TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Andre Viibur

Sagedasemate kirjavigade parandamise õpiprogramm

Bakalaureusetöö (9 EAP)

Juhendaja: Sven Aller

Tartu 2023

Sagedasemate kirjavigade õpiprogramm

Lühikokkuvõte:

Käesoleva bakalaureusetöö eesmärk oli kasutada eesti keele õppijakorpust, et leida eksamitel tehtavad sagedasemad kirjavead ning neid vigu kasutada veebiõpiprogrammi loomisel. Programm peaks sobima põhikooli kui ka haridustee lõpetanud eestlastele ja mitte-eestlastele. Eesmärgi täitmiseks loodi Google Colab leht vigade leidmiseks ja korrektsete versioonide lisamiseks ning täispinu rakendus, kus kasutaja saab oma keeleoskusi kontrollida. Töös on kirjeldatud algoritmi ja veebirakendust.

Võtmesõnad:

Keeletehnoloogia, ortograafia õpiprogramm

CERCS: P175 Informaatika, süsteemiteooria

Learning programm for correcting more frequent spelling errors

Abstract:

The aim of this Bachelor's thesis was to use the estonian language studentcorp to find the most common typos in exams and use those mistakes to create a web application. The programm should be usable from middle school students to people who no longer study, both estonians and non-estonians. To make this happen, a Google Colab page was made to find the mistakes, and a full-stack application to test your language knowledge was created. The thesis includes a description of the algorithm and web application.

Keywords:

Language corps, web app, data processing

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Kirjakeel on see, mis eestlased rahvusena ühte seob (Ross, 2019). Ühtne keel on seega väga tähtis kultuuri ja ajaloo edasiandmise jaoks. Seetõttu pööratakse koolides väga palju tähelepanu ka kirjaliku keele normikohaste reeglite õppimisele, sealhulgas ortograafiale ehk „kirjalikus keeles kasutatav normikohane reeglite süsteem, mis hõlmab sõnakuju kirjapanekut“ (ortograafia, 2023).

Õpilastel on vaja läbitava kooliastme diplomi saamiseks sooritada vastava taseme eesti keele test. Keskkooli lõpetamise jaoks on selleks testiks riigieksam, kus tuleb kirjutada kirjand ning vastata lugemisküsimustele. Kirjandid parandatakse ja osad neist jõuavad ka erinevatesse andmebaasidesse, kus neid saab keeletehnoloogiliste vahenditega uurida ning töödelda. Võimalik on teha erinevaid analüüse õpilaste soo, elukoha, emakeele ning vigade tüüpide ja sageduste kohta. Märgendatud tekste saab kasutada ka õppematerjalide loomisel.

Käesolev töö on rakenduslik uurimistöö, mille eesmärgiks on uurida andmebaasides olevaid märgendatud kirjandeid, neid töödelda ning kasutada õpiprogrammi loomisel. Andmetest loodi kirjavigade filtreerimise algoritm, mille tulemusi kasutada vigade parandamise veebi õpiprogrammis. Eesmärk on luua ortograafia õppimise materjal, mida saavad kasutada nii algajad kui ka arenenumad keeleoskajad.

Töö on jaotatud kuueks peatükiks. Esimeses peatükis kirjeldatakse ortograafia olulisust ning erinevaid tekstide andmebaase ehk korpuseid. Teises peatükis tutvustatakse juba olemasolevaid õigekirja õppimise veebimaterjale. Kolmandas peatükis seletatakse lahti korpusest saadud andmete töötlemine. Neljandas peatükis kirjeldatakse töödeldud andmete põhjal valmis saanud veebirakendust ning teenuseid, millega see veebis üleval on. Viies peatükk on rakenduse jaoks valminud Dockeri failide tutvustamiseks ning kuuendas peatükis on analüüsitud kasutajate tagasisidet, tekkinud probleeme ning edasiarendamise võimalusi.

1. Ortograafia tähtsus ning tekstikorpused

Selles peatükis tuuakse välja mitmeid erinevaid õigekirjaga seotuid uurimistöid ning mis viisidel kehv õigekiri inimeste arvamust mõjutada võib. Tutvustatakse ka korpuseid: mis need on ning kuidas on neid loodud.

1.1. Ortograafia õpetamine ja õppimine

Eesti koolides õpetatakse eesti keelt alates esimesest klassist (Põhikooli riiklik õppekava, 2011). Enne seda omandavad lapsed keele kohta teadmisi lasteaiast, vanematelt, mängukaaslastelt, saades sellega keelest ning selle korrektsest kasutamisest oma arusaama. Peale õpingute algust on neil 9-12 aastat, et saada keel piisavalt selgeks, et edukalt sooritada õpitava haridustaseme lõpueksamid.

Eesti keele lõpueksam on kohustuslik nii põhi- kui keskkooli lõpetamiseks, mis tähendab, et seda teevad igal aastal kümned tuhanded õpilased. Igal aastal väljastatakse ka statistilised andmed, milliste koolide õpilased saavutavad paremaid tulemusi ning millistel koolidel läheb halvemini. Keskkooli tasemel sõltub eksamitulemus väga palju reeglite teadmisest, aga gümnasistidelt eeldatakse juba reeglite teadmist ning keskendutakse rohkem kirjalikule väljendusoskusele.

1.2. Ortograafia olulisus

Õigekiri on väga tähtis keele aspekt, mida hinnatakse eksamitel rangelt, aga mis ka õpilastele väga ei meeldi. Seda kinnitab ka 2022. a. tehtud lõputöö, mis analüüsis emakeeleolümpiaadi lõppvooru. Olümpiaadi tagasisidest ilmnes, et ainult üks õpilane valis õigekirja ülesanded lemmikuteks (Jõgar, 2022, lk 20). Samas polnud keegi seda teemat valinud ka kõige keerulisemaks (Jõgar, 2022, lk 25).

Magistritöö, mis uuris riigieksamikirjandite kirjavigasid, tõi välja, et heades ja keskmistes töödes esineb kõige rohkem interpunktsioonivigu ning nõrkades eksiti rohkem kokku- ja lahkukirjutamisega (Põldaru, 2018, lk 57). Nõrkades töödes esines ka kolm kuni neli korda rohkem häälikuortograafiavigu kui keskmistes või heades töödes, millest enamus oli võõrsõnade tõttu (Põldaru, 2018, lk 25). Vigade tegemisest ei pääsenud ka hindajad, kus 400st uuritud tööst leidus 31 viga, mis ei olnud hindamise hetkel kehtinud keelenormingu põhjal tegelikult vead (Põldaru, 2018, lk 48).

Saab järeldada, et õigekiri on keeruline ning vigu teevad nii õpilased, kes on eksamite ajaks seda üle poole oma elust koolipingis õppinud, ning ka õpetajad ja keeleteadlased.

Õigekeelsusvead riigieksamikirjandites tähendavad madalamat punktide arvu, mis võib teha ülikoolis soovitud erialale sissesaamise raskemaks.

Ortograafiavead mõjutavad ka inimeste arvamust kirjutajast. Senkeli tööst (Senkel, 2016, lk 36) tuli välja, et kirjavead mõjutasid uuritavatest parameetritest kõige rohkem teiste inimeste poolt kirjutaja intelligentsuse tajumist. Inimesed seovad vigaseid tekste madalama intelligentsiga, mis võib teha näiteks tööle saamise keerulisemaks.

1.3. Tekstikorpused

Ortograafia uurimisel omavad tähtsat rolli tekstikorpused, mis on tekstidest koosnevad elektroonilised andmekogud. Korpused võivad sisaldada ühe või mitme keele tekste ning neid uurides saavad teadlased teada, kuidas keelt reaalsuses kasutatakse (Mis on tekstikorpus, 2022).

Korpuseid läheb vaja, et uurida keelt, selle muutumist ning kasutamist. Nad sisaldavad palju materjali, mida saab ära kasutada erinevate õpiprogrammide ja masinõppe algoritmide jaoks, luues paremaid teksti tuvastamis, tõlkimis ning tõlgendamisviise.

Käesolevas töös kasutatud vigade ning neid vigu sisaldavate lausete peamine allikas on EMMA (Korpusest, s.a.) ehk eesti keele õppija korpus. Korpus on loodud Tartu Ülikoolis ning sisaldab erinevaid eestikeelseid tekste nagu näiteks 12. klassi eksami- või tasemetööd.

Teiseks allikaks on EVKK, mis on Tallinna Ülikooli poolt loodud eesti vahekeele ehk õppijakeele korpus. EVKKle on loodud korduvate õigekirjavigade sagedusloend, mis on koostatud korpuse alusel ning mis sisaldab vigu ja vigaseid lauseid (Eesti vahekeele korpus. Statistika, s.a.).

1.3.1. Eesti keele õppija korpus EMMA

Korpuses EMMA (Korpusest, s.a.) on üle 13 500 teksti, millest ~3800 tuleb Tartu Ülikoolilt ning ~9700 Eesti Keele Instituudilt. Kokku on nendes tekstides üle 231 000 lause ning peaaegu 3 miljonit sõna. Tekstide hulgas on 9. ja 12. klassi eesti keele lõpueksamite tekste, kirjalikke tunnitööde tekste ja loovtöösid. Filtreerida saab töid aasta, klassi, emakeele, maakonna jne järgi.

Eesti keele õppija korpus sisaldas algselt 1999.-2017. aasta 9. ja 12. klassi eksamitöid ning muid õppimisprotsessis koostatud tekste. Alates 2019. aastast tehakse koostööd Eesti Keele Instituudi ning Haridus- ja Noorteametiga. Koostöö tulemusena laienesid korpuse, tekstide kui ka eagruppide valikud (Korpusest, s.a.).

Korpuse loomiseks sisestatakse andmebaasi andmeid käsitsi. Peale seda kasutatakse EstNLTK 1.4, et sisend lausestada ja märgendada. Hetkel on ka 2470 teksti, mis sisaldavad käsitsi märgendatud vigasid. Nende tekstide peale on tuvastatud peaaegu 70 000 viga, mis vastavad eesti keele riigieksamil eristatavatele veatüüpidele (Korpusest, s.a.), mille alla kuuluvad sisu, õigekeelsus, stiil ja vorm.

Vead on kategoriseeritud ka vigade leidjate järgi, kus märgendites on välja toodud, milline parandaja vastava vea üles märkis.

1.3.2. Eesti vahekeele korpus EVKK

Tallinna Ülikooli keelekorpus on EVKK (Eesti vahekeele korpus. Statistika, s.a.) ehk eesti vahekeele korpus, millesse on koondatud erinevaid eesti keele tekste ning alamkorpuseid. Tekste on kokku ~12 500, sõnu on kokku ~3 450 000. Tekstid jagunevad keele ning tekstitüübi järgi ning ainus, mis on kõigi tekstide puhul teada, on keel, milles need kirjutati. Kõigil muudel jaotustel on domineerivaks teadmata klassifikatsioon.

EVKK keelekorpusest on välja kasvanud keskkond ELLE, mis on eesti keele õppimiseks ja õpetamiseks ning kus saab sooritada erinevaid keelelisi analüüse. Keskkonda kasutatakse EVKK täiendamiseks, loovutades ELLE-s kirjutatud ja analüüsituid tekste (Mis on ELLE?, s.a.). Korpus ning keskkond on seega pidevas arengus ja pakutav tekstianalüüs muutub ainult paremaks.

1.4. Väljavalitud vead

Kirjavigade tüüpe on erinevaid ning inimesed teevad neid erinevates kogustes. Tallinna Pedagoogikaülikooli magistritöös kirjakeele seisundist Eesti koolis (Raik, 2003) leiti, et sõnade kokku- ja lahkukirjutamine, on üks levinuimaid veatüüpe ning moodustab põhiosa enamike kirjandite ortograafiavigadest. Seda kinnitab ka Põldaru (Põldaru, 2018, lk 28), kes küll oma töös ei olnud neid õigekirjavigade liike eraldi välja toodud. Lisaks leidis ka Raik, et suur osa vigadest tuli sulghäälikute valesti kirjutamisest ning lühikese ja pika hääliku vastu eksimisest (Raik, 2003, lk 16-17).

Kokku- ja lahkukirjutamine on üleüldiselt väga levinud vealiik ja sulghäälikute ning lühikese ja pika hääliku vastu eksimine on väga levinud düsgraafide puhul. Antud veatüübid on seega märgatavad probleemid nii lõpukirjandite kirjutamisel kui ka inimeste elus peale kooli. Lõputöö käigus valmib algoritm EMMA tekstidest õigekirjavigade leidmiseks ning brauseri õpiprogramm, kus leitud vigasid parandada saab.

2. Varem loodud eesti keele veebiõppematerjalid

Interneti leviku ja populaarsuse kasvuga levivad ka veebis olevad õppematerjalid. Seega veebilehed, kus on väga palju erinevaid õppematerjale nagu Coursera, Udemy, Khan Acadamy jt muutuvad järjest populaarsemaks. Koroonaviiruse pandeemia oli aeg, millal paljud inimesed õppisid oma kodudes, kasutades veebimaterjale ning internetti. See näitas, et koolist ja klassiruumist eemal on võimalik õppida ning õpetada kooli lõpetamiseks vajaminevat. Seda teadmist saab ära kasutada ka veebis olevate õppematerjalide loomisel ning kasutamisel.

Tartu Ülikoolis varem lõputöö käigus loodud eesti keele õppevahendeid on mitmeid. Üks neist on eesti keele käänete õppimise programm „Õpime käändeid“ (Halling, 2016, lk 14). Ortograafia õppimiseks on olemas ka „Eesti õigekiri“ ja „Käänuk“. Esimese neist on loonud Erika Rummel, ning seal on teemade kaupa reegleid, näiteid ning ülesandeid (Rummel, s.a.). Teine neist on keeleteadlaste loodud õppematerjal sihitiste jaoks (Käänuk, 2019). Mõlemad on näited veebis olevatest eesti keele õpiprogrammidest, mis on loodud inimeste keelelise taseme tõstmiseks.

2.1. Õpime käändeid

„Õpime käändeid“ valmis bakalaureusetööna 2016. aastal ning kasutab eesti keele keeleressurssi nimega ilukirjanduskorpus, ning õpiprogrammi eesmärk on abistada kasutajat käänete õppimisel (Halling, 2016, lk 5). Valida saab kahe õppeviisi ning viie erineva testi pikkuse vahel. Testi pikkused on 5 kuni 25 lauset ning iga järgmine valik on viie lause võrra suurem eelmisest.

Valides testi tüübiks õige käände leidmise, peab kasutaja lauses ära märgitud sõnale määrama käände ning selle, kas see on ainsuses või mitmuses (Halling, 2016). Valides korrektselt nii käände kui mitmuse, kiidetakse kasutajat ja saab minna järgmise lause juurde. Valides ühe neist ebakorrektselt, antakse kasutajale sellest teada ning tuuakse välja ka õige vastus.

Valides testi tüübiks sõna õigesse käändesse panemise, saad valida, milliseid käändeid soovitakse harjutada. Käänded on grupeeritud järgnevatesse valitavatesse gruppidesse:

- Nimetav, omastav ja olev kääne
- Kohakäänded
- Osastav kääne
- Saav, rajav, ilmaütlev ja kaasaütlev kääne

- Kõik käänded

Valides ühe neist, viiakse kasutaja uuele lehele, kus antakse ette ülesanne. Kasutajale on antud sõna, kääne ning näitelause algus. Kirjutades vale vastuse, kuvatakse kasutajale modaalaken, kus on antud sõna õige versioon. Korrektselt käänamise puhul tuleb ette õnnitlus.

Õpiprogrammi lausete jaoks valiti eesti ilukirjanduskorpus, sest korpus on suur ning ilukirjanduslikke tekste kasutatakse koolides olevates eesti keele grammatikat õpetavates materjalides (Halling, 2016, lk 18). Halling tõi välja ka korpuse iseärasused, näiteks ei olnud kõik laused ühtemoodi märgendatud ning kasutusel olid ka erinevad jutumärgid. Erinevused tegid korpuse töötlemise raskemaks, aga märgatavaid probleeme ei esinenud.

2.2. Eesti õigekiri

Eesti õigekiri on väga põhjalik ning hea veebis olev õppevahend õigekeelsuse õppimiseks. Õppida saab numbrite õigekeelsust, kokku- ja lahkukirjutamist, sõnaliike, suurt ning väikest algustähte, võõrsõnade õigekeelsust ning täheortograafiat. Need jaotused on omakorda tehtud alampeatükkideks, kuhu on lisatud töölehed, harjutused, näited, reeglid ning erandid (Rummel, s.a.).

Õppetöö on mitmekesine ning õppimisele lähenetakse väga mitmekülgselt. Harjutusi on mitmeid erinevaid tüüpe:

- Ristsõna
- Vea hiirega sõnad reeglite juurde
- Täida lünk
- Vali õige

Lisaks veebimaterjalidele on lehel saadaval ka alla laaditavad töölehed, mida saavad õppijad ning õpetajad soovi korral välja printida, ise lahendada või õpilastele jagada.

Leht valmis koostöös Tiigrihüppe Sihtasutusega ning on väga põhjalik ja kasulik materjal kõigile eesti keele õppijatele. Õppematerjal on väga kõrge kvaliteediga ning annab õppijale kõik vajaliku, et arendada oma keelelisi oskusi. Leht vajaks aga graafilise kasutajaliidese uuendamist ning olenevalt vajadusest ka reeglite kaasajastamist.

2.3. Käänuk

Käänuk on keeleteadlaste loodud veebimäng, mille sihtgrupp on eesti keelt kui teist keelt kõnelevad inimesed. Õppematerjali loomisele on kaasa aidanud Tartu Ülikool ning erinevad koolid ja töörühmad (Käänuk, 2019).

Õppematerjali eesmärk on olla mänguline ning huvitav viis sihitise kasutamise õppimiseks. Veebilehel on näitena kasutatud lauset „Mia sööb melonit“, kus sihitis on sõna „melon“. Õiget sõnavormi kasutama õpetatakse läbi koomiksite, kus koomiksis edasi saamiseks on vaja täita eelnevas osas tekstis olev lünk korrektselt.

Tegu on huvitava ning unikaalse lähenemisega õppematerjalile. Pika teksti ning reeglite asemel on lühikesed koomiksid ning lünkade täitmisel on valikuks mõned sarnased variandid, mis kontrollivad inimese keeletunnetust. Veebirakendus on kaasaegne lähenemine keele õppimisele ning sarnaseid lahendusi on kindlasti veel ja tuleb ka tulevikus juurde.

3. Andmetöötlus

See lõputöö kasutab peamise andmeallikana korpust EMMA ning sealt saadavaid tekste. Korpust pole varem kasutatud andmeallikana veebiõppeprogrammi loomiseks, seetõttu oli tekstide valik skoobi piiramiseks väga spetsiifiline. Rõhk oli 12. klassi lõpukirjanditel, kus peaks õpilaste õigekirja- ning keeletunnetus olema kõige parem.

Korpusest saadud tekste töödeldakse, mille tulemusel saadakse kätte vigased laused ning sõnad. Proovitakse leida ka korrektseid versioone nendele sõnadele, kasutades TartuNLP Neurotõlget ning juba manuaalselt parandatud vigu.

Andmete kuvamiseks valmis ka veebirakendus[1], kus saab neid kirjavigasid parandades testida oma keelelisi oskuseid. Tegu on täispinu rakendusega, kus on andmebaas, tagasüsteem ning eessüsteem. Andmebaasis olevad andmed on saadud tekstikorpuse andmete töötlemisel ning leitud vigade käsitsi parandamisel.

3.1. Andmetöötlus

Andmetöötluses kasutatud vigaste sõnade andmed tulid EMMA korpusest XML-failina, mille töötlemiseks kasutati Pythonit ning selle teeke. Andmeid töödeldes leiti vigasid sisaldavad laused, otsiti need vead üles ning siis võimalusel leiti vigastele lausete sõnadele või sõnapaaridele õiged versioonid. Seejärel salvestati töödeldud andmed CSV-faili.

Tekstid, kust vigasid otsiti, paiknesid ühes suures XML-failis, kus iga rida oli ühe teksti üks lause. Lause oli ja elementide vahel ning esimese, alustava elemendi sees oli allalaetud faili info. Lausetes leiduvad vead olid välja toodud eraldi elementidena, kus igal veatüübil oli oma element. Veatüüpide elemendid võisid sisaldada vahest ka teiste veatüüpide elemente.

Õigekirjavead olid märgitud elementidega <v_oigekiri_p…>…</v_oigekiri_p…>. Elemendi p tähe taga olev number näitas, milline parandaja vea üles märkis. Leidus ka teisi vigu näitavaid elemente, näiteks stiili- ja faktivead, mida selle töö käigus ei kasutatud, sest töö rõhk oli kirjavigadel.

All on välja toodud, kuidas näeb välja üks lause XML-failis, kuidas see on üles ehitatud ning kuidas eelnevalt mainitud lisainfo välja näeb:

[1] https://viibur.github.io/project-fe/

<p aasta="2002" aine="Eesti keel emakeelena" allikas="Innove / HARNO" emakeel="eesti" klass="12" KOV="" liik="Eksami kirjutamisosa" lisaja="Tartu Ülikool" maakond="Ida-Viru" sugu="naine" kood="100010" tekstiID="2616">Kahjuks oli Raskolnikovi idee <v_oigekiri_p1>läbikaalutlemata</v_oigekiri_p1> ning naiivne ja mõistagi määratud nurjumisele.

Kõige raskem osa andmete töötlemise puhul olid laused, mis sisaldasid mitmeid vigasid, sest valminud algoritm leidis neist ainult esimese ning seejärel puhastas kõik märgendid töödeldavast lausest. Samamoodi tegi ta ka teise märgendi puhul, mistõttu tekkisid .CSV faili erinevad vead samade lausetega ning nende vigastele sõnadele korrektsete määramisel tuli lauses parandada ka teine kirjaviga.

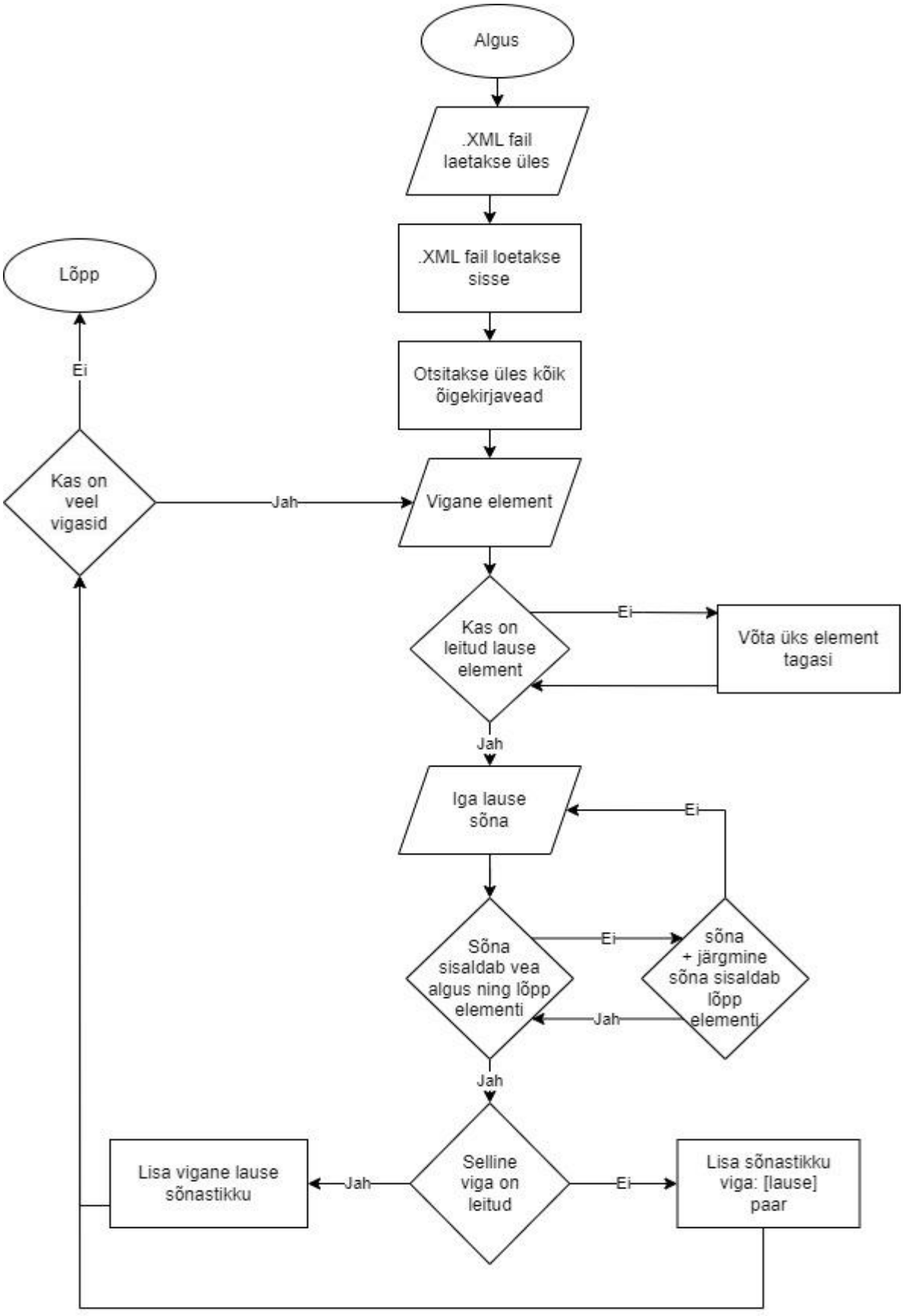
3.1.1. Tekstidest vigaste lausete leidmine

Vigaste lausete leidmiseks oli kasutuses eelmainitud Python. Seda programmeerimiskeelt kasutatakse väga palju andmete kogumisel, lugemisel, töötlemisel ning analüüsimisel (Marchand, 2022). Pythonil on mitmeid teeke, mis on loodud just sellel eesmärgil, näiteks Pandas, NumPy, BeautifulSoup, MatplotLib ja paljud teised.

Failist vigaste lausete leidmiseks kasutati teeke nagu BeautifulSoup, Levenshtein ning regulaaravaldiste teeki re. Pythoniga sai failist kõik read sisse lugeda ning peale seda sai edasi anda tekst BeautifulSoup teegile. Selle abil sai leida kõik õigekirjavigade elemendid ning need kokku koguda. Andmetöötlus töötas elementide põhiselt, et leida üles ka laused, kus oli mitu viga. Selline lähenemine võimaldas saada mitme veaga lausest kätte kõik vead, arvesse võtmata teisi lauses leiduvaid vea märgendeid.

Peale vigaste elementide leidmist sai iga element läbi käidud ning sealt otsitud lause kõige välimine element, milleks oli eelnevalt mainitud element. Seejärel tuli lause välja eraldada, teha see sõnade loendiks. Loend käidi järk-järgult läbi ning sealt leiti üles vigase sõna/sõnapaari algus element <v_oigekiri_...> ning kontrolliti, kas ka lõppelement </v_oigekiri_...> on samas sõnas. Kui on, siis mindi edasi, kui mitte, siis võeti järgmine sõna ja kontrolliti uuesti. Kui ka siis ei leidunud lõpp elementi, siis mindi edasi, sest selliste lausete puhul ei saanud ära kasutada peatükis 3.1.4 välja toodud Levenshteini kauguse meetodit.

Vea leidmisel sai lause uuesti kokku panna, puhastada see elementide markeritest ning lisada koos leitud veaga sõnastikku. Sama viga uues lauses leides lisandus vigane lause sõnastikus vastavale veale kuuluvasse loendisse (vt j*oonis 1*).



Joonis 1. Vigade leidmise algoritm

3.1.2. Veebimaterjalis kasutatud EVKK vigased laused

Esialgsele versioonile lisandusid ka EVKKst saadud vead koos eelmise ning järgmise sõnaga, millest jõudsid kasutusse kõik, mille sagedus oli üle 80. Selle piiranguga jäi alles 21 viga, mis tuli lauseteks teha. Kuna oli antud ainult vigane sõna ning olemasolul ka eelmine või järgmine sõna, siis tuli neile sõnadele manuaalselt laused juurde teha. Manuaalselt juurde lisatud laused olid lühikesed ja kerged, et lisada töödeldud lausetele juurde kergemaid näiteid.

Mõlema faili vead said parandatud manuaalselt, kasutades sõnade korrektsete versioonide leidmiseks sõnaveebi ja EKI-t. Need manuaalselt parandatud algse versiooni laused on kasutuses ka lõputöö käigus õppematerjaliks valminud veebilehe andmebaasis.

Andmeid läbides eemaldati sealt lauseid, mis õppematerjalis ebasobivad oleks. Sinna alla kuulusid poliitilised ning ideoloogilised laused, liiga pikad laused ning laused, mille ülesehitus oli puudulik. Eemaldatud said ka laused, mille mõte või viga jäi segaseks. Allpool on mõned näited lausetest, mis ei jõudnud materjali.

- Praegust poliitikat ja teed jätkates ei pruugi meil varsti enam üldse isamaalisus tunnet olla ja sellest mäletavad vaid vähesed.
- Käed ära ning raamatukaan pannakse minu muusika peale.
- Olgu selleks, siis 3D, 4D või 5D film. – Kus veana oli märgitud „3D“
- // „Elu tuleb elada nii, et oleks millepärast surra.“ (M.

Olles välja filtreerinud 325 lauset, viga ning korrektset sõna, teisendati sõnastik JSON failiks. JSON formaadis andmed salvestati kasutades tagasüsteemi POST-päringu lõpp-punkti */muuda_andmed* andmebaasi, et kasutajaliides saaks neid kasutada.

3.1.3. Korrektsete versioonide manuaalselt lisamine

Peale vea sidumist lausetega, kus need vead esinevad, tuli leida võimalikult paljudele vigadele ka korrektsed versioonid. Selleks on mitmeid viise, millest lõputöö käigus valminud vigade töötlemise algoritm kasutab kolme. Nendeks on:

- Juba parandatud sõnade sagedaseim korrektne versioon
- TartuNLP Neurotõlke eesti-eesti tõlge
- Lahku kirjutatud sõnade kokku liitmine.

Manuaalselt parandatud vead on salvestatud andmebaasi ning neid saab sealt pärida. Päring tagastab nimekirja vigadest ning nendele vigadele kõige sagedasema korrektse vaste. Kui

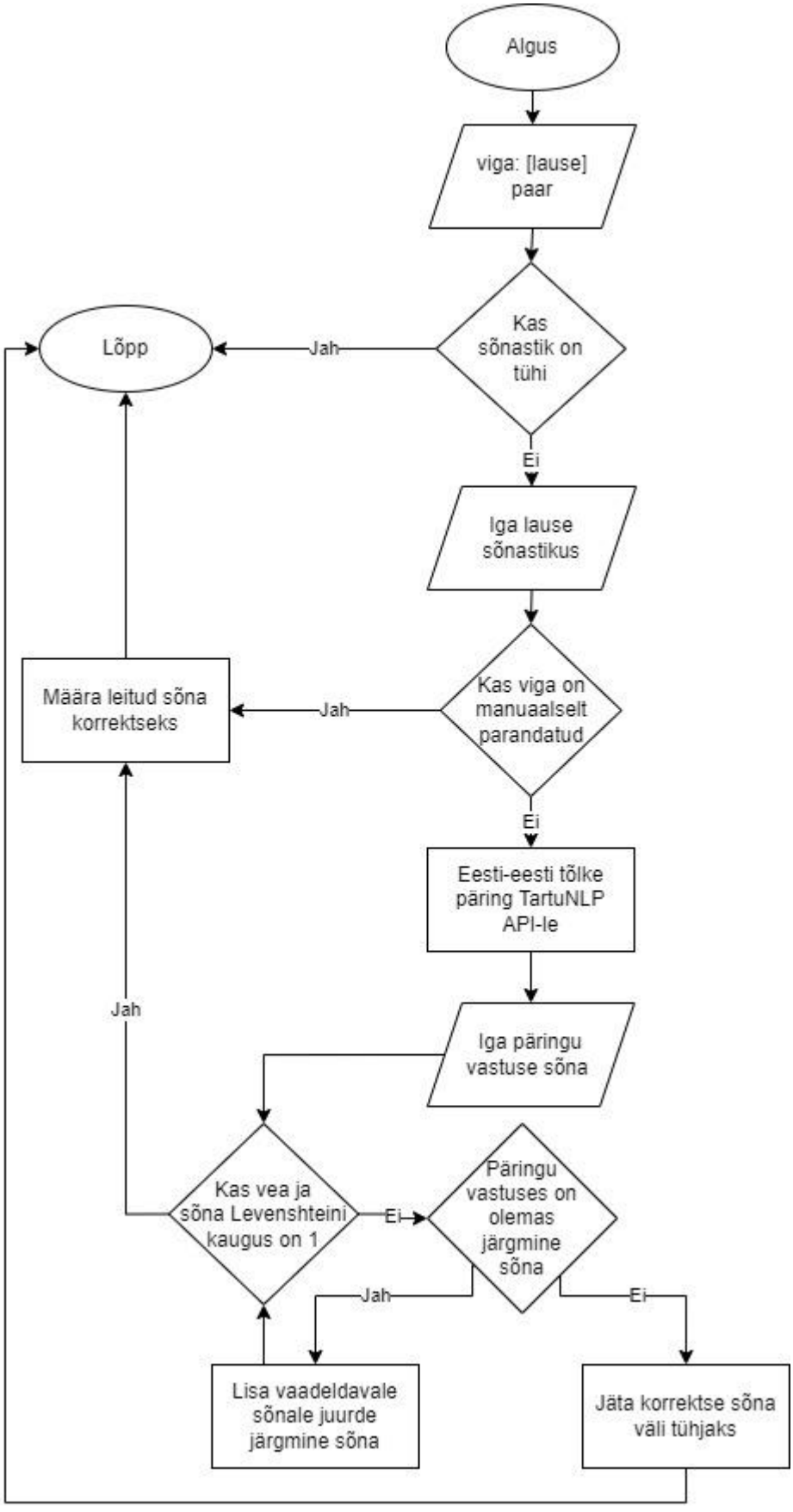
algoritm jõuab kohta, kus on vaja kontrollida, kas selline vigane sõna leidub juba parandatud sõnade nimekirjas, siis otsitakse, kas sõna leidub. Leidmise korral määratakse see sõna korrektseks ning minnakse järgmise vea juurde. Kui viga pole varem parandatud, siis minnakse edasi TartuNLP Neurotõlke juurde.

3.1.4. TartuNLP Neurotõlke abil parandamine

Vigastele sõnadele leiti korrektseid versioone ka kasutades TartuNLP Neurotõlke tõlkemootorit. See on Tartu Ülikooli teadlaste poolt välja töötatud masinõppe mudel, millega saab soome-ugri tekste omavahel tõlkida (TartuNLP Neurotõlge, s.a.). Lõputöö kirjutamise ajal on selliste keelte valik Google'i tõlkemootori puhul veel puudulik.

Kui tõlkemootori loojad olid mudelid välja treeninud, siis tuli välja, et on võimalik parandada grammatiliselt ebakorrektseid lauseid (TartuNLP Neurotõlge, s.a.). Seda omadust sai ära kasutada vigaste lausete korrektseteks muutmisel ning sealt vigase sõna põhjal korrektse leidmisel.

Korrektse versiooni leidmisel oli põhirõhk Levenshteini kaugusel. Algoritmi kasutatakse sõnade, lausete, tekstide sarnasuse kirjeldamiseks, kasutades tähtede lisamist, muutmist või eemaldamist. Leides kas kauguse sõnade või sõnapaaride (vt joonis 2) vahel on 1, sai otsustada, kas vaadeldav sõna on potentsiaalselt vigase sõna parandatud versioon.



Joonis 2. Korrektsete sõnade leidmine

Kauguseks sai valitud 1, sest enamik kirjavigasid olid kokku- ja lahkukirjutamisel ning suur osa tuli ka tugevate/nõrkade tähtede ning i/j sassi ajamisel. Seetõttu oli kaugus 1 parim viis õigete vastete leidmisel, vältimaks valepositiivseid tulemusi.

Seda valikut kinnitas ka manuaalselt parandatud sõnade ning nende vigaste vastete võrdlemine, kus 94,7% ajast oli selliste paaride Levenshteini kauguseks 1.

Probleemseks kohaks osutus ka tõlkemootori poolt parandatavad stiilivead, mis muutsid lauseid parandamisel kohati väga erinevaks originaalsetest lausetest. Andes ette lause:

Mina soovib uut jalgratast.

oleks vigane sõna „soovib“, mille õige versioon oleks „soovin“. Tõlkemootor aga näeb, et lause ei ole ainult grammatiliselt, vaid tema jaoks ka stiililiselt vigane. Seega parandatud lause on:

Ma tahan uut jalgratast.

Seda lauset läbi käies algoritm ei leia vastava Levenshteini kaugusega vastet ning jätab salvestatavas failis korrektse sõna lahtri tühjaks.

Katsetati ka lausete asemel ainult üksikute vigase sõnade saatmist, aga paljud vigased sõnad on teistsuguses kontekstis korrektsed, näiteks lausele:

Need, kes on selle saavutanud, ei taha minna õpitust tasemest madalamale töökohale,

kus vigasele sõnale *õpitust* andis TartuNLP Neurokõne API vastuseks:

Need, kes on selle saavutanud ei taha minna õpingutest madalamale tööle.

Kuigi korrektne versioon vigasest sõnast oleks *õpitud*, ning andes ainult vigane sõna, parandas tõlkemootor selle kui *õppimine*. Andes ainult üks sõna ette, tekkis väga palju valepositiivseid tulemusi ning seega sai otsustatud terve lause korraga ette andmise kasuks.

Võrreldes TartuNLP vastuseid manuaalselt parandatud versioonidega, parandas otsingumootor korrektselt 40,3% ning 56,9% ajast jättis vastuse tühjaks. Seega 97,2% ajast lisati vigasele sõnale kas korrektne versioon, või jäeti lünk tühjaks, et manuaalselt parandada. Valepositiive tekkis 2,8%, ehk 9, mida on 325 lause peale võrdlemisi vähe.

3.1.5. Tähtede asendamine sõnade parandamisel

Proovitud sai ka üksiktähtede automaatselt muutmine. Teades, et sõnad on vigased ning et enamus vigadest on ühe tähe kaugusel korrektsetest, sai proovitud üksikute tähtede muutmise abil parandamist.

Üks katsetatud variant oli leida üles tugev/nõrk täht ning muuta see vastupidiseks, et ära parandada koht, kus sõnas viga tehti. Sellisel vigade parandamise meetodil oli potentsiaali, sest enamus vigasid, mis ei olnud kokku-ja lahkukirjutamise vead, olid vahetusse läinud kptgbd vead. See lähenemine tõi endaga aga kaasa väga palju valepositiivseid tulemusi. Suurimaks probleemiks olid kokku kirjutatud sõnad, mis tegelikult lahku käivad.

Üheks selliseks probleemseks sõnaks oli *läbikaalutlemata*, millel on mitu potentsiaalselt ortograafiaviga. Teadmata, et tegemist on kokku- ja lahkukirjutamise veaga, võis algoritm muuta sõna üheks järgnevateks:

- *läbikalutlemata* ehk *aa -> a*
- *läpikaalutlemata* ehk *b -> p*
- *läbikaaludlemata* ehk *t -> d*

Kõik neist on ebakorrektsed ning vigaste sõnade parandaja töömahu vähendamise asemel tekitaksid sellised veaparandused hoopis tööd juurde, sest lisaks tuleks välja praakida väga paljud valepositiivseid tulemusi.

Tähtede automaatselt parandamine töötas üpriski hästi vigaselt lahku kirjutatud sõnade puhul. Selliste sõnade puhul on parandaja tihti märkinud vigaseks mõlemad sõnapooled, seega neid on lihtne üles leida ning kokku liita. Probleemid tekkisid siin nendel lausetel, kus parandaja või märgendaja on vigaseks märkinud ainult ühe sõna kahest. Selliseid lauseid esines töös õnneks vähe.

3.1.6. Andmete salvestamine

Peale andmete töötlemist ning võimalike korrektsete sõnade leidmise muundati andmed inimesele muudetavaks ning loetavaks. Korretkne sõna, vigane sõna ning lause salvestati Pythoni klassi objektis, mis omakorda lisati objektide loendisse.

Pythoni klassidele saab lisada meetodi „__iter__“ (vt joonis 3), mida kasutab ära teek *csv*, ning millega saab lugeda ning salvestada andmeid CSV-failidesse (Vu, 2021). Loend objektidega salvestati UTF-8 kodeeringuga faili „vead.csv“, mida saab avada erinevate programmidega. Paremini töötab aga LibreOffice, sest programmis saab määrata kodeeringu ja eraldajad.

```
class Viga:
  def __init__(self, korrektne, viga, lause):
    self.korrektne = korrektne
    self.viga = viga
    self.lause = lause

  def __iter__(self):
        return iter([self.korrektne, self.viga, self.lause])
```

Joonis 3. Pythoni klass andmete salvestamiseks

4. Veebirakendus

Andmetöötlusest saadud tulemusi kasutatakse, et kasutajad saaksid õppida EMMAst saadud lausete peal. Selleks valmis lõputöö käigus täispinu veebirakendus PostgreSQL, Java ja Angulari baasil. Rakendus on ka läbi 3 teenuse kasutatav pilves, lokaalselt midagi üles panemata. Valmis ka docker-compose.yml fail, millega saab rakendust lokaalset jooksutada.

4.1. Andmebaas

Peale andmete töötlemist ning nende CSV-faili salvestamist on need võimalik manuaalselt läbi vaadata ning korrigeerida. Google Colabis saab ka parandatud CSV-faili muuta tekstifailiks, mis sisaldab .CSV faili sisu JSON formaadis. Selle faili sisu saab tagasüsteemile saata kasutades POST-päringu lõpp-puntki */muuda_*andmed, mis lisab andmed vastavasse lokaalsesse või pilves olevasse andmebaasi.

Docker-compose failiga saab kohalikku arvutisse luua töötava projekti, mis kasutab PostgreSQL andmebaasi, kus on üks tabel andmete jaoks. Tehnoloogia on avatud lähtekoodiga relatsioonandmebaas, mis on eksisteerinud ja pideva arendustöö all olnud üle 35 aasta (PostgreSQL, 2023). PostgreSQL on hästi dokumenteeritud ning töökindel, sellele on ka Spring Bootis liides, millega on tagasüsteem ühendamine andmebaasiga lihtne ning probleemide korral on väga palju abimaterjale.

Andmebaasi tabelis on 6 tulpa korrektsete ja vigaste sõnade, lausete ning id ja vigase sõna valesti parandamise sageduse jaoks. Kõik väljad on salvestamisel kas kohustuslikud (sõnad ja laused) või täidetakse automaatselt (id, sagedus, raporteeritud). Andmete lisamisel määratakse sagedus nulliks ning raporteeritud false'ks.

Andmebaasi salvestamisel piisab korrektse ja vigase sõna ning seda sõna sisaldavast lausest. Kasutajaliides pärib õpiprogrammi jaoks kõik andmed ja andmetöötluse ajal saadetakse päring vigane korrektne sõnade paaridele. Kasutajaliidese jaoks sorteeritakse andmed valesti pakkumise sageduse järgi kahanevalt, et aidata luua raskustaseme süsteem kasutajatele õppimise efektiivsemaks tegemiseks.

4.1.1. Neon

Lõputöö tarbeks valminud ortograafiavigade parandamiseks mõeldud veebiõppematerjal kasutab andmete hoiustamiseks Neoni ning selle tasuta taset. Neon on veebileht, kus saab luua ja hallata serverita PostgreSQL andmebaase (What is Neon?, 2023).

Veebisaidil saab luua tasuta versioonis oma projekti, valida PostgreSQL versioonide 14 ja 15 vahel ning maailma osa, kus andmebaas püsti pannakse. Projektile genereeritakse andmebaas ning link, millega sinna ühendada. Saidil on võimalik luua tabelid oma valitud veergude ja piirangutega ning tagasüsteemile päringuid tegevale osale URL-i anda, millega andmebaasi ühendada.

4.2. Tagasüsteem

Tagasüsteem kasutab Javat, mis on populaarne keel, olles mitmekülgne ja skaleeruv lahendus, mis on üle mitme platvormi ühilduv (Winter, 2023). Keele esimene versioon avaldati 1995 ning uusi versioone ning täiendusi lastakse välja ka tänapäeval. Java on tänapäeval kasutusel nii eessüsteemides, Android rakendustes ja pilveteenustes, aga selle objektorienteeritud lähenemine on väga mugav just tagasüsteemide arendamisel (What Is Java Used For?, 2022). Keel on ka väga populaarne ning selle kohta on lihtne leida materjali, seda kasutatakse ka mitmes Tartu Ülikooli enda ainetes.

Andmebaasist andmete kasutajaliidesesse saamiseks on kasutusel Java raamistik Spring Boot. See on avatud lähtekoodiga Spring raamistiku peale ehitatud moodul, mis aitab kaasa Javas rakendusliideste loomisel. Sisaldades endas mitmeid erinevaid sisse ehitatud ning lisatavaid pluginaid, mis vähendavad arendaja töökoormust (What is Java Spring Boot?, s.a.).

Spring Bootis on lihtne luua rakendusliideseid, mida saab kasutada kasutajaliidese ja andmebaasi omavahelisel suhtlusel. Kasutades ära erinevaid mooduli annotatsioone (näiteks @Getter, @Setter või @Data), saab vähendada stereotüüpset koodi, mida peaks muidu iga klassitüübiga vaikimisi looma.

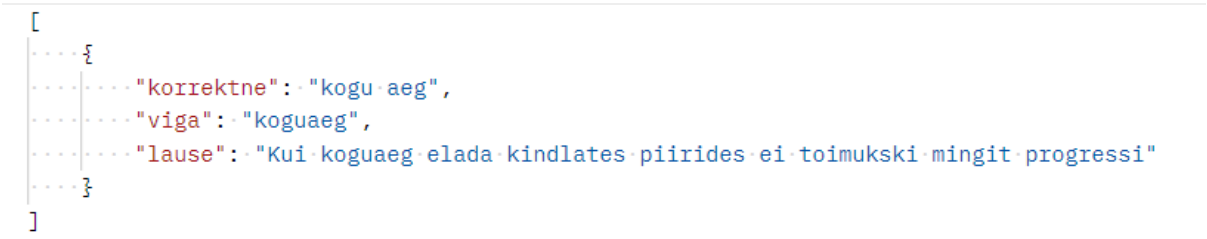
Annotatsioone saab ka ära kasutada, et hõlpsalt luua ühenduse andmebaasiga, lisada loogika ning teha avalikuks URL-i, kuhu saab päringu teha. Sedasi toimivad töödeldud andmete lisamine, kasutajaliidese päring ning juba parandatud vigane korrektne paaride tagastamise.

4.2.1. Andmete lisamine ning täiendamine

Töödeldud andmeid saab lisada andmebaasi läbi tagasüsteemi avatud lõpp-punkti. URL oleneb teenusest, mida kasutatakse tagasüsteemi üleval hoidmiseks, aga POST-päringu saab koos lisatava JSON kujul andmestikuga saata */muuda_andmed* lõpuga lõpp-punkti.

Tagasüsteem oskab lugeda JSON-i korrektsel kujul (vt joonis 4) sisse ning selle sisu muuta Java objektideks, mida andmebaasi salvestada. Baasi salvestades lisanduvad juurde *id*, mis

aitab andmetel järge pidada ning unikaalsust tagada ja *sagedus*, mida kasutajaliides kasutab raskustasemete loomisel.

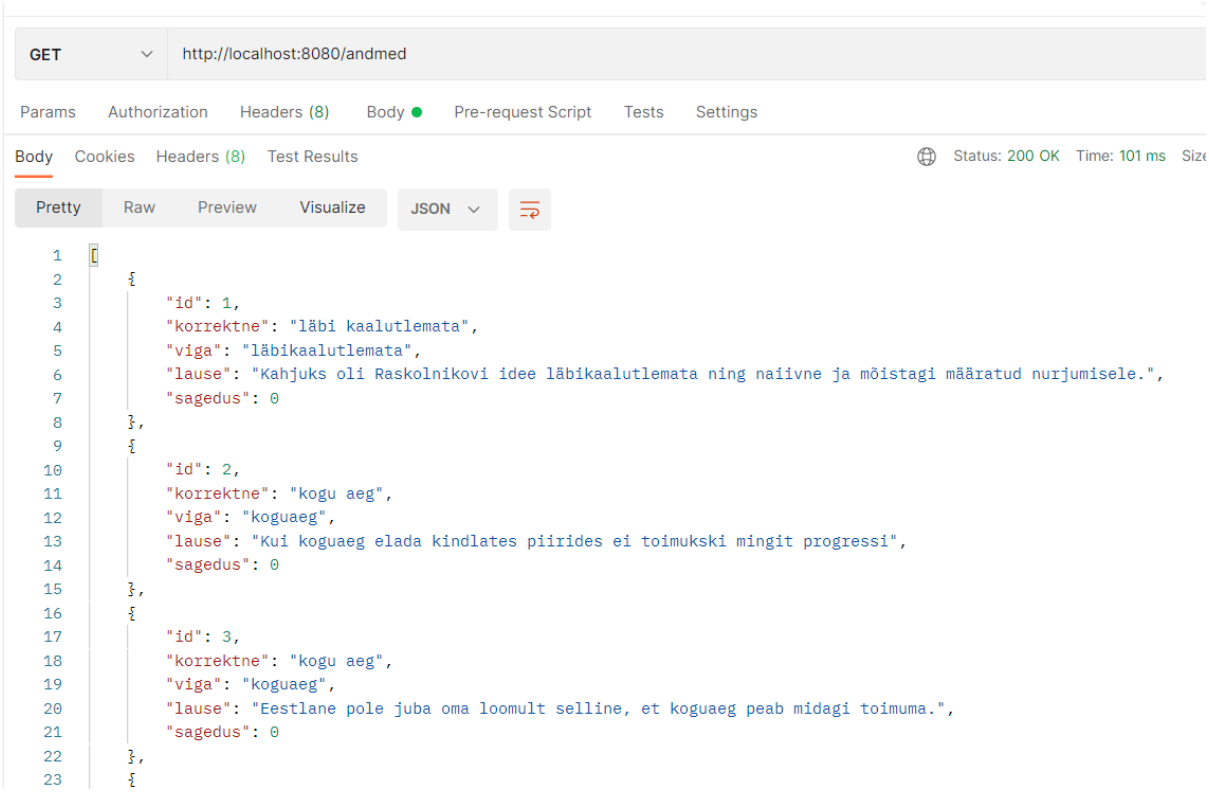


Joonis 4. Andmebaasi salvestatavate andmete JSON kuju näide

Juhul, kui kasutaja parandab kasutajaliideses vigase sõna ebakorrektselt, suurendatakse selle sõna sagedust ühe võrra. Need andmed kogutakse kokku ja kui lehekülg suletakse saadetakse kõik valesti vastatud sõnad tagasüsteemi, kus nende sagedus andmebaasis *id* põhjal uuendatakse.

4.2.2. Andmete pärimine

Andmeid saab pärida tagasüsteemist GET-päringuga lõpp-punktile */andmed*, mis tagastab JSON kujul loendi objektidest (vt joonis 5). Päritud andmed on sorteeritud kahanevalt sageduse järgi ning ei sisalda lauseid, mida kasutajad on raporteerinud.



Joonis 5. Kasutajaliidese päritavate andmete JSON kuju

4.2.3. Paaride “vale-korrektne“ pärimine

Andmetöötluse jaoks valminud Google Colabi leht saab GET-päringuga tagasüsteemi lõpp-punktist */paarid* pärida vigaste sõnade ja nende kõige sagedasemate korrektsete versioonide paarid. Vigadele leitakse kõige sagedasemad parandatud variandid, sest osadele vigadele võib potentsiaalselt leiduda mitu erinevat korrektset sõna.

Kõigepealt tehakse andmebaasi päring, millega tagastatakse kõik seal olevad objektid. Need andmeobjektid grupeeritakse vigaste sõnade järgi nii, et igale vigasele sõnale oleks sõnastikus vastamisse pandud loend objektidest, kus veaks on see vigane sõna.

Sõnastik käiakse võtme haaval läbi ning iga vigase sõna kohta läbitakse vastav loend ja leitakse kõikide korrektsete sõnade esinemissagedus. Peale loendi läbimist käiakse läbi korrektse versiooni ja sageduse sõnastik ja leitakse kõige sagedasem versioon, mis lisatakse päringuga tagastatavasse loendisse.

4.2.4. Raporteeritud sõnad

Läbi eessüsteemi raporteeritud sõnu saab pärida lõpp-punktist */raporteeritud*, mis tagastab GET-päringuga kõik andmebaasis olevad andmed, mille *raporteeritud* väli on määratud true’ks.

Sellele lisaks on ka POST-päringu lõpp-punkt */kustuta_raporteeritud*, mis kustutab kõik päringuga kaasas olevad objektid, mille *raporteeritud* väli on true. Peale lausete kustutamist muudetakse kõik andmebaasis olevate andmete *raporteeritud* väli false peale.

4.2.5. Railway

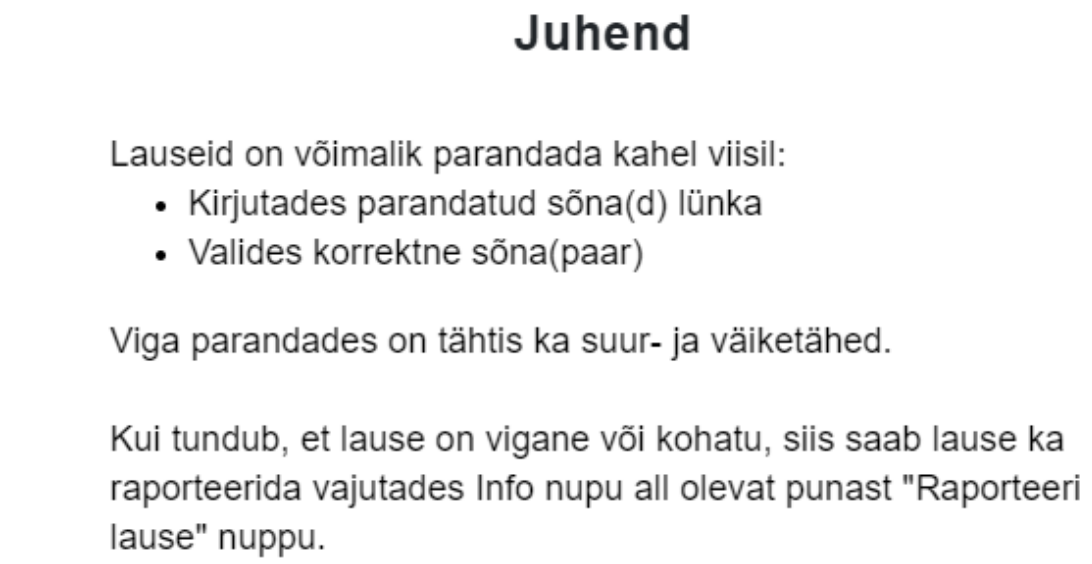
Tagasüsteemi majutatakse veebi juurutamise platvormil Railway, mida saab ühendada GitHub projektiga ning seda kuus 500 tundi või 5$ arvutusjõu eest tasuta kasutada. Projektile saab määrata ka muutujaid, nende seas andmebaasi URL-i.

Läbi muutujate on tagasüsteem seotud Neonis oleva andmebaasiga. Sarnaselt on läbi loodud lõpp-puntkide ja Railway poolt genereeritud URLi tagasüsteemi funktsionaalsused kasutavad ka eessüsteemis.

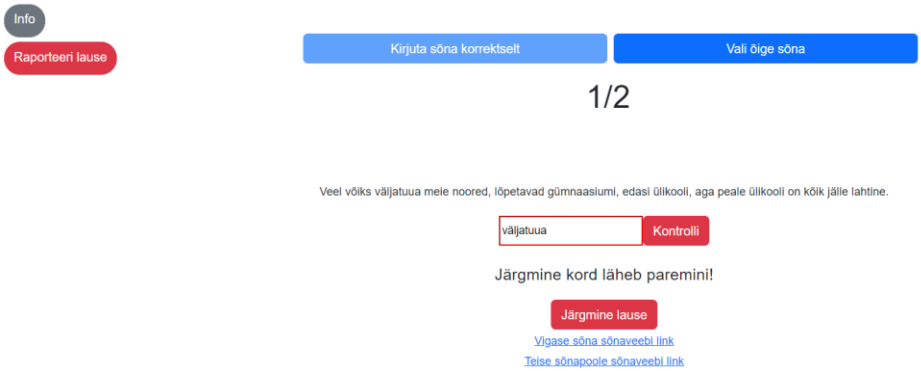
4.3. Eessüsteem

Kasutajaliides valmis kasutades Javascript raamistikku Angular, mis on raamistik ning arendusplatvorm veebilehtede jaoks. Raamistiku sihtrühm on kasutajad, kellel on vaja luua efektiivseid ühe-lehe veebilehtesid. Angulari lehed on loodud kasutades HTML-i, CSS-i ja JS-i, eriti selle staatilist alamkeelt TypeScript (Introduction to the Angular docs, 2023).

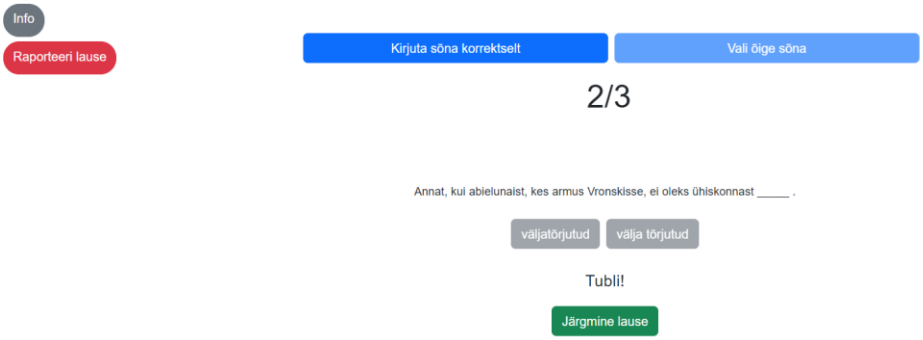
Valminud veebilehel on juhend (vt joonis 6), kaks erinevat vigade parandamise viisi (vt joonis 7 ja 8). Esimene viis on leides lausest vigane sõna ning kirjutades see ümber korrektselt. Teine viis on valida lünka sõna, kus valikutes on nii vigane kui korrektne sõna.



Joonis 6. Veebilehel olev juhend



Joonis 7. Kirjuta sõna korrektselt



Joonis 8. Vali õige sõna

Sõnad päritakse tagasüsteemi kaudu andmebaasist, ning seejärel sorteeritakse vastavalt lausete eksimuse sageduse abil kümnesse võrdsesse raskustaseme järku. Selline algeline raskustasemete süsteem võimaldab ka inimestel, kelle keeleline oskus on parem, leida enda jaoks järjepidavalt lauseid, mis ka nende teadmisi laiendaks.

Juhendit on võimalik alati välja tuua kasutades vasakul üleval nurgas olevat nuppu „Info“. Vastates saab ka tagasisidet, kas kirjutatud või valitud sõna on korrektne (vt joonis 8). Peale valiku tegemist on võimalik edasi minna järgmise sõna juurde.

Lauseid on võimalik ka raporteerida vajutades joonistel 7 ja 8 näha olevat nuppu „Raporteeri lause“. Sellega saadetakse tagasüsteemi lause id, millega andmebaasis *raporteeri* väljale vastav muudatus tehakse.

Vastates valesti antakse kasutajale lingid sõnaveebi, et saada rohkem infot vigase sõna kohta ning lihtsamaks teha reeglite otsimise. Lisaks saab ka peale valesti vastamist liikuda edasi järgmise sõna juurde, mis aitab vältida seda, et kasutaja ei leiagi õiget vastust. Valesti vastamine suurendab ka lause eksimise sagedust andmebaasis, et järgnevatele kasutajatele luua täpsem raskustaseme süsteem.

4.3.1. GitHub Pages

Eessüsteem on üleval kasutades populaarse koodi repositooriumi GitHub teenust GitHub Pages. Tasuta versioonis saab ühte avalikku projekti määramata aja majutada (Websites for you and your projects., 2023). Kasutades Angulari käsurea liidest, saab kahe käsuga oma Angulari eesüsteemi repositooriumi üles laadida ning seal GitHub Pages paigalduse jooksutada (Jahirhussain, 2022), mis teeb muudatuste tegemise ning nende tootmiskeskkonda jõudmise kiireks ja lihtsaks.

5. Docker

Lõputöö käigus valminud veebi õppematerjali saab üles seada ka enda arvutis ning oma andmetega. Selleks on kasutusele võetud Dockerfile'id ja docker-compose.yml, millega saab korraga üles seada kõik vajalikud Docker konteinerid.

Docker on tehnoloogia, mis on loodud selleks, et inimesed saaksid luua, jagada ja jooksutada oma programme, ilma et tuleks probleeme erinevate arvutite tehnoloogiate ja versioonidega (Use containers to Build, Share and Run your applications, s.a.). Konteinerid on standardiseeritud ja igaüks saab enda programmid dokeriseerida, üles laadida ja jagada. Pilves on ka mitmete erinevad tehnoloogiad, millele on hea dokumentatsioon ning mida väga paljud erinevate rakendusliideste arenduses kasutavad (The Good and the Bad of Docker Containers, 2022).

Konteinerite loomiseks tuleb nii tagasüsteemist kui ka kasutajaliidesest luua Dockerfile'id. Need tuleb ümber nimetada, versioon lisada ning üles laadida. Peale seda on need avalikkusele kättesaadavad ning igaüks saab neid alla tõmmata ja tööle panna.

5.1. Andmebaas

Andmebaasi ning loogilise draivi, kus neid andmeid talletatakse, luuakse kõige uuema postgres versiooniga. Juurde antakse kataloog, kus andmeid hoida ning lõpp-punkt, kuhu päringud teha.

Joonisel 9 on näha väljad loogilise draivi loomiseks ning kataloog, kus andmid talletatakse, mis versiooni PostgreSQLi kasutatakse, konteineri nimi, mis lõpp-punkte lahtiseks teha ja kuidas neid lahtiste ja kinniste vahel kaardistatud on. Joonisel on näha ka keskkonnamuutujad, *environment* all on:

- Autentimisel kasutatav kasutajanimi
- Autentimisel kasutatav parool
- Andmebaasi enda nimi
- Kataloog, kus andmeid talletatakse

```yaml
volumes:
  postgres-volume:

services:
  db:
    image: 'postgres:latest'
    container_name: db
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=admin
      - POSTGRES_DB=postgres
      - PGDATA=/var/lib/postgresql/data
    volumes:
      - postgres-volume:/var/lib/postgresql/data
    ports:
      - 5432:5432
    expose:
      - 5432
```

Joonis 9. Docker-compose.yml faili andmebaasi osa

5.2. Tagasüsteem

Tagasüsteem, millega rakendusliides üles ehitatakse pärib Dockeri pilvest kindla projekti ning versiooni, ehitab selle, annab nime, annab välja lõpp-punktid, kuhu päringuid teha, näitab et sõltub andmebaasist ning et kõigepealt peab andmebaasi üles ehitama (vt joonis 10). Keskkonna muutujateks on:

- Logides SQL päringute näitamine
- Pärib andmebaasi andmed
- Annab ette, mis andmebaasiga tegu on
- Annab ette andmebaasi URLi. db:5432 on andmebaasi konteineri nimi ning avalikustatud lõpp-punkt
- Andmebaasiga autentimise kasutajanimi

- Andmebaasiga autentimise parool

```yaml
be:
  image: 'jqreijin/project-be:v2'
  build:
    context: .
  container_name: app
  ports:
    - 8080:8080
  depends_on:
    - db
  links:
    - db
  environment:
    - SPRING_JPA_SHOW-SQL=true
    - SPRING_JPA_HIBERNATE_DDL_AUTO=update
    - SPRING_DATASOURCE_PLATFORM=postgres
    - SPRING_DATASOURCE_URL=jdbc:postgresql://db:5432/postgres
    - SPRING_DATASOURCE_USERNAME=postgres
    - SPRING_DATASOURCE_PASSWORD=admin
```

Joonis 10. Docker-compose.yml tagasüsteem lõik

5.3. Kasutajaliides

Kasutajaliides ehitatakse sarnaselt tagasüsteemiga üles. Ette on antud konteineri nimi, peale seda pildi nimi ja versioon, mis Dockeri pilvest tõmmatakse. Peale neid on argumendina kaasa antud tagaosa URL, millest andmeid pärida ning seejärel avatud lõpp-punktid, millel avaneb leht veebibrauseris. Jooksutamiseks on vaja, et andmebaasi konteiner töötaks ja kasutajaliidese konteiner on lingitud tagasüsteemi konteineriga (vt joonis 11).

Joonis 11. Docker-compose.yml kasutjaliidese osa

6. Küsitluse tagasiside

Veebirakenduse testimiseks sai läbi viidud ka küsitlus, millega koguti tagasisidet veebirakenduse ning seal olevate lausete kohta. Küsitlus uuris vastajate arvamust veebilehe juhendi kohta, aga ka rakenduses kasutuses olevatest lausete, vigade ning vastamisviiside kohta. Küsitlusele vastas 8 inimest, sealhulgas 2 õpetajat. Väikse vastajate arvu tõttu oli küsitlus ning analüüs kvalitatiivse rõhuga.

6.1. Tagasiside

Tagasiside oli üldiselt positiivne, tõsteti esile lausete mitmekesisust, mis tähendab, et igale keeleoskuse tasemele on midagi. Oli ka kriitikat ning soovitusi, mida teha paremini ning teisiti.

Rakenduse põhimõtteid seletav juhend oli kõigile vastanutele arusaadav. Kaks erinevat õppeviisi oli ka õpetajate arvates tore lisand, kus kahe sõna vahel valimine sobib rohkem algajale keeleõppijale. Samas öeldi, et mõnda lünka sõna valimine oli raske, eriti kokku- ja lahkukirjutamise valikutes.

Kuus inimest tõi välja lausete raskustasemete suvalist esitusjärjekorda, kus väga rasked ja väga kerged laused tulid üksteise järel. Lausete parandamisel eksimine tõstab selle lause raskustaset ning järgmistel kordadel on see lause juba teiste raskemate seas. Seega peab kasutaja rohkem lauseid järjest korrektselt parandama, et nendeni jõuda.

Probleemse kohana toodi välja ka see, et osade vigade puhul on lubatud ka paralleelvormid. Üks vastanutest tõi välja näitena

t’ekkima <28: t’ekkida, tekin; 27: t’ekkida, t’ekkin>,

kus korrektse versioonina loeti sõna *tekin*, aga sobis ka *tekkin*. Selliseid paralleelvormidega rakendus ei arvestanud ning see on potentsiaalne edasiarendus võimalus.

Esile tõsteti ka see, et väga paljudes lausetes on stiilivigu, mis muudavad need kehvaks keele õppimise viisiks, eriti algajatele. Välja toodi veel soovitusi graafilise kasutajaliidese muutmiseks, näiteks juhendile lausete lõppu punktide lisamine või erinevatele nuppudele vahede lisamine, et veebirakendus vähem kokku surutud tunduks.

6.2 Edasiarendamise võimalused

Veebirakenduse arendamise käigus ning hiljem kasutajate tagasisidet analüüsides tulid välja mitmed probleemid ning murekohad, mida tulevased edasiarendused saaksid parandada.

Suurim takistus rakendust õppevahendina kasutamisel on kirjavead, mis ei ole seotud õigekirjaga. Selle alla kuuluvad stiili-, fakti- ning interpunktsioonivead, mille esinemine parandatavates lausetes muudab terve keele õppimise aspekti kvaliteedi kehvemaks. Seda saaks parandada, jättes välja kõik laused, mis sisaldavad teiste vigade märkimise elemente.

Teine probleem on kui vigase lause lahendamiseks on mitu korrektset versiooni. Töö käigus kasutatud andmestruktuur seda ei arvesta ning seetõttu ei olnud võimalik ühele lausele määrata mitut õiget vea parandusviisi. Kergeim lahendus oleks luua ühe tabeli asemel kaks, mis on omavahel üks-mitmele seoses. Selline lähenemine võimaldaks ühel lausel omada mitut korrektset versiooni.

Uued tööd saaksid ka implementeerida rohkem kui ainult ühte vea tüüpi, kasutades näiteks interpunktsioonivigasid. Võimalus oleks ka lasta kasutajatel pakkuda, kas lause sisaldab stiiliviga või mitte. Korpuses olevate märgenditega saab teha mitmeid erinevaid õppematerjale, kasutades ära fakti, et tekstid on kirjutatud gümnasistide poolt ning seega sisaldavad endas vigu, mida õppetöö käigus pole ära õpitud ja tehakse seetõttu ka koolivälselt.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärgina loodi algoritm, et leida EMMAs olevaid õigekirjavigadega lauseid, ning veebirakendus, et neid lauseid teadmiste kontrollimisel kasutada.

Algoritm loodid Google Colab keskkonnas, kus see võtab aluseks korpusest saadava .XML faili ning teeb sellest .CSV faili, kus on vead, vigased laused ja võimalusel ka potentsiaalne korrektne versioon veale. Õpiprogramm on Java ning Angular täispinu rakendus, mis kasutab andmebaasina PostgreSQLi. Rakendusse on sisse ehitatud kaks erinevat käsitsi vigade parandamise viisi ning vead on jagatud vastavalt nende vastu eksimise arvu järgi kümnesse erinevasse raskusastmesse. Veebirakendusest valmis ka Docker image, mida saab õpiprogrammi kas lokaalselt või ligipääsetaval serveril üles seada.

Viidi läbi ka tagasiside küsitlus, mille tulemused olid enamasti positiivsed, aga toodi välja ka mitmeid erinevaid potentsiaalseid vigu ning arendamisvõimalusi. Suur osa neist oli graafilise kasutajaliidese ja lausete vigasuse kohta, aga soovitati ka valida ainult lauseid, millel puuduvad stiilivead ning tehti märkus mitme korrektse versiooni eksisteerimise kohta.

Töö käigus tulid välja ka mitmed kitsaskohad näiteks mitme korrektse versiooni lubamine. Tekkisid ka mitmed edasiarendus ideed, näiteks erinevate veatüüpide kaasamine või välja jätmine, mis lisaks õppeviise juurde ning ei õpetaks vigaste lausete pealt vastavalt.

Viidatud kirjandus

Eesti vahekeele korpus. Statistika. (s.a.). Vaadatud 13.11.2022 https://evkk.tlu.ee/vers1/statistics.html

EstNLTK -- Open source tools for Estonian natural language processing. (2023). Vaadatud 12.03.2023 https://github.com/estnltk/estnltk

Halling, A. (2016). *Eesti keele keeleressursse kasutav õppeprogramm käänete õppimiseks. Bakalaureusetöö.* Vaadatud 22.11.2022 https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=53482&year=2016

Halling, A. (2016). *Õpime käändeid*. Vaadatud 24.11.2022 http://prog.keeleressursid.ee/opimekaandeid/game.php

Introduction to the Angular docs. (2023). Vaadatud 12.03.2023 https://angular.io/docs

Jahirhussain, A. K. (2022). *Easy Steps to Host an Angular App in GitHub Pages*. Vaadatud 29.04.2023 https://www.syncfusion.com/blogs/post/easy-steps-to-host-an-angular-app-in-github-pages.aspx

Jõgar, L. (2022). *Õpilaste tagasiside 2022. aasta emakeeleolümpiaadi lõppvoorule*. Vaadatud 02.12.2022 https://dspace.ut.ee/handle/10062/83901

Korpusest (s.a.). Vaadatud 13.11.2022 https://korpused.keeleressursid.ee/emma/index.php?v=info

Käänuk. (2019). Vaadatud 01.12.2022 http://k44nuk.ee/

Marchand, W. R. (2022). *The Rising Popularity of Python*. Vaadatud 03.04.2023 https://pythoncircle.com/post/763/the-rising-popularity-of-python/

Mis on ELLE? (s.a.). Vaadatud 13.11.2022 https://evkk.tlu.ee/about/us

Mis on tekstikorpus. (2022). Vaadatud 17.12.2022 https://teatmik.eki.ee/teatmik/mis-on-tekstikorpus/

ortograafia. (2023) Vaadatud 09.05.2023: https://sonaveeb.ee/search/unif/dlall/dsall/ortograafia/1

PostgreSQL. (2023). Vaadatud 08.02.2023 https://www.postgresql.org/

Põldaru, S. (2018). *Õigekeelsusvead riigieksamikirjandites.* Vaadatud 28.11.2022 http://dspace.ut.ee/handle/10062/60309

Põhikooli riiklik õppekava. (2011). Vaadatud 17.03.2023 https://www.riigiteataja.ee/akt/108032023005

Raik, M. (2003). *Eesti kirjakeele seisund eesti koolis.* Vaadatud 28.11.2022 https://dspace.ut.ee/handle/10062/40641

Ross, K. (2019). *MEIE JA TEIE EESTI KIRJAKEEL*. Vaadatud 01.05.2023 https://keeljakirjandus.ee/ee/archives/26013

Rummel, E. (s.a.). *Eesti õigekiri*. (Tiigrihüppe Sihtasutus) Vaadatud 01.12.2022 https://www.opetaja.edu.ee/ortograafia/

Senkel, R. (2016). *Keelevigade roll suhtumise kujundamisel teksti autorisse.* Vaadatud 12.11.2022 http://dspace.ut.ee/handle/10062/51871

TartuNLP Neurotõlge. (s.a.). Vaadatud 03.04.2023 https://translate.ut.ee/#info

The Good and the Bad of Docker Containers. (2022). Vaadatud 10.03.2023 https://www.altexsoft.com/blog/docker-pros-and-cons/

What Is Java Used For? (2022). Vaadatud 11.03.2023 https://www.coursera.org/articles/whatis-java-used-for

Use containers to Build, Share and Run your applications. (s.a.). Vaadatud 12.03.2023 https://www.docker.com/resources/what-container/

Vu, H. (2021). How to Write a Class Object to CSV. Vaadatud 20.12.2022 https://dev.to/htv2012/how-to-write-a-class-object-to-csv-5be1

Websites for you and your projects. (2023). Vaadatud 03.05.2023 https://pages.github.com/

What is Java Spring Boot? (s.a.). Vaadatud 09.02.2023 https://azure.microsoft.com/enus/resources/cloud-computing-dictionary/what-is-java-spring-boot/

What is Neon? (2023). Vaadatud 09.02.2023 https://neon.tech/docs/introduction/about

Winter, R. (2023). An Intro to Java for Back-End Programming. Vaadatud 07.05.2023 https://blog.hubspot.com/website/javascript-for-backend

Lisad

I. Lingid

1. Veebirakendus aadressil: https://viibur.github.io/project-fe/
2. Link Google Colab lehele: https://colab.research.google.com/drive/1afiYlhAmoDBtHcY3ELiCxDasPRV0PwtC?usp=sharing
3. GitHub link projekti tagaosale: https://github.com/Viibur/project_be
4. GitHub link projekti esiosale: https://github.com/Viibur/project-fe

II. Küsitlus

Veebiõppematerjal

Seoses lõputööga valmis ka õppematerjal. Küsitlus on seoses veebilehel olevate küsimuste ning vigadega.

Juhend testimiseks:
Lahendage veebirakenduses paarkümmend lauset, kasutades mõlemat lahendusviisi.
Lahendage lauseid nii korrektselt kui ka valesti.
Kui leiate lauseid või vigu mis teie arvates kohatud või valed, saate neid raporteerida. Jätke need ka meelde/pange kirja, sest nende kohta küsitakse ka küsitluses.

Link: https://viibur.github.io/project-fe/

* Indicates required question

Nimi(ei avalikustata, lihtsalt vastajate kontrolliks) *

Your answer

Kas juhend oli arusaadav? *

- ◯ Jah
- ◯ Ei

Kui ei, siis mis võiks juhendis teisiti olla?

Your answer

Teie arvamus vigastest lausetest *

Your answer

Kas leidsite mõne lause, mis oli Teie arvates vigane? (s.t., et lause või viga ise oli ebakorrektne või ebasobiv) *

- ◯ Jah
- ◯ Ei

Kui jah, nimetage, millised lauseid leidsite ning miks need ebasobivad on

Your answer

Hinnake lausete raskusastet (1 - väga lihtne, 10 - väga raske) *

1	2	3	4	5	6	7	8	9	10
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

Teie arvamus kahe erineva õppeviisi vahel valimisest *

Your answer

Kumb parandamisviis on Teie arvates lihtsam? *

- ◯ Vigaste sõnade ümber kirjutamine
- ◯ Lünkadesse vea valimine
- ◯ Other: _____

Kommentaarid, soovitused, arvamused. *

Your answer

Submit Clear form

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Andre Viibur,

(autori nimi)

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

Sagedasemate kirjavigade parandamise õpiprogramm,

(lõputöö pealkiri)

mille juhendaja on

Sven Aller,

(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.

Andre Viibur

09.05.2023