lause on tõepoolest liiga pikk. Näiteks klassifitseeris analüsaator järgneva lause keeruliseks:

Kuna käesolevas uurimistöös uuritakse verbe, mida on kasutatud otsekõne saatelausetes, ning vähesel määral ka saatelausete paiknemist, siis selles peatükis antakse lühike ülevaade, kuidas üldse tegelaskõnet kirjanduses on võimalik edasi anda ning milline tähtsus on saatelausetel dialoogide puhul. [17].

See lause ei olnud teksti autori arust keeruline. Seetõttu on lausepikkuse ja -keerulisuse analüsaator teksti autorile siiski kõigest abivahend ning kirjutaja peab lõpuks ise otsustama, kas muudab lauset. Tagasiside tulemuste kohta on rohkem räägitud peatükis 6.



Joonis 23 Lausepikkuse analüüsi tulemus veebiliideses

Joonis 23 näitab lausepikkuse analüüsi tulemust veebiliideses. Kasutaja saab vajutada nupule *Lausepikkuse analüüs*, misjärel avaneb lahter, kust näeb pikaks tuvastatud lauseid.

5.7 Kantseliidi analüsaator

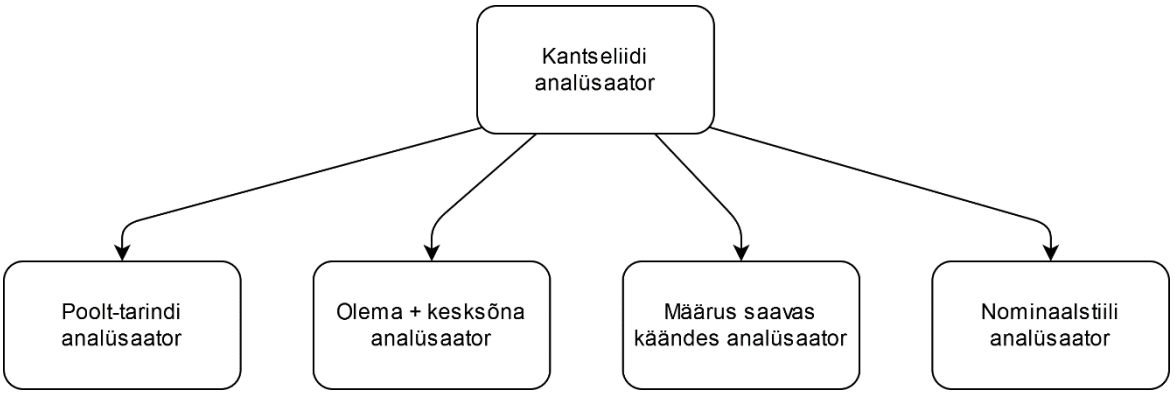
Lõputöö analüsaator kontrollib, kas sisendtekstis leidub kantseliiti. Kantseliit on ebaselge, keerukas, raskesti ja mitmeti mõistetav keelekasutus, mis tekib teiste keelte mallide ja lauseehituse ülevõtmisest eesti keelde[27].

Kantseliidi kontrollid on suuresti üle võetud Kaarel Sõrmuse 2017. aasta bakalaureusetööst „Kantseliidi- ja paronüümituvastaja“ [18]. Sõrmuse lõputöös oli mitmeid analüüse, mis tuvastasid erinevaid kantseliitlikke tekstitunnuseid. Lõputöö analüsaatorisse tõsteti ümber poolt-tarindi, määrus saavas käändes, olema + kesksõna ja nominaalstiili analüüsid.

Katsetamisest selgus, et mõnda analüsaatorit sai ka edasi arendada. Esialgsed analüsaatorid andsid tihti valepositiivseid vastuseid. Valepositiivsete tulemuste vähendamiseks lisati mõnele analüüsaatorile juurde lisareegleid.

[27] Kantseliit https://www.taskutark.ee/m/kantseliit-uks-levinud-stiiliaps/

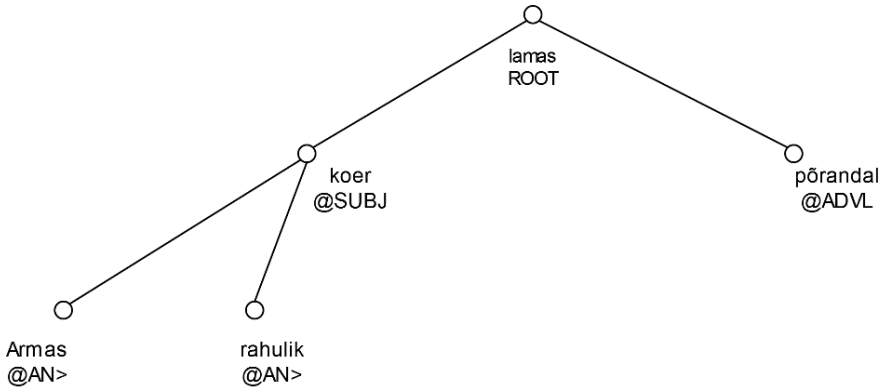
Kuna Sõrmuse lõputöös oli kasutatud EstNLTK versiooni 1.4.1, pidi funktsioone muutma, et nad oleksid kohalduvad versiooniga 1.6. Kõige suuremaks erinevuseks osutusid EstNLTK versioonidevaheliste andmestruktuuride erinevused.



Joonis 24. Kantseliidi analüsaatori jagunemine väiksemateks analüsaatoriteks

Lõputöö analüsaatoris on loodud eraldi moodul `officialese_analyzer`, mis otsib tekstis kantseliitlikke tunnuseid. Moodul kutsub välja eri funktsioone ehk analüsaatoreid, millest igaüks otsib tekstist kindlat sorti kantseliidi tunnust (vt Joonis 24).

Mooduli analüsaatorid vaatavad igat lauset eraldiseisvalt. Esmaselt analüüsitakse lause süntaksit. Selleks kasutatakse VISLCG3[28] süntaksi analüsaatorit, mis on EstNLTK teegist kättesaadav[29]. Sarnaselt lause-, sõna- ja morfoloogilise analüüsikihtidega (vt Joonis 8), lisab süntaksi analüsaator igale sõnale süntaksi analüüsikihi.



Joonis 25. Sõltuvussüntaksi puu

Süntaksi analüüsikiht on oluline seetõttu, et selle kaudu on võimalik saada kätte lause sõltuvussüntaksi puu. Sõltuvussüntaksi puu esitab lause süntaktilist struktuuri, esitades tervet lausestruktuuri kahe sõnavormi vahelise ebasümmeetrilise suhtena (vt Joonis 25)[19].

[28] VislCG3 https://visl.sdu.dk/constraint_grammar.html
[29] EstCG - Tools and resources for Estonian Constraint Grammar. https://github.com/EstSyntax/EstCG

Sõltuvussüntaksi puus on igal sõnal oma *ülemus* (traditsiooniliselt *põhi*) ja *alluv* (traditsiooniliselt *laiend*), mis märgistavad sõnade omavahelist sõltuvussuhet [19]. Joonis 25 näitab, et lause *Armas rahulik koer lamas põrandal* on sõna *lamas* lause *juur*. Juureks nimetatakse sõna, millel puudub sõltuvussüntaksi puus ülemus [19]. Sõna *koer* sõltub sõnast *lamas*, mistõttu *lamas* on sõna *koer* ülemus. Edasi on näha, et sõnad *Armas ja suur* sõltuvad sõnast *koer*, ehk *koer* on nende sõnade ülemus. Kantseliidi analüsaatorites kasutatakse sõltuvussüntaksi puud, et leida sõnade ülemusi.

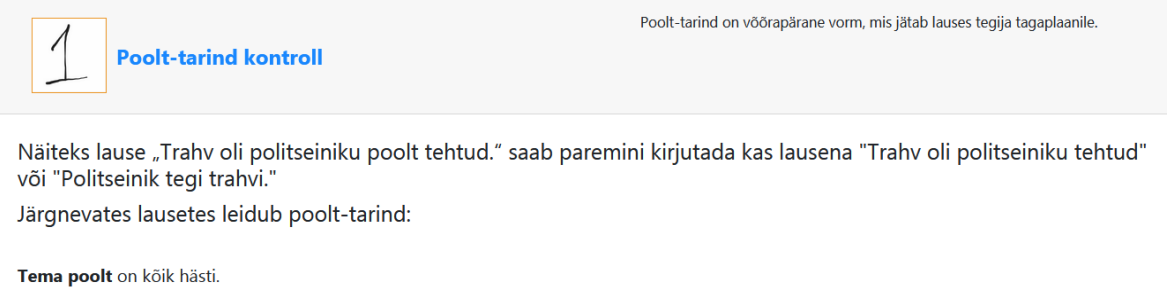
Kõik neli kantseliidi analüsaatorit analüüsivad lauset. Kui lauses leidub kindel kantseliidi tüüp, tuuakse see välja. Analüsaatoritest on juttu järgnevates alapeatükkides.

5.7.1 Poolt-tarind

Poolt-tarind on võõrkeelne vorm, mis jätab lauses tegija tagaplaanile [18]. Tõenäoliselt tekib seda eestikeelsesse lausesse siis, kui autori lauseehitust mõjutab tugevasti inglise keel. Näiteks ingliskeelse lause *The work was done by the employee* puhul oleks eestikeelne otsetõlge *Töö oli töötaja poolt tehtud*.

Selline lause on poolt-tarindi tõttu kantseliitlik [18]. Poolt-tarindit saab vältida, kui kirjutada hoopis *Töö oli töötaja tehtud* või *Töötaja tegi tööd*.

Poolt-tarindi kontrolli loogika on täielikult Sõrmuse tööst ümber tõstetud. Esmaselt käiakse kõik lause sõnad läbi ja uuritakse EstNLTK loodud morfoloogilise analüüsi kihti. Kui vaadeldava sõna sõnajuur on *poolt*, siis vaadatakse sellele eelnenud sõna [18]. Kui eelnev sõna on omastavas käändes, tuvastab analüsaator, et lauses on poolt-tarind [18]. Veebiliideses tuuakse poolt-tarindiga lause esile (vt Joonis 26).



Joonis 26. Poolt-tarindi analüüsi tulemus veebiliideses

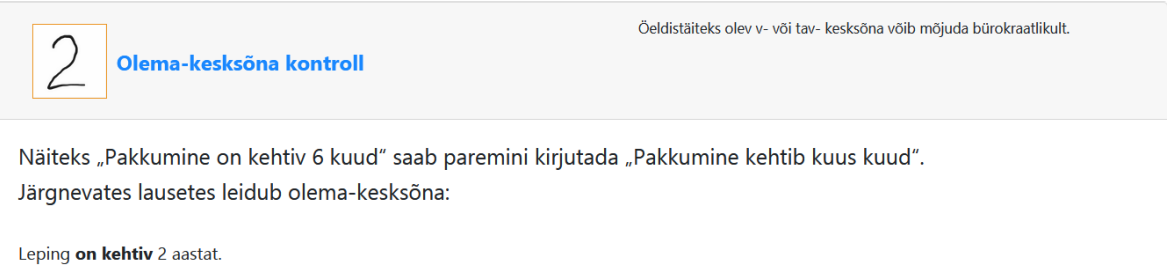
Joonis 26 näitab, et sõna *Tema* on omastavas käändes. Sellele järgneb sõna, mille sõnajuur on *poolt*. Seetõttu märgitakse ära, et lauses leidub poolt-tarind.

Isiklikust katsetamisest ja kasutajatega testimisest selgus, et poolt-tarindi kontrolli tulemustes leidus kõige vähem valepositiivseid kirjeid. Katsetatud lausetes, kus analüsaator oli välja toonud poolt-tarindi, see ka tõepoolest lauses sisaldus. Kantseliiti sai vältida sellega, et muudeti lause sõnastust või eemaldati sõna *poolt*.

5.7.2 Olema + kesksõna

Öedistäide, mis on *v*- või *tav*-kesksõna, võib mõjuda kohmakalt [20]. Oleviku kesksõna väljendab tegevust, mis iseloomustab lauses tegijat või tegijaobjekti [21]. Näiteks lauses *Leping on kehtiv 2 aastat* on oleviku kesksõna *kehtiv*. Seda lauset saab see-eest paremini kirjutada hoopiski *Leping kehtib 2 aastat*.

Olema + kesksõna analüsaator on võetud Sõrmuse lõputööst, ent sellele on lisatud mõned reeglid. Esmaselt vaadatakse üle kõik lause sõnad. Nagu Sõrmuse töös, otsitakse sõna, mis on öeldistäide ja mille lõpus on *tav* või *v*. Kui selline sõna on leitud, uuritakse selle sõna süntaktilist ülemust. Kui ülemuse algvorm on *olema*, siis on tuvastatud olema + kesksõna kantseliit. Kantseliitlik tekst tuuakse veebiliideses esile (vt Joonis 27).



Joonis 27. Olema + kesksõna kontrolli tulemus veebiliideses

Katsetamisest selgus, et analüsaator toob tihti esile valepositiivseid tulemusi. Analüsaator toob valesti välja omadussõnad, mille lõpus on *v* või *tav*. Näiteks toodi kantseliidina välja sõnapaar *on huvitav*. Kuna sõna *huvitav* ei ole kesksõna, vaid omadussõna, ei tohiks analüsaator seda kantseliidina välja tuua.

Katsetati, kas on võimalik eristada kesksõnu ja omadussõnu nende liigi järgi. Näiteks, kas on võimalik, et sõnapaaris *on kehtiv* märgitakse sõna *kehtiv* verbiks, ent sõnapaaris *on huvitav* sõna *huvitav* omadussõnaks.

Katsetamisest selgus, et kesksõnu ja omadussõnu ei olnud võimalik eristada. Ei süntaksi ega morfoloogilise analüüsi sõnaliigi märgendajad ei suutnud tuvastada, kas sõna on kesksõna,

mis väljendab tegevust (näiteks *kehtiv*, *lööv*) või on sõna tavaline omadussõna (näiteks *huvitav*, *põnev*).

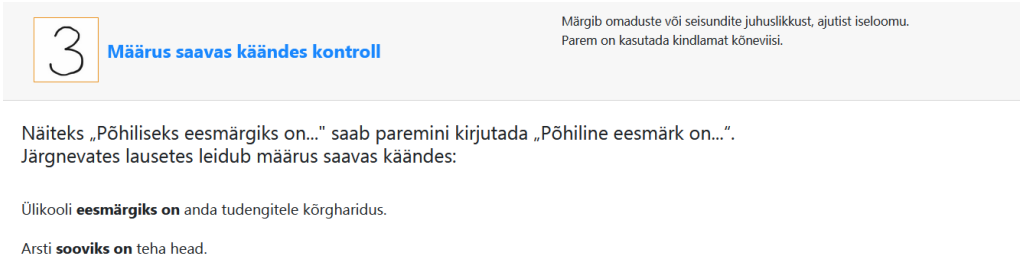
Märgendajad märkisid kõik kesksõnad omadussõnadeks. See-eest, kui märgendajad suudaks kesksõna ja omadussõna eristada, oleks lihtne kantseliidi tuvastamisel võtta arvesse ainult sõnu, mis on kesksõnad. Niiviisi väheneks valepositiivsete tulemuste arv märgatavalt.

Mõningate valepositiivsete tulemuste vältimiseks loodi erandid. Kui öeldistäide on kas *huvitav* või *vaieldav*, siis seda kantseliidina arvesse ei võeta. Selleks, et valepositiivsete arv väheneks veelgi, on vaja veel erandeid leida ja juurde lisada. Erandite leidmiseks saab kasutada ka omadussõnade korpust.

5.7.3 Määrus saavas käändes

Määrus saavas käändes vihjab seisundite juhuslikkusele või ajutisele olekule, ent mõnikord on parem kasutada kindlamat kõneviisi [18]. Näiteks lauset *Arsti sooviks on teha head* oleks parem kirjutada *Arst soovib teha head*.

Ka see analüsaator on võetud üle Sõrmuse lõputööst, ent sellele on tehtud mõned edasiarendused. Esmaselt vaadeldakse kõik lause sõnad läbi. Nagu Sõrmuse töös, otsitakse esmaselt sõna, mille süntaksimärgend on määrus ning kääne on saav. Kui selline sõna on leitud, võetakse vaatluse alla selle sõna süntaktiline ülemus [18]. Kui ülemuse algvorm on *olema*, on tuvastatud määrus saavas käändes [18]. Kantseliit tuuakse veebiliideses esile (vt Joonis 28).



Joonis 28. Määrus saavas käändes kontrolli tulemus veebiliideses

Katsetamisest selgus, et analüsaatori tulemustes esines palju valepositiivseid kirjeid. Lause *Arsti sooviks on teha head* on tõepoolest kantseliitlik, ent analüsaator tõi välja ka sõnapaare, kus probleemi tegelikult ei esinenud. Näiteks pidas analüsaator kantseliitlikuks järgmised sõnapaarid: *näiteks on*; *lisaks on*; *selleks on*; *selliseks on* ja *tulemuseks on*. Seetõttu tehti edasiarendus: kui sõna vastab eelmises lõigus mainitud reeglitele, ent sõna on kas *lisaks*, *näiteks*, *selleks*, *selliseks* või *tulemuseks*, siis seda kantseliidiks ei märgita.

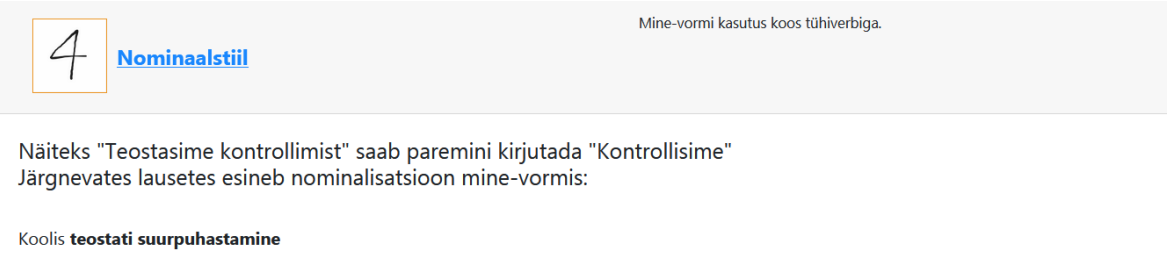
Kuigi määrus saavas käändes analüsaator võib tuvastada palju valepositiivseid, on sellel potentsiaali olla kasulik. Kasutajatega testimisest selgus, et analüsaator leidis ka õigeid ja problemaatilisi tekstiosi, mida sai tõesti parandada.

5.7.4 Nominaalstiil

Nominaalstiiliks ehk tühiverbiga nominalisatsiooniks kutsutakse liialdamist nimisõnadega, eriti verbidest loodud mine-tuletistega[30]. Nominaalstiili puhul kasutatakse lausetes tühiverbe ehk tegusõnu, mis on kaotanud oma esialgse tähenduse[30]. Näiteks on tühiverbid *teostama, toimuma* ja *sooritama*.

Kuna alusele vastav täiend ja sihitisele vastav täiend on nominalisatsiooni puhul samas vormis, võib lugejale tunduda lause mitmeti mõistetav [22]. Lause *Kati pesemine võttis kaua aega* puhul ei ole kindel, kas kaua aega võttis Kati pesuskäik või hoopis see, et Kati pesi pesu.

Nominaalstiili analüüs on ümber tõstetud Sõrmuse lõputööst. Vaadatakse kõiki sisendteksti lauseid. Nagu Sõrmuse tööski, kui tuvastatakse sõna, mille lemma on *mine*-lõpuline (nt *puhastamine*), vaadatakse selle sõna ülemust. Kui sõna ülemus kuulub tühiverbide hulka (*kuuluma, teostama, toimuma, vajama*), tuvastatakse nominaalstiil. Nominaalstiil tuuakse veebiliideses esile (vt Joonis 29).



Joonis 29. Nominaalstiili kontrolli tulemus veebiliideses

Sõrmuse nominaalstiili kontrollis oli veel lisareegleid, mida lõputöö teksti analüsaatorisse ümber ei tõstetud. Näiteks tuvastab Sõrmuse funktsioon ka sihitisega nominaalstiili (*teostati suurpuhastust*).

Katsetamisest selgus, et nominaalstiili analüsaator niivõrd palju valepositiivseid tulemusi ei andnud. Analüsaatori esile toodud lausetes leidus tõepoolest nominaalstiili tunnuseid.

[30] Nominaalstiil https://kirjandiabi.wordpress.com/2012/11/27/stiilivead-nominaalstiil-ehk-nominalisatsioon-ehk-nimisonatobi/

5.8 Puuduvate komade analüsaator

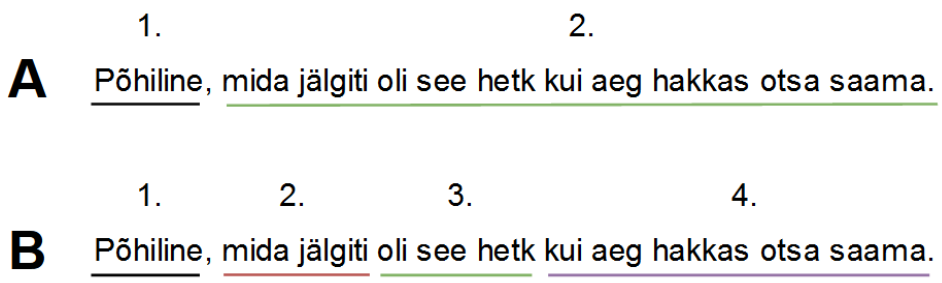
Programmi luues ja isiklikult katsetades märgati, et erinevates lõputöödes, nii arvutiteaduse instituudi kui teiste instituutide tudengite lõputöödes, leidus tihti komavigu. Seetõttu otsustati luua analüsaator, mis leiab puuduvate komadega laused ja annab seejärel soovituse, kuhu vajalikud komad panna. Analüsaator asub moodulis `missing_commas_analyzer`.

Analüsaatori puhul otsustati katsetada, kas on võimalik kasutada EstNLTK `ClauseSegmenter` osalausestajat ja selle erinevaid režiime selleks, et leida sisendtekstis puuduvad komad.

EstNLTK `ClauseSegmenter` klassi on võimalik kasutada kahes põhilises režiimis. Esimene on harilik režiim (*basic mode*), mis eeldab, et tekst on korrektse komakasutusega kirjutatud. Harilik režiim ei võta arvesse puuduvaid komakohti ja töötab korrektse teksti puhul võrdlemisi hästi.

Teine režiim (`ignore_missing_commas` *mode*) ignoreerib puuduvaid komasid. Eesti keeles on komad olulised osalausete eraldajad. Osalausestaja harilikus režiimis esineb tihti vigu seepärast, et sisendtekstis on komavead [23]. Seetõttu on võimalik kasutada `ClauseSegmenter` klassi režiimis, mis arvestaks puuduvate komadega sisendtekstis. Režiimi kasutamiseks on vaja `ClauseSegmenter` klassi isendit luues anda sisend-parameetriks `ignore_missing_commas=True`.

Analüsaatori töö algatamisel luuakse esmaselt kaks `ClauseSegmenter` klassi isendit. Esimene osalausestaja (edaspidi viidatud kui osalausestaja A) töötab harilikus režiimis, ent teine osalausestaja (edaspidi viidatud kui osalausestaja B) suudab arvestada võimalusega, et tekstis on puuduvaid komasid. Kahe osalausestaja osalausestamise tulemuste erinevust demonstreerib Joonis 30.

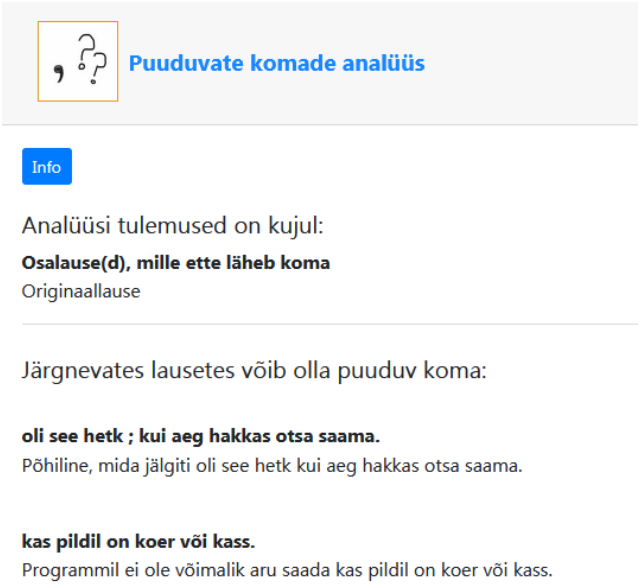


Joonis 30. Kahe osalausestaja osalausestamise tulemuste võrdlus

Iga sisendteksti lause puhul osalausestavad mõlemad osalausestajad vaadeldava lause. Seejärel võrreldakse tulemusi. Kui osalausete arv on nii osalausestajal A kui osalausestajal B tulemustes võrdne, siis võetakse vaatluse alla järgmine lause. Lauses on analüsaatori reeglite järgi puuduv koma siis, kui osalausestaja B osalausete arv on suurem kui osalausestaja A osalausete arv.

Kui lauses on tuvastatud puuduv koma, siis antakse soovitus, kuhu koma lisada. Tehakse eeldus, et kui osalausestaja A osalausetes leidub osalause, mis on osalausestaja B osalausetes poolitatud vähemalt kaheks erinevaks osalauseks, siis peab B poolitatud osalausete vahele lisama puuduvad komad või koma.

Näiteks Joonis 30 näitab, et osalause *mida jälgiti oli see hetk kui aeg hakkas otsa saama* on osalausestaja B poolitanud kolmeks erinevaks osalauseks. Seetõttu tehakse siinkohal korrektne eeldus, et B leitud osalausete vahele on vaja komasid. Leitakse üles B poolitatud osalausete algusindeksid ehk kohad, kuhu on vaja lisada komad. Veebiliideses näidatakse seejärel tulemusena lauset ning osalauseid, mille ette peaks lisama komad (vt Joonis 31).



Joonis 31. Puuduvate komade analüüsi tulemus veebiliideses

Puuduvate komade analüsaatori efektiivsus ja kasulikkus sõltub suuresti sisendteksti korrektsusest. Nagu oli EstNLTK dokumentatsioonis mainitud, töötab osalausestaja B paremini tekstidel, kus esineb rohkem komavigu [23]. Mida rohkem on sisendtekstis komavigu, seda vähem on valepositiivseid tulemusi. Näiteks leidis puuduvate komade analüsaator ühe 6000-sõnalise sisendteksti puhul 9 komaveaga lauset, millest 8 olid tõepoolest vigased ja üks oli valepositiivne tulemus.

Korrektsemate lausete puhul on alati võimalus, et osalausestaja B osalausestab lauset valesti. Näiteks korrektse lause *Kuna metoodikas olid teatavad erinevused, ei ole nad täielikult võrreldavad* puhul leiab osalausestaja B, et *teatavad erinevused* on eraldi osalause. Seetõttu toob analüsaator välja, et *teatavad erinevused* ette käib koma.

Kokkuvõtteks võib öelda, et `ClauseSegmenter` klasside kasutamine puuduvate komade leidmiseks on igati sobiv valik. Osalausestajaid on võimalik edukalt kasutada selleks, et tuvastada sisendtekstis puuduvad komakohad.

6. Kasutajate tagasiside

Programmi testiti 5 inimesega. Testijad andsid tagasisidet rakenduse kolme iteratsiooni kohta. Iga iteratsiooni vahel oli 2 nädalat, mille jooksul tehti rakendusele vastavalt tagasisidele edasiarendusi. Seejärel paluti järgmises iteratsioonis hinnata rakendust uuesti. Niiviisi prooviti määrata, kas rakendusele tehtud edasiarendused on rakendust kasulikumaks ja paremaks teinud.

Esialgu sooviti võtta testijateks vaid inimesi, kes olid õpikeskkonda CGLearn registreeritud. Eesmärk oli vaadata, kas iganädalased raportid läksid tudengite jaoks kasulikumaks. Lõputöö kirjutamise aastal oli CGLearn keskkonnas ainult 2 eestikeelse lõputöö kirjutajat, kes oleks saanud rakendust testida. Seetõttu otsustati otsida testijaid ka mujalt. Leiti juurde 3 inimest, kes kasutasid lõputöö tekstide analüsaatori testimiseks loodud veebiliidest. Valdavalt olid tudengid arvutiteaduse instituudist.

CGLearn keskkonda registreeritud tudengid said oma analüüsi näha CGLearn õpikeskkonnas. Teised testijad said analüüside tulemusi näha veebiliideses.

Otsustati, et 5 on iteratiivse testimise jaoks piisav arv testijaid. Seda seepärast, et arvutiteadlane Jakob Nielsen on väitnud, et suurema testijate arvu puhul on suurem tõenäosus, et tagasiside kordub ja ei anna efektiivset lisandväärtust [24].

6.1 Küsitlus

Küsitlusi tehti kokku kolm ning igas küsitluses olid küsimused jagatud mitmeks sektsiooniks. Igas sektsioonis olid küsimused ühe spetsiifilise analüsaatori kohta. Sektsiooni lõpus paluti hinnata analüsaatori kasulikkust skaalal 0-10, kus 0 tähendas, et analüsaator ei olnud üldse kasulik ning 10 tähendas, et analüsaator oli väga kasulik.

Umbisikulisuse analüsaatori sektsioonis paluti vastajatel märkida tõeseks väited, mis kehtisid analüüsi tulemuste kohta.

Lausepikkuse ja -keerulisuse analüsaatori puhul paluti vastata, mitu lauset see esile tõi. Seejärel küsiti, mitut lauset analüüside tulemusena tegelikkuses muudeti. Niiviisi sai määrata, kuivõrd palju võeti analüsaatori tulemusi oma lõputöö kirjutamisel arvesse. Kui muudeti mingit lauset, paluti kirjutada, mis lauset muudeti ja kuidas. Lisaks paluti kommenteerida analüsaatori rangust ehk seda, kas välja toodud laused olid liiga pikad või olid need tegelikult sobivad.

Korduvate sõnade analüsaatori puhul paluti märkida väited, mis kehtisid analüsaatori tulemuste kohta. Paluti kommenteerida, kas analüsaator tõi välja sobival arvul sõnu, kuivõrd palju sai teha muudatusi, kuivõrd palju tegelikult muudatusi tehti. Sõnaklastrite puhul küsiti nende arvukuse ja tehtud muudatuste kohta.

Kantseliidi analüsaatori juures paluti hinnata igat spetsiifilist kantseliidi analüsaatorit eraldi. Küsiti, mitu tulemust iga analüsaator välja tõi. Seejärel paluti hinnata iga kantseliidi analüsaatori kasulikkust skaalal 0-10.

Puuduvate komade analüsaatori puhul paluti vastata, mitu puuduva komaga lauset tõi analüsaator välja. Seejärel küsiti, mitmesse lausesse lisati tulemuste põhjal puuduv koma. Niiviisi sai tuvastada, mitu tuvastatud puuduva komaga lauset olid tegelikult valepositiivsed.

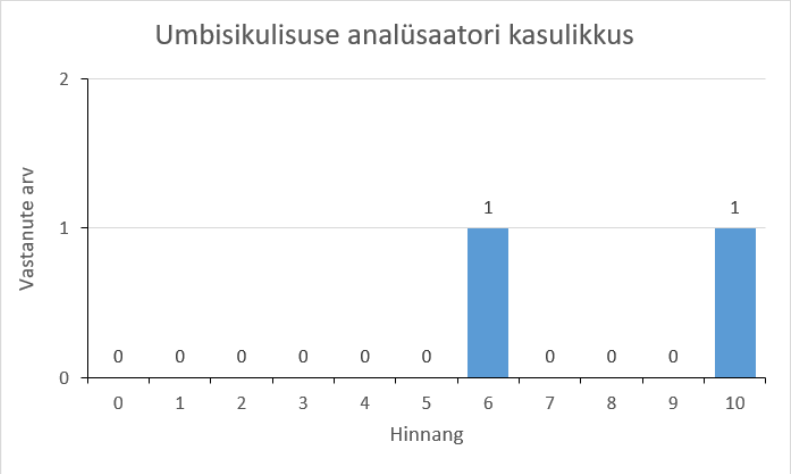
Kuna programm logis andmebaasi kõiki algatatud analüüse, sai soovi korral analüüsida testijate tulemusi. Niiviisi sai lihtsasti analüüsida, milliseid tulemusi lõputöö teksti analüsaator välja tõi. Analüüside tulemuste failid on lisades (vt Lisa III).

Lõpuks sai iga küsitluse lõpus testija lisada oma kommentaare või avaldada oma arvamust. Tulemustest on juttu järgnevates alapeatükkides.

6.2 Esimene iteratsioon

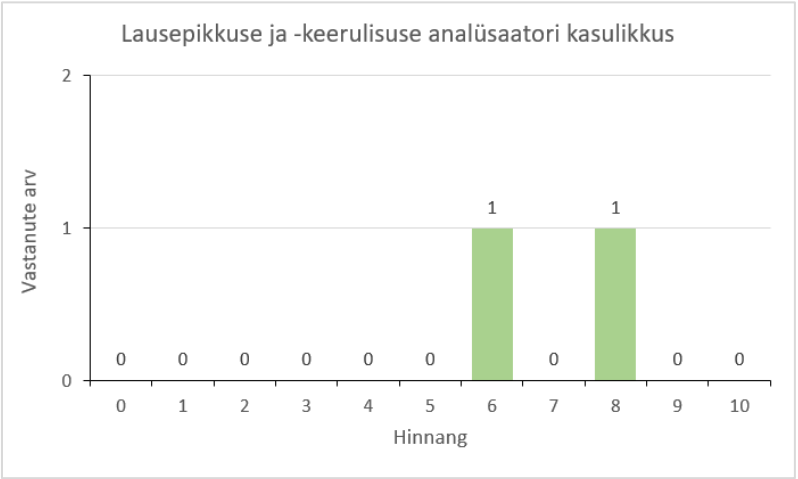
Esimesele iteratsioonile andis tagasisidet kaks tudengit, kes olid mõlemad CGLearn veebikeskkonda registreeritud. Ülejäänud kolm tudengit lisandusid järgnevas iteratsioonis. Sel hetkel oli lõputöö teksti analüsaatori programmis vaid kolm analüsaatorit: umbisikulisuse, lausepikkuse ja -keerulisuse ning korduvate sõnade analüsaator.

Umbisikulisuse analüsaatori puhul vastasid mõlemad tudengid, et analüüs tuvastas isikulised sõnad, mille nad muutsid ümber umbisikulisteks. Analüsaatori kasulikkust hinnati hinnetega 6 ja 10 (vt Joonis 32). Saab järeldada, et analüsaator oli tudengitele kasulik, kuna mõlemad tegid oma lõputöö tekstides muudatusi.



Joonis 32. Umbisikulisuse analüsaatori hinnangud 1. iteratsioonis

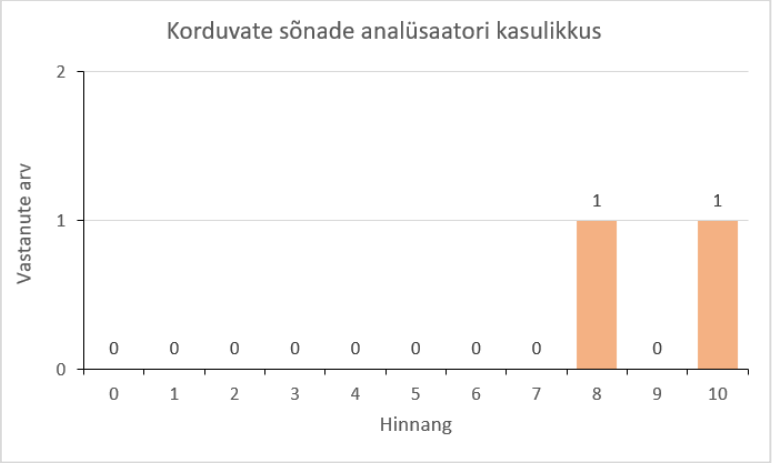
Lausepikkuse ja -keerulisuse analüsaatori kohta vastasid mõlemad testijad, et analüsaator ei tuvastanud ühtegi pikka lauset. Sellegipoolest hinnati analüsaatorit hinnetega 6 ja 8 (vt Joonis 33). Sellest võib järeldada, et hoolimata sellest, et analüsaator ei tuvastanud pikki lauseid, oli sellest siiski kasu. Analüsaatori tulemus andis kasutajatele kindlust, et nende tekstis ei leidu liiga pikki lauseid.



Joonis 33. Lausepikkuse ja -keerulisuse analüsaatori hinnangud 1. iteratsioonis

Korduvate sõnade analüsaatori puhul leidis üks vastaja, et analüsaator tõi välja liiga palju korduvaid sõnu. Teine leidis see-eest, et analüsaator tõi välja sobival arvul sõnu. Lisaks leidsid mõlemad, et kõiki sõnu, mida analüsaator välja tõi, ei saanud asendada teiste sõnadega. Korduvate sõnade analüsaatorile anti hindeks 8 ja 10 (vt Joonis 34). Selle põhjal võiks järeldada, et kasutajatele oli sellest analüsaatorist kõige rohkem kasu.

Kommentaarina lisati, et oleks hea, kui korduvate sõnade analüsaator pakuks sõnadele sünonüüme. See ei olnud aga lõputöö kirjutamise ajal EstNLTK versioonis 1.6 veel võimalik.

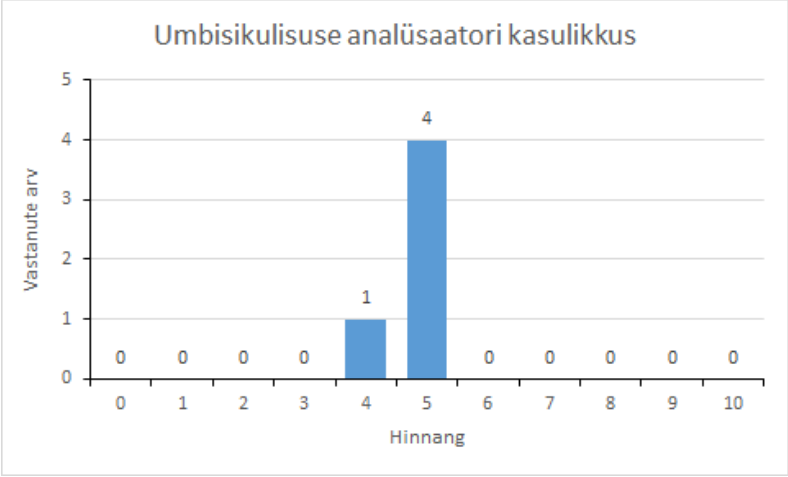


Joonis 34. Korduvate sõnade analüsaatori hinnangud 1. iteratsioonis

6.3 Teine iteratsioon

Teiseks iteratsiooniks lisati töösse kantseliidi analüsaator. Lisandus kolm tudengit, kes polnud CGLearn õpikeskkonda registreeritud. Nemad kasutasid testimiseks loodud veebiliidest. Testijaid oli kokku viis.

Umbisikulisuse analüsaatori puhul oli 3 testijat, kellel ei tuvastatud tekstis mina- või meie-vormis sõna. Ühel inimesel tuvastati mina- või meie-vormis sõna, aga see oli kaldkirjas mõiste. Umbisikulisuse analüsaatorit hinnati võrdlemisi keskmiselt, neli vastanut andis hindeks 5 ning üks andis hindeks 4 (vt Joonis 35). Tudeng, kes andis hindeks

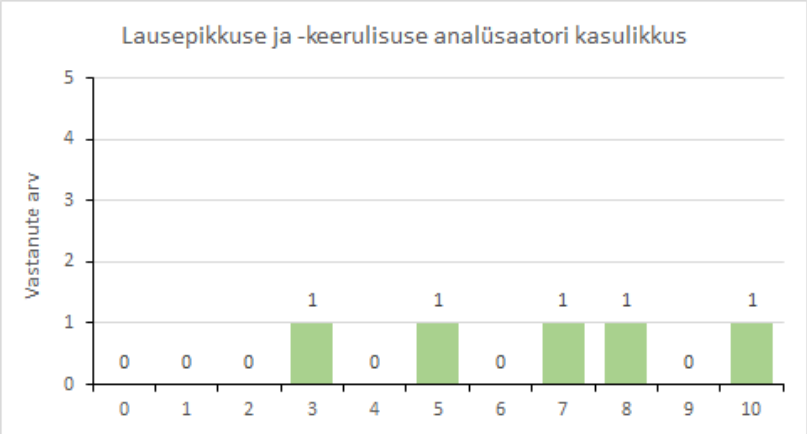


Joonis 35. Umbisikulisuse analüsaatori hinnangud 2. iteratsioonis

4, ütles, et tema instituudis on mina- või meie-vormi kasutamine lubatud, mistõttu see analüsaator tema jaoks nii kasulik ei olnud.

Kahel inimesel ei tuvastanud lausepikkuse ja -keerulisuse analüsaator mitte ühtegi liiga pikka lauset. Ühel tuvastati 1-3 lauset ning kahel 7-9 lauset. Kaks vastanut muutsid analüsaatori tulemusena oma lauseid. Viiest vastanust leidsid kolm, et analüsaator töötab hästi. Esile tõstetud laused olid nende meelest tõepoolest liiga pikad.

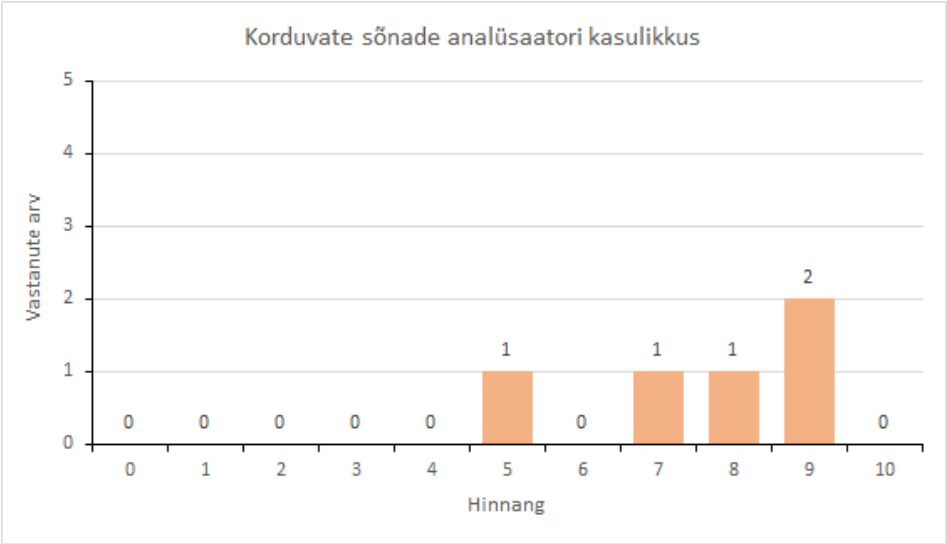
Neljas vastanu tõi lausepikkuse ja -keerulisuse analüsaatori kriitikana välja selle, et lausestaja pidas lõigu pealkirja ja esimest lauset üheks lauseks, mistõttu analüsaator andis valepositiivseid tulemusi. Lausestamise viga sai alles järgmiseks iteratsiooniks parandatud. Viies vastanu oli erapooletu. Kuna enamik vastanutest oli analüsaatori tulemustega rahul, jäeti see järgmiseks iteratsiooniks samaks. Kokkuvõttes hinnati lausepikkuse ja -keerulisuse analüsaatorit positiivselt. Vastanute hinnanguid näitab Joonis 36.



Joonis 36. Lausepikkuse- ja keerulisuse analüsaatori hinnangud 2. iteratsioonis

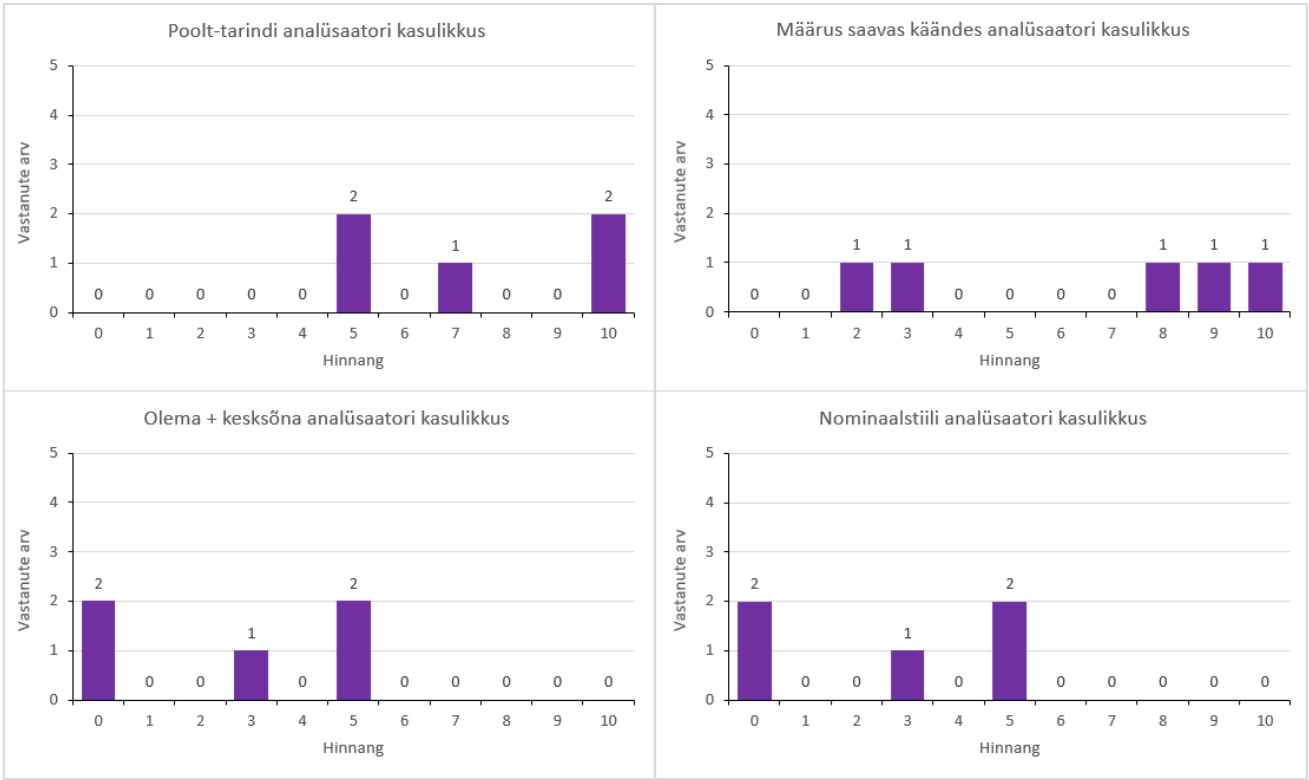
Korduvate sõnade analüsaatori puhul leidis kolm vastanut, et analüüs tõi välja sobival arvul korduvaid sõnu. Kaks vastanut arvas see-eest, et analüsaator on liiga palju korduvaid sõnu välja toonud. Kõik vastanud ütlesid, et nad ei saanud kõiki või mõnda sõnu asendada, kuna need olid spetsiifilised mõisted, millele ei leidunud sünonüümi. Näiteks toodi välja, et kui kirjutatakse automaatkontrollide teemast, siis on raske leida sünonüüme sõnadele *automaatne*, *kursus* ning *tagasiside*. Üks vastanu tegi oma tekstis muudatusi, asendades korduvad sõnad sünonüümidega.

Korduvate sõnade analüsaatori hinnanguid näitab Joonis 37. Hinnangutest võib järeldada, et korduvate sõnade analüsaator oli vastanute jaoks üks kasulikumaid analüsaatoreid.



Joonis 37. Korduvate sõnade analüsaatori hinnangud 2. iteratsioonis

Teises iteratsioonis lisati edasiarendusena kantseliidi analüsaator. Testijatel paluti igat kantseliiditüübi analüsaatorit eraldi hinnata 10-palli skaalal. Joonis 38 näitab kõikide kantseliidi analüsaatorite hinnangud.



Joonis 38. Kantseliidi analüsaatorite hinnangud 2. iteratsioonis

Tagasisidest selgus, et testijate jaoks oli kantseliidi analüsaatoritest kõige kasulikum pooltarindi analüsaator. Isiklikest katsetustest tundus samuti, et see analüsaator andis kõige vähem valepositiivseid tulemusi.

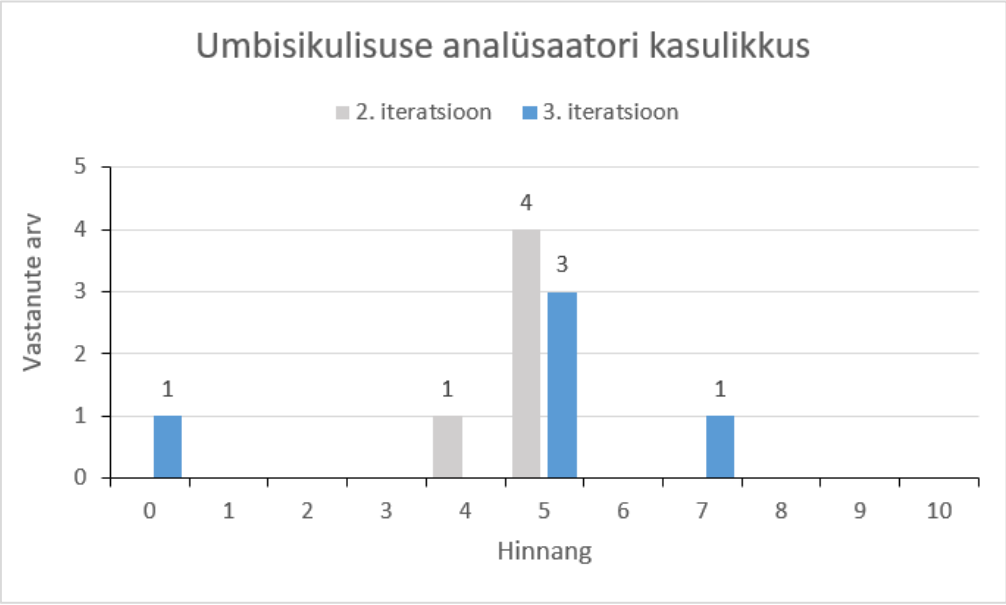
Määrus saavas käändes analüsaatorit hinnati erinevalt. Kolme vastanu jaoks oli analüsaatorist palju kasu, ent kahe jaoks peaaegu üldse mitte.

Olema + kesksõna ja nominaalstiili hindasid kõik vastanud võrreldes teiste kantseliidi analüsaatoritega madalamalt. Vastanud panid hindeks 0, kuna nende jaoks ei toonud analüsaatorid midagi välja. Olema + kesksõna analüsaatori hinnangud on aga vastavuses varasemate isiklike katsetustega, kus leidus palju valepositiivseid tulemusi.

6.4 Kolmas iteratsioon

Kolmandas iteratsioonis tehti mitmeid edasiarendusi. Korduvate sõnade analüsaatorile lisati reegel, et sõna ei võeta arvesse, kui see on jutumärkide vahel. Vahetati välja korduvate sõnade analüsaatori lemmade sageduste andmestik. Lausepikkuse ja -keerulisuse analüsaatoris tehti parandus, et joonise või peatüki pealkiri ei oleks kokku liidetud lõigu või peatüki esimese lausega. Lisati puuduvate komade analüsaator.

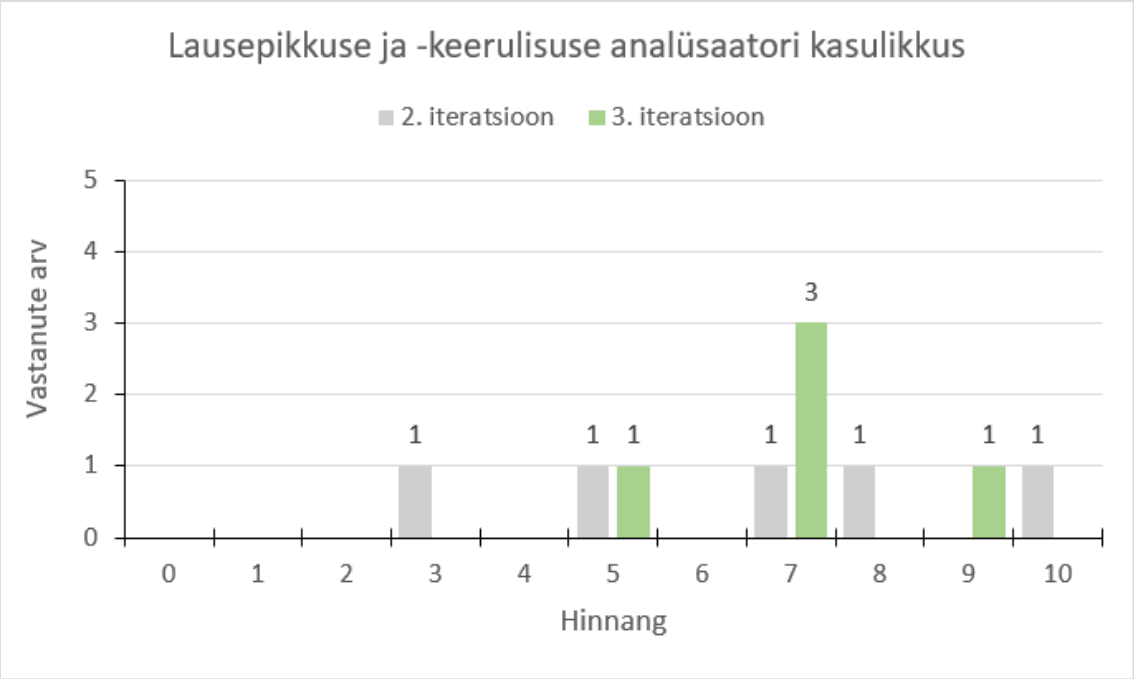
Umbisikulisuse analüsaator ei tuvastanud kolmel testijal mitte ühtegi mina- või meie-vormis sõna. Ülejäänud kaks, kellel tuvastati, ei muutnud oma teksti, kuna isikulised sõnad olid kas kommentaarid iseendale või tegelikult sobivad. Analüsaatori hinnanguid näitab Joonis 39.



Joonis 39. Umbisikulisuse analüsaatori hinnangute võrdlus

Võrreldes eelmise iteratsiooniga oli umbisikulisuse analüsaatorit hinnatud sarnaselt. Tudeng, kes oli 2. iteratsioonis hinnanud analüsaatorit hindega 4, oli seekord pannud hindeks 0. Ta oli kirjutanud, et analüüs tuvastas küll isikulised sõnad, aga tema instituudis on mina- või meie-vormi kasutamine lubatud. Kuna analüsaatorist polnud kasu, oli pandud madalam hinne.

Lausepikkuse ja -keerulisuse analüsaatorit hinnati seekord natuke kõrgemalt kui teises iteratsioonis. Ühe testija tekstist ei tuvastatud ühtegi pikka lauset, kolme testija tekstidest tuvastati igaühel 1-3 pikka lauset. Viimase testija tekstist tuvastati 4-6 lauset. Muudatusi tegi siiski ainult üks vastanu, kes ütles, et muutis kaks pikka lauset eraldi lauseteks. Kõik viis testijat leidsid, et analüsaatori välja toodud laused olid tõepoolest pikad või keerulised. Joonis 40 näitab lausepikkuse ja -keerulisuse analüsaatori kasulikkuse hinnanguid.

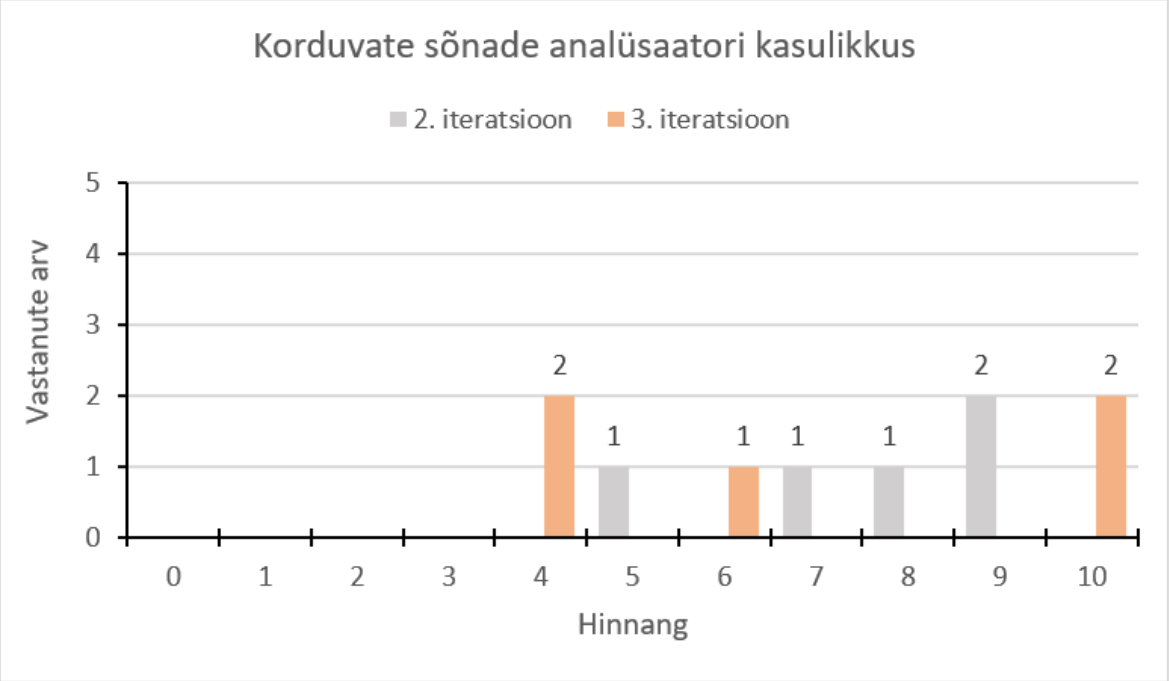


Joonis 40. Lausepikkuse ja -keerulisuse analüsaatori hinnangute võrdlus

Tudeng, kes oli eelmises iteratsioonis hinnanud analüsaatorit hindega 3, pani seekord hindeks 7. Ta mainis, et kui ta 2. iteratsioonis rakendust katsetas, oli tal tekst juba juhendajaga läbi vaadatud ja parandatud. Kui ta 3. iteratsioonis rakendust uuesti proovis, ei olnud teksti varasemalt ära parandatud, mistõttu analüsaator oli kasulikum. Lisaks mainis ta 2. iteratsioonis, et siis olid välja toodud peatükkide esimesed laused, millele olid liidetud peatükkide nimed. Kuna 3. iteratsiooniks sai see viga parandatud, oli valepositiivsete arv tema jaoks välja toodud lausete hulgas väiksem.

Teine tudeng oli lausepikkuse- ja keerulisuse analüsaatori puhul muutnud oma hinnet 8 pealt 5 peale. Eelnevas iteratsioonis oli analüsaator tema jaoks toonud välja pika lause, mille ta oli 3. iteratsiooniks ära parandanud. Kuna seekord ei tuvastanud analüsaator ühtegi pikka lauset, oli see tema jaoks vähem kasulikum.

Korduvate sõnade analüsaatori puhul leidsid 3 vastanut, et analüüs tõi välja sobival arvul sõnu. Üks leidis, et toodi välja liiga palju sõnu ning viimase testija tekstis ei toodud välja ühtegi liigselt korduvat sõna. Neli vastanut leidsid, et kõiki sõnu ei olnud võimalik asendada sünonüümidega. Kolm inimest leidis, et sõnaklastreid toodi välja sobival arvul ning ainult üks tegi klastrite põhjal muudatusi.



Joonis 41. Korduvate sõnade analüsaatori hinnangute võrdlus

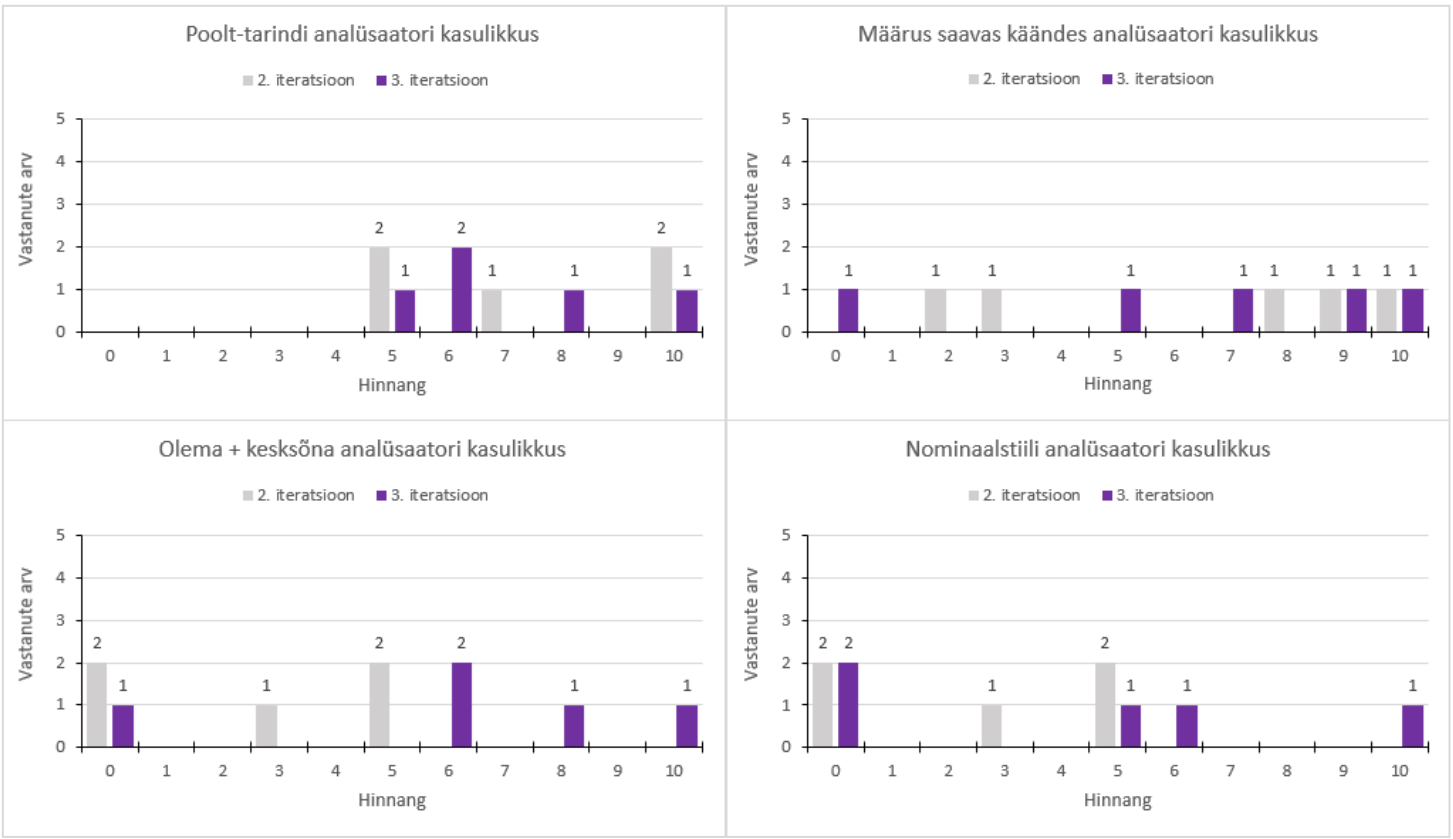
Nagu näitab Joonis 41, on võrreldes eelmise iteratsiooniga korduvate sõnade analüsaatori kasulikkuse hinnangud üpriski sarnased.

Mitmed testijad hindasid korduvate sõnade analüsaatorit madalamalt kui eelmisel korral. Tudeng, kes pani eelmises iteratsioonis hindeks 9, pani seekord hinde 6. Madalama hinde põhjus oli, et 2. iteratsioonis oli ta oma tekstis sõnaklastrite põhjal mitmeid muudatusi teinud. Ka seekord tegi tudeng analüsaatori tulemuste põhjal muudatusi, ent kuna eelmise iteratsiooni tulemuste põhjal said suuremad sõnakordused parandatud, oli analüsaator vähem kasulikum.

Teine tudeng, kes oli oma hinnet muutnud 7 pealt 4 peale, mainis seekord, et tema arust oleks analüsaator pidanud rohkem sõnakordusi välja tooma. Ta ütles, et teda ennast häiris, et tema tekstis oli liiga palju kasutatud sõnu *väga* ja *enam*, ent analüsaator ei olnud neid liigselt korduvatena välja toonud.

See-eest mõned tudengid tõstsid oma hinnet. Üks tudeng tõstis hinde 8 pealt 10 peale ja teine 9 pealt 10 peale. Tulemuste pealt on raske hinnata, kuivõrd tugevalt ja hästi mõjutas analüsaatori kasulikkust uus teaduskirjanduse lemmade korpus.

Kuigi kantseliidi analüsaatoreid 3. iteratsiooniks ei muudetud, said need seekord siiski positiivsemat vastukaja. Hinnanguid näitab Joonis 42.



Joonis 42. Kantseliidi analüsaatorite hinnangute võrdlused

Poolt-tarindi analüsaatorit hinnati seekord 2. iteratsiooniga sarnaselt. Peaaegu kõikide testijate hinnangud olid samad või erinesid vaid mõne punkti võrra.

Olema + kesksõna analüsaatori puhul võib esmapilgul näha kõige suuremat hinnangute erinevust. Üks tudeng, kes oli eelmises iteratsioonis pannud hindeks 0, oli seekord tõstnud hinnet 8 peale. Seda seetõttu, et 2. iteratsioonis ei toonud analüsaator tema jaoks välja mitte ühtegi kantseliitlikku lauset, aga 3. iteratsioonis leidis analüsaator ühe vea. Teine tudeng,

kes oli varasemalt pannud hindeks 3, oli seekord teadmata põhjustel tõstnud hinnet 10 peale, olgugi et analüsaator ei leidnud kummalgi korral ühtegi olema + kesksõna kantseliidiga lauset.

Määrus saavas käändes analüsaatorit hinnati positiivsemalt kui varem. Üks tudeng tõstis oma hinnet 3 pealt 9 peale. Analüsaator oli 2. iteratsioonis toonud tema tekstis välja 8 kantseliitlikku lauset, aga 3. iteratsioonis 11 lauset. Logidest oli näha, et 2 lauset, mis olid 2. iteratsioonis välja toodud, olid 3. iteratsiooniks ära parandatud. Tõenäoliselt olid 3. iteratsioonis välja toodud laused tudengi silmis kvaliteetsemad.

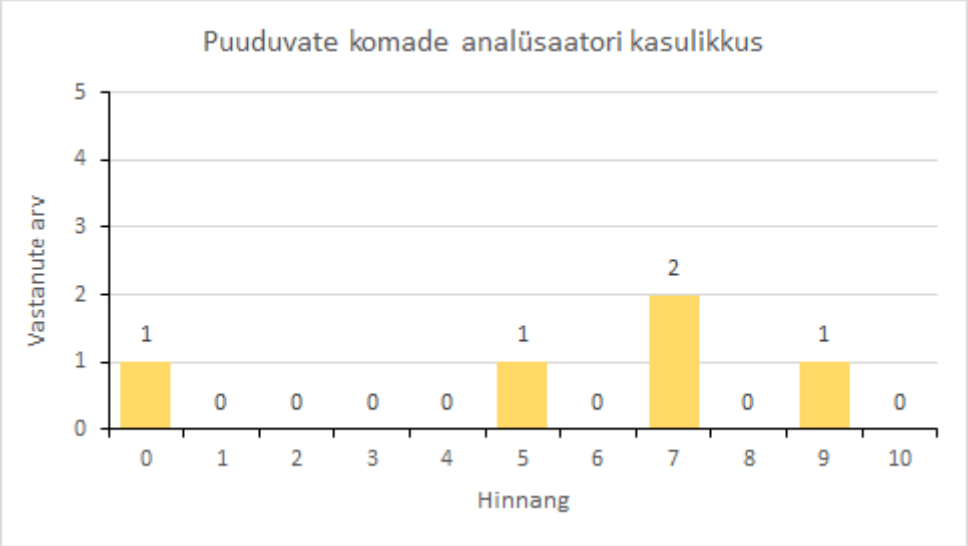
Teine tudeng muutis määrus saavas käändes analüsaatori hinde 2 pealt 0 peale, kuna analüsaator ei toonud talle ühtegi kantseliitlikku lauset välja. Kolmas tudeng muutis hinde 8 pealt 5 peale, kuigi välja toodud laused olid mõlemas iteratsioonis samad. Logisid uurides tundus, et tema lauseid oleks saanud parandada. Kantseliitlikud lauseosad olid *eesmärgiks on* ja *võimaluseks on,* aga 3. iteratsiooniks neid ei olnud parandatud. Sama tudeng parandas see-eest hiljem, pärast testimist, ära lauseosa *eesmärgiks on*.

Nominaalstiili hinnangud erinesid enamikel tudengitel vaid mõne punkti võrra. Üks tudeng oli teadmata põhjustel muutnud oma hinde 3 pealt 10 peale. Mõlemal korral ei toonud nominaalstiili analüsaator tema jaoks välja ühtegi kantseliitlikku lauset.

Neli inimest arvas, et kantseliidi analüsaatorite pakkumised olid asjakohased. Üks vastanu kommenteeris, et poolt-tarind ja olema-kesksõna analüsaatorid olid isegi valepositiivsete tulemuste puhul kasulikud, kuna need tuletasid meelde korrektset kirjastiili.

Kolmandas iteratsioonis lisati edasiarendusena puuduvate komade analüsaator. Neljal vastanul ei toonud puuduvate komade analüsaator välja ühtegi komaviga. Ühel vastanul tõi see välja 2 komaviga, mis mõlemad parandati ära.

Puuduvate komade analüsaatori kasulikkust hinnati positiivselt (vt Joonis 43). Tulemuste põhjal võib öelda, et EstNLTK osalausestaja pakub head võimalust selleks, et tuvastada lauses puuduvaid komasid.



Joonis 43. Puuduvate komade analüsaatori hinnangud 3. iteratsioonis

Kokkuvõttes leiti, et analüsaatoreid hinnati positiivselt. Testijate tagasisidest ja kommentaaridest selgus, et programmist on kasu, olgugi et vahepeal leitakse vale-positiivseid tulemusi. Tundub, et lõputöö teksti analüsaator on lõputöö kirjutamisel kasulik tööriist, aga kindlasti on seda võimalik veel edasi arendada.

7. Edasiarendamise võimalused

Nii isiklikud katsetused kui kasutajatega testimine tõid esile mitmeid edasiarendamise võimalusi. Selles peatükis on välja toodud mõned võimalused analüsaatori edasi arendamiseks.

7.1 Analüsaatorite edasiarendused

Testijate tagasisides mainisid tudengid, et korduvate sõnade analüsaator võiks korduvatele sõnadele pakkuda välja sünonüüme. Lõputöö algfaasis, kui kasutati EstNLTK versiooni 1.4.1, leidis analüsaator sõnadele sünonüüme. Sünonüümide leidmise funktsionaalsust ei olnud lõputöö kirjutamise ajal EstNLTK versiooni 1.6 implementeeritud. Kui EstNLTK arendajad tõstavad üle sünonüümide leidmise funktsionaalsuse versiooni 1.6, on seda funktsionaalsust võimalik ka lõputöö analüsaatori programmis kasutada.

Mõiste- ja tsitaadituvastaja ehk `QuoteAnalyzer` klassi puhul ei ole arvestatud haruldase erijuhuga, kus jutumärkides teksti sees leidub veel jutumärkide vahel olevat teksti. Näiteks võib tsitaadi sees olla otsekõne. Sellises olukorras võib `QuoteAnalyzer` arvata, et otsekõnet alustav jutumärk lõpetab tsitaati. Kuigi klass suudab küll eristada tsitaati või mõistet alustavat jutumärki ja lõpetavat jutumärki, võib sellegipoolest säärane olukord tekkida.

Kõige õigem lahendus selle jaoks oleks `QuoteAnalyzer` klassile anda loendur vastavalt sellele, kui sügaval mõistes või tsitaadis ollakse. Näiteks, kui tegu on tavatekstiga, on loenduri väärtus 0. Kui algab tsitaat, on loenduri väärtus 1. Kui tsitaadi sees on veel otsekõne, on loenduri väärtus 2. Kui otsekõne lõpeb, saab langetada loenduri väärtust 1 peale. Tsitaadi lõppedes on loenduri väärtus jälle 0, ehk tekst ei ole jutumärkide vahel.

Olema + kesksõna analüsaatori valepositiivsete tulemuste arv väheneks kõige rohkem siis, kui kuidagi leida meetod selleks, et eristada kesksõna ja omadussõna. EstNLTK lõputöö kirjutamise ajal sellist võimalust ei paku.

Puuduvate komade analüsaatori kvaliteet sõltub suuresti sellest, missuguseid edasiarendusi tehakse EstNLTK `ClauseSegmenter` klassile. Enamik praegustest valepositiivsetest lausetest tekib seetõttu, et korrektset lauset osalausestatakse valesti.

7.2 Sisendteksti formaadi muutmine

Programm võtab veebiliideses sisendiks JSON kujul infot. See tähendab, et saadetakse HTTP POST *request*, mille sisu on analüüsitav tekst. Analüüsitav tekst on sõnena puhtal tekstikujul.

Tihti märgistatakse tsitaate või mõisteid mitte jutumärkidega, vaid hoopis kaldkirjaga. JSON puhul ei saa tuvastada, et lauses „Sõna *I* tähendab eesti keeles *mina*“, et sõnad *I* ja *mina* on tegelikult mõisted ilma, et sisendtekst oleks muus formaadis. Selles spetsiifilises näites annaks umbisikulise tegumoe analüüs tagasisidet, et lauses leidub mina-vormis sõna *mina*. Umbisikulise tegumoe analüsaatori põhiline eesmärk on hoida autori isikut tagaplaanil, ent antud näite puhul see oma eesmärki ei täida.

Üks võimalus, kuidas seda probleemi lahendada, on võtta sisendtekst vastu mitte tavalise puhta tekstina, vaid HTML kujul. Näiteks Google Docs ja Microsoft Word dokumente on võimalik salvestada tekstifail HTML kujul. Sellisel puhul oleks alapealkirjad, tekstilõigud ja mõisted ka vastavate *tag*-idega märgistatud. Eelnevas lõigus mainitud problemaatilised lõigud oleksid vastavalt:

Edasiarendamise võimalused → <h1>5. Edasiarendamise võimalused</h1>

*Sõna I tähendab eesti keeles mina → Sõna I tähendab eesti keeles mina *

Niiviisi oleks võimalik parsida tekstist välja kaldkiri, pealkirjad ning alapealkirjad. Kuigi tulemus oleks kindlasti täpsem, ei saa garanteerida, et see lahendus täielikult töötaks. Näiteks salvestati idee katsetamiseks Microsoft Word-is lõputöö tekst *.html* faililaiendiga. Kuigi pealkirjad ja kaldkiri olid tõepoolest õigete *tag*-idega ümbritsetud, esines probleeme kodeeringuga. Vale kodeeringu puhul võivad täpitähed olla asendatud küsimärkidega kujul „  “. Lisaks tekkis ka faili salvestamisel palju ebavajalikku müra.

7.3 Teistele keeltele laiendamine

Programmi on võimalik laiendada ka teistele keeltele. Umbisikulisuse, lausepikkuse ja -keerulisuse ja sõnakorduste analüsaatorite puhul on üldreeglid kasutatavad teiste keelte jaoks.

Näiteks inglise keele jaoks oleks põhilisi muudatusi vaja teha just lausestaja, sõnestaja ja osalausestaja väljavahetamisega. EstNLTK funktsioonid töötavad vaid eesti keelega, ent näiteks inglise keele jaoks oleks võimalik kasutada rahvusvahelist loomuliku keele töötluse teeki NLTK (*Natural Language Toolkit*)[31].

NLTK teek pakub sarnaselt EstNLTK-ga võimalusi lausete, sõnade, osalausete, sõnaliikide ja muude tekstiomaduste leidmiseks, ent seda inglise keele jaoks. Analüsaatorid saaksid ingliskeelse teksti puhul kasutada NLTK võimalusi, et vajalik info kätte saada (vt peatükk 5.1).

Seejärel oleks vaja teisendada NLTK funktsioonide väljundid samale kujule, nagu need on EstNLTK puhul. Lõpuks saaks näiteks lausepikkuse- ja keerulisuse analüsaatoris rakendada samu reegleid, mida kasutati ka eestikeelse teksti puhul.

Korduvate sõnade analüsaatoris peab välja vahetama eesti lemmade sageduste andmestiku. Seejärel saaks kasutada Zipf seaduspärasust ingliskeelsete tekstide jaoks.

7.4 Kasutaja analüsaatorispetsiifilised eelistused

Üks rakenduse puudujääke on see, et kasutaja ei saa hetkel valida, missuguseid analüüse ta mingitel spetsiifilistel tekstilõikudel rakendada tahab.

Näiteks tuli kasutajatega testimisest välja, et ühe testinud tudengi instituudis on lubatud kasutada mina- või meie-vormi. Tema ei saanud see-eest kuskil valida, et umbisikulisuse analüsaator ei märgiks mina- või meie-vormi veaks.

Lisaks ei saa märkida, missuguseid valepositiivseid tulemusi ei soovi kasutaja järgnevates analüüsides enam näha. Näiteks võib tuua olukorra, kus kasutaja proovib lõputööde analüsaatorit teist korda.

Olgu edaspidi esimene analüüs viidatuna kui analüüs A ja teine analüüs kui analüüs B.

Analüüsis A ütles lausepikkuse ja -keerulisuse analüsaator, et mingi kindel lause on liiga pikk. Kasutaja ei nõustunud esile tõstetud lausega, pidades seda ise mõistlikuks lauseks. See-eest ei ole hetkeseisuga rakendusel kuidagi võimalik meelde jätta, et analüüsis B seda

[31] Natural Language Toolkit koduleht https://www.nltk.org/

lauset uuesti ebasobivaks ei märgita. Tulemusena märgitakse ka analüüsis B seesama lause liiga pikaks.

Lahendus oleks, et igal kasutajal oleks profiil, mis seotakse eelnevate analüüside logidega. Analüüsis B kontrollitakse iga pikaks märgitud lause puhul, kas lause on juba analüüsis A esile tõstetud. Kui pikk lause esineb nii analüüsis A kui ka B ja kasutaja on selle tegelikult sobivaks lauseks märkinud, siis võib selle analüüsis B vahele jätta. Siinkohal tekib probleem: kui analüüsis B on lause muutunud kasvõi ühe karakteri võrra, siis ei tuvastata seda enam sama lausena. Tulemusena märgitakse lause ikkagi liiga pikaks.

Funktsionaalsuse lisamine annaks kindlasti palju lisandväärtust, kuna analüüsides esineb valepositiivseid tulemusi. Valepositiivsed tulemused võivad aga mitmete analüüside jooksul (näiteks iganädalaste CGLearni saadetud tekstide puhul) osutuda soovimatuks müraks, mis häirivad tulemuste uurimist. Näiteks CGLearn saadab iganädalaselt raporti koos lõputöö analüsaatori tulemustega. Kui lõputöö kirjutaja ei soovi valepositiivset lauset muuta, oleks kasulik, kui ta saaks märkida lauset sobivaks. CGLearnis on küll olemas kasutajaprofiilid, ent lõputööde analüsaatori *backend* veel kasutajaspetsiifilisi eelistusi ei toeta.

7.5 Skaleeruvus

Peatükis 4.3 kirjeldati programmi analüüsiaegu ja piirangut, et server suudab korraga teenindada vaid nelja klienti. Väheste kasutajate puhul ei osutu see murekoht probleemiks, ent kindlasti mõjutab see piirang suuresti skaleeruvust. Kui kliente on juba rohkem, võib esineda olukord, kus server on ülekoormatud. Seetõttu ei pruugi kõik päringud saada analüüsitulemusi kätte.

Probleemi leevendamiseks on mitmeid variante. Üks võimalus on eraldada programmi *backend* ja veebiliides. Kuna *backend* on niikuinii eraldiseisev API, saab veebiliidese teenust serveerida eraldi serveril. Niiviisi ei pea üks server serveerima nii veebiliidest kui tegema analüüsi.

Kui kasutada API jaoks mitut serverit, on võimalik kasutada *koormuse jaotamist*. Niiviisi saaks analüüside koormust jaotada mitme serveri vahel, mis tähendaks seda, et kasutajad ei peaks niivõrd palju ootama teiste kasutajate analüüside taga.

Teine valik oleks kasutada võimsamat serverit. Nagu oli näha peatükis 4.3, siis mida võimsam on serveri protsessor, seda kiirem on analüüs.

Kõige aeganõudvamad protsessid on EstNLTK 1.6 funktsioonid, nagu näiteks lausestamine ja süntaksanalüüs. Kuna versioon 1.6 on veel arendamisjärgus, võib eeldada, et tulevikus optimeeritakse versiooni veel rohkemgi. Lõputöö kirjutamise ajal ütles EstNLTK arendaja Siim Orasmaa, et versioon 1.6 puhul oli optimeerimisega veel vähe tegeletud. Optimeerimine tähendaks kiiremat vastust, kuna analüüs saaks kiiremini valmis.

8. Kokkuvõte

Käesolevas bakalaureusetöös loodi lõputööde teksti analüsaator, mis annab lõputöö kirjutajale automaatset tagasisidet lõputöö teksti kohta. Katsetati erinevaid reegleid ja analüüse, mis võiksid lõputöö tekstide automaatsel tagasisidestamisel kasulikud olla.

Põhiline eesmärk oli katsetada analüüse, mida saaks kasutada CGLearni juhendaja ja juhendatavate tööde tekstide jaoks. Katsetades leiti, et loodud analüüsid töötasid hästi ka tudengite tööde puhul, kes pole CGLearn õpikeskkonnaga seotud.

Analüüside väljatöötamiseks uuriti teadusteksti omadusi. Kirjeldati teadustekstile omaseid tunnuseid ning prooviti luua analüsaatoreid, mis aitaksid kirjutajal teadusteksti tavasid järgida.

Vaadati olemasolevaid sarnaseid programme ja uuriti, mida need hästi teevad ning mis on nende suurimad puudused. Sarnased programmid võeti analüsaatorite loomisel eeskujuks ning prooviti parandada nende suurimaid puudusi. Näiteks olid sarnaste programmide puudused lahtise API puudumine või see, et need töötasid ainult ingliskeelsete tekstidega.

Loodi umbisikulise tegumoe, korduvate sõnade, lausepikkuse ja -keerulisuse ning puuduvate komade analüsaatorid. Kaarel Sõrmuse lõputööst võeti üle kantseliidi analüsaatorid, millele tehti mõningaid edasiarendusi. Selleks, et loodud analüsaatorite kasulikkust ja efektiivsust tõsta, loodi abistavad klassid. Abistavad klassid tuvastasid ja eemaldasid viiteid ning jutumärkide vahel olevat teksti.

Analüsaatoreid katsetati erinevate reeglitega. Reegleid kohandati või muudeti vastavalt sellele, kas analüüside tulemustega jäädi isiklikult katsetades või kasutajatega testides rahule.

Kasutajatega testimiseks loodi veebiliides. Niiviisi said analüsaatoreid katsetada ka tudengid, kes polnud CGLearn õpikeskkonda registreeritud. Rakendust testisid 5 tudengit, kellest 3 kasutasid testimiseks veebiliidest.

Kasutajatega testides prooviti iteratiivset testimist. See tähendab, et tudengitel paluti kasutada rakendust ja anda sellele tagasisidet. Tagasiside jaoks kasutati küsimustikku, mille vastuste põhjal tehti analüsaatorites muudatusi. Kui muudatused said valmis, algas uus iteratsioon ja tudengitel paluti rakendust uuesti testida. Kokku testiti rakendust 3 iteratsiooni vältel.

Tudengitel paluti igas iteratsioonis hinnata analüsaatoreid skaalal 0-10, kus hinnang märgistas analüsaatori kasulikkust. Uuriti, mis asjaoludel võisid hinnangud erinevates iteratsioonides erineda.

Küsimustikust selgus, et ühes iteratsioonis oli näiteks lausepikkuse ja -keerulisuse analüsaatorit hinnatud kõrgelt, aga järgmises madalalt seetõttu, et järgmiseks iteratsiooniks said vigased laused parandatud. Kuna tekstis ei olnud enam viga, peeti analüsaatorit vähem kasulikumaks.

Kuigi kantseliidi analüsaatorite tulemustes esines valepositiivseid vastuseid, hinnati isegi valepositiivseid tulemusi kohati hästi. Seda seetõttu, et need tuletasid tudengile meelde head kirjastiili.

Iteratiivsest testimisest selgus, et üldiselt jäid tudengid programmiga rahule. Analüsaatoreid hinnati positiivselt. Tagasisidest võiks järeldada, et kõige kasulikumad olid tudengite jaoks lausepikkuse ja -keerulisuse ning korduvate sõnade analüsaatorid. Mõned probleemid, mida märgati isikliku katsetamise jooksul või mida tudengid välja tõid, said lõputöö jooksul parandatud. Enamik probleeme ja murekohti saab lahendada edasiarendustena.

Tänan käesoleva lõputöö juhendajat Raimond-Hendrik Tunnelit, kes aitas analüsaatorite loomise ja kirjaliku osaga. Tema soovitused ja arvamused muutsid lõputöö kvaliteeti paremaks. Tänan kõiki tudengeid, kes testisid lõputöö teksti analüsaatorit ja andsid sellele põhjalikku tagasisidet. Tänan Sandra Sternhofi, kes aitas keeleliste küsimuste ning lõpliku töö ülevaatuse ja parandamisega.

Viited

[1] TÜ arvutiteaduse instituudis kaitstavate lõputööde nõuded ja hindamine. https://www.cs.ut.ee/sites/default/files/cs/ati_loputoo_juhend_ja_hindamiskriteeriumid_2017.pdf (10.12.2019).

[2] E-kursus „Teadustöö alused“. Tartu ülikool. https://sisu.ut.ee/teadustoo_alused/avaleht (10.12.2019).

[3] Tunnel, Raimond-Hendrik. Lõputööde tõhusam juhendamine. Universitas Tartuensis. Tartu Ülikooli ajakiri. Oktoober 2019, nr 9. https://www.ajakiri.ut.ee/artikkel/3364 (21.04.2020).

[4] Readable blogi. The Flesch Reading Ease and Flesch-Kincaid Grade Level. https://readable.com/blog/the-flesch-reading-ease-and-flesch-kincaid-grade-level/ (21.04.2020).

[5] Microsofti infoleht. Editor’s spelling, grammar and refinement availability by language. https://support.office.com/en-us/article/editor-s-spelling-grammar-and-refinement-availability-by-language-ecd60e9f-6b2e-4070-b30c-42efa6cff55a (29.04.2020).

[6] Intel tootespetsifikatsioon. Intel® Xeon® Processor E5-2650. https://ark.intel.com/content/www/us/en/ark/products/64590/intel-xeon-processor-e5-2650-20m-cache-2-00-ghz-8-00-gt-s-intel-qpi.html (01.05.2020).

[7] AMD tootespetsifikatsioon. AMD EPYC™ 7702 protsessor. https://www.amd.com/en/products/cpu/amd-epyc-7702 (01.05.2020):

[8] Full Stack Python koduleht, Flaski alamosa. https://www.fullstackpython.com/flask.html (10.04.2020).

[9] „Riiklik programm: Eesti keeletehnoloogia“ projekt EKT57. https://www.etis.ee/Portal/Projects/Display/0b4c5efa-6b2c-40cd-aff6-4a6b6668560a (10.04.2020).

[10] Reinsalu, Riina. Viitamine iseendale teadustekstides: töös käsitletakse, töös käsitlen, töö käsitleb.... Oma keel, 2, 2015, 72-77. http://www.emakeeleselts.ee/omakeel/2015_2/10.pdf (10.04.2020).

[11] Hirsjärvi, Sirkka; Remes, Pirkko; Sajavaara, Paula. Uuri ja kirjuta. Tallinn: Medicina. 2005.

[12] Piantadosi, Steven T.. Zipf’s word frequency law in natural language: A critical review and future directions. *Psychonomic bulletin & review*, 2014, 21

[13] Smith, Reginald. Investigation of the Zipf-plot of the extinct Meroitic language. Glottometrics, 2007, 15

[14] Corral, Álvaro; Boleda, Gemma; Ferrer-i-Cancho, Ramon. Zipf's Law for Word Frequencies: Word Forms versus Lemmas in Long Texts. PLoS ONE, 2015, 10. https://doi.org/10.1371/journal.pone.0129031 (13.04.2020).

[15] Reet, Kasik. Kuidas kirjutada selget tarbeteksti. Oma Keel, 14, 2007, 46-51 http://www.emakeeleselts.ee/omakeel/2007_1/OK_2007-1_05.pdf (01.05.2020).

[16] Liivaku, Uno 1972. Kust king keelt pigistab. Tallinn: Valgus.

[17] Sternhof, Sandra. Kõneaktiverbid otsekõne saatelauses Viivi Luige romaanis „Seitsmes rahukevad". Tartu Ülikooli eesti ja üldkeeleteaduse instituudi bakalaureusetöö. 2019. http://dspace.ut.ee/bitstream/handle/10062/64418/Sternhof_bak2019.pdf?sequence=1&isAllowed=y (23.04.2020).

[18] Sõrmus, Kaarel. Kantseliidi- ja paronüümituvastaja. Tartu Ülikooli arvutiteaduse instituudi lõputöö. 2017. https://dspace.ut.ee/handle/10062/65688 (14.04.2020).

[19] Muischnek, Kadri; Müürisep, Kaili. Eesti keele sõltuvuspuude pank ja selle keeleteoreetilised lähted. Emakeele Seltsi aastaraamat, 2016, 62

[20] Pullerits, Egle. Kuidas hoiduda kantseliidist. Tartu: Keelehooldekeskus. 2010.

[21] Eesti Keele Käsiraamat 2007, Oleviku kesksõna https://www.eki.ee/books/ekk09/index.php?link=M_78 (27.04.2020).

[22] Eesti keele käsiraamat 2007, Nominalisatsioon https://www.eki.ee/books/ekk09/index.php?link=S%C3%9C_138 (27.04.2020).

[23] EstNLTK 1.6 Clause Segmenter dokumentatsioon https://github.com/estnltk/estnltk/blob/version_1.6/tutorials/nlp_pipeline/B_09_clause_segmenter.ipynb (27.04.2020).

[24] Nielsen, Jakob. Why You Only Need to Test with 5 Users. Nielsen Norman Group. https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/ (15.04.2020).

Lisad

I. Sõnastik

1. **Abitegusõna** – tegusõna, mis täiendab teiste tegusõnade semantilisi kategooriaid, näiteks pööret, kõneviisi ja aega[32].
2. **API** – rakendusliides, millega programmid saavad üksteise vahel infot vahetada[5]. Näiteks CGLearn õpikeskkond esitab päringu lõputöö teksti analüsaatori API-le, et saata analüüsimiseks lõputööde tekste.
3. **Backend** – rakenduse osa, mis koosneb andmebaasist, serverist ja programmi loogikast. Põhiline eesmärk on anda API kaudu kasutajaliidesele informatsiooni[33].
4. **CPU-bound** – rakenduse omadus, kus selle töökiirus muutub vastavalt protsessori jõudlusele[34].
5. **JSON** – Javascript Object Notation ehk formaat, millega rakendused saavad üksteise vahel infot jagada[35].
6. **Korpus** – loomuliku keele tekstide kogu[36].
7. **Lemma** – sõna algvorm.
8. **Mikroteenus** – programm, mis teenindab teist suuremat rakendust. Tarkvaraarenduses on mikroteenused arhitektuuristiil, kus rakendus koosneb mitmest väiksemast teenusest, kus igal teenusel on oma eesmärk[37]. Näiteks CGLearn on eraldiseisev rakendus, mis kasutab lõputöö teksti analüsaatori teenust.
9. **Raamistik** – tarkvara, mis on loodud selleks, et arendajatel oleks lihtsam uusi rakendusi luua[38].
10. **Sõnajuur** – sõna morfoloogiliselt jagamatu osa ehk lihttüvi, millel pole muid liiteid[39].
11. **Sõnaliik** – keele sõnade klassid[40]. Näiteks nimisõna, omadussõna, tegusõna.
12. **Virtuaalmasin** – rakendus, mille eesmärk on simuleerida teist arvutit. Sisuliselt ühe arvuti jooksutamine teises arvutis[41].

---

[32] Abitegusõna https://www.lexico.com/definition/auxiliary_verb
[33] Backend https://blog.udacity.com/2014/12/front-end-vs-back-end-vs-full-stack-web-developers.html
[34] CPU-bound https://stackoverflow.com/questions/868568/what-do-the-terms-cpu-bound-and-i-o-bound-mean
[35] JSON https://www.json.org/json-en.html
[36] Korpus https://et.wikipedia.org/wiki/Keelekorpus
[37] Mikroteenused https://microservices.io/patterns/microservices.html
[38] Raamistikud https://hackr.io/blog/what-is-frameworks
[39] Sõnajuur http://entsyklopeedia.ee/artikkel/juur_(keeleteadus)1
[40] Sõnaliik https://www.taskutark.ee/m/sonaliigid/
[41] Virtuaalmasin https://azure.microsoft.com/en-us/overview/what-is-a-virtual-machine/

II. Veebiliidese kasutusmugavuse edasiarendus

Praeguses veebiliideses saab kasutaja sisestada oma teksti ainult kopeerides. See tähendab, et kasutaja peab võtma lahti oma lõputöö tekstifaili ja seejärel kopeerima liidesesse oma teksti. Olgugi et see on võrdlemisi kiire, võtab see sellegipoolest aega, mida kasutaja tõenäoliselt kulutada ei taha.

Üks variant, kuidas protsessi lihtsustada, oleks lisada veebiliidesele võimalus laadida üles oma lõputöö teksti fail. Seejärel saaks rakendus parsida faili sisu ja muuta see ümber tekstikujule. Lõpptulemusena saaksid analüüsid toimuda samamoodi nagu ennegi, ent kasutajamugavus on paranenud. Õpikeskkonnas CGLearn on olemas parser, mis suudab sisse lugeda PDF-faile ning mida oleks võimalik funktsionaalsuse jaoks kasutada.

III. Kaasapandud failid

Programmi lähtekood asub kaustas *thesisanalyzer*.

Programmi käivitusjuhend ja käivitamiseks vajalikud failid asuvad kaustas *setup*.

Küsimustikud ja nende vastused asuvad kaustas *feedback*.

Logitud analüüside tulemuste failid asuvad kaustas *logs/results*.

Salvestatud logide HTML failide loomise juhend asub failis *logs/instruction.txt*.

IV. Analüsaatori tulemused CGLearn veebiliideses

Joonistel 44, 45, 46 ja 47 on toodud välja CGLearni raportid, kus on näha lõputöö teksti analüsaatori tehtud analüüse.

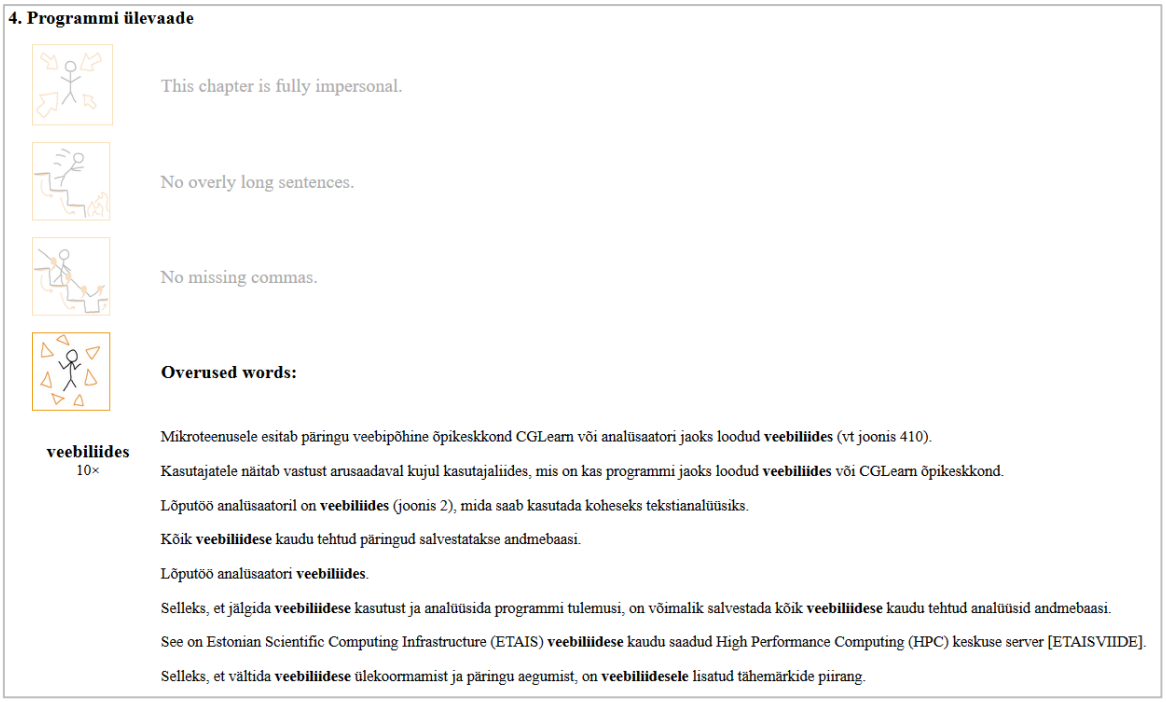
Overly long sentences:

Näiteks klassifitseeris analüsaator järgneva lause keeruliseks: Kuna käesolevas uurimistöös uuritakse verbe, mida on kasutatud otsekõne saatelausetes, ning vähesel määral ka saatelausete paiknemist, siis selles peatükis antakse lühike ülevaade, kuidas üldse tegelaskõnet kirjanduses on võimalik edasi anda ning milline tähtsus on saatelausetel dialoogide puhul.

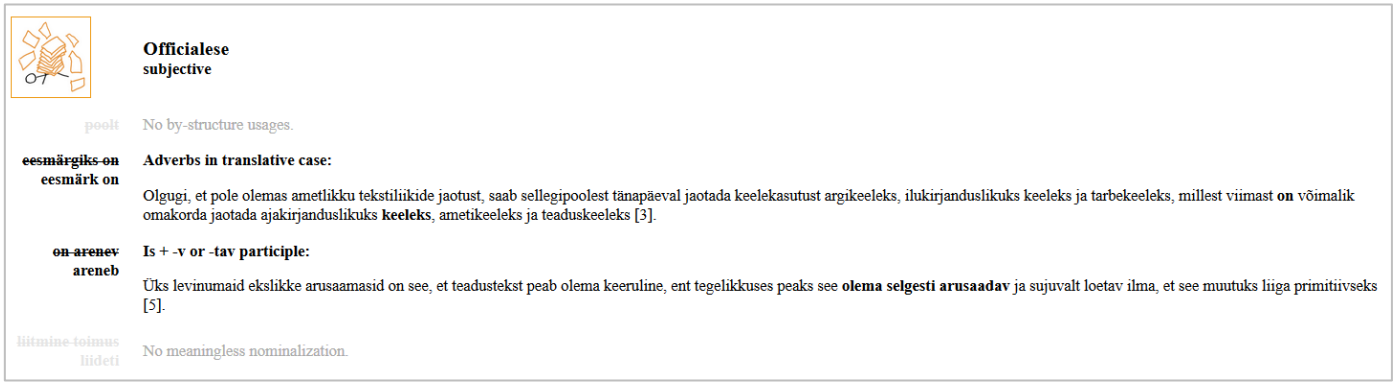
Näiteks pidas analüsaator kantseliitlikuks järgmised sõnapaarid: näiteks on; lisaks on; selleks on; selliseks on ja tulemuseks on.

Seetõttu tehti edasiarendus: kui sõna vastab eelmises lõigus mainitud reeglitele, ent sõna on kas lisaks, näiteks, selleks, selliseks või tulemuseks, siis seda kantseliidiks ei märgita.

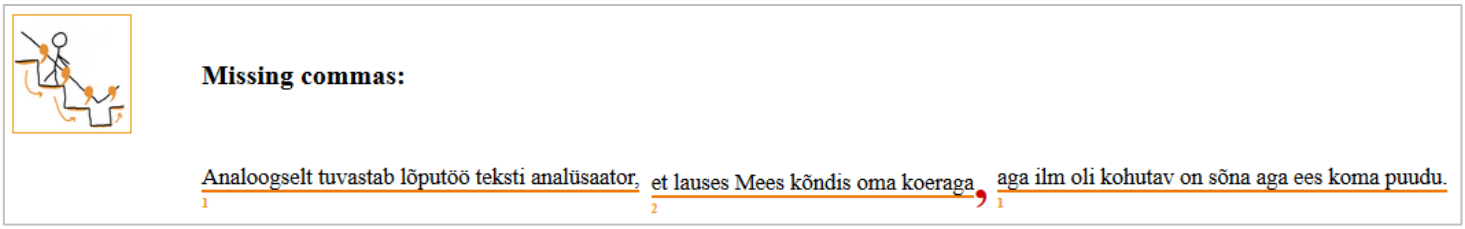
Joonis 44. Lausepikkuse ja -keerulisuse analüsaatori välja toodud laused



Joonis 45. Peatüki „Programmi ülevaade“ analüüsi tulemus



Joonis 46. Kantseliidi analüsaatorite välja toodud laused



Joonis 47. Puuduvate komade analüsaatori välja toodud vigane lause

V. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Karl Erik Karindi**,

(autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose
   Lõputöö teksti analüsaator,

   (lõputöö pealkiri)

   mille juhendaja on **Raimond-Hendrik Tunnel**,

   (juhendaja nimi)

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Karl Erik Karindi

07.05.2020