Tallinna Ülikool

Informaatika Instituut

EESTI VAHEKEELE KORPUSE MÄRGENDUSLIIDESE ARENDAMINE JA VÕIMALIKUD RAKENDUSED

Bakalaureusetöö

Autor: Marko Sultsing

Juhendajad: Erika Matsak ja Jaagup Kippar

Autor:..... “.....“2012

Juhendaja: ..... “.....“2012

Juhendaja: ..... “.....“2012

Instituudi direktor: ..... “.....“2012

Tallinn 2012

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

....

(kuupäev) (autor)

Sisukord

Mõistete seletused

CSS – *Cascading Style Sheets*, kaskaadlaadistik. Võimalusterohke stiilikeel, mis võimaldab kerge vaevaga paika panna veebilehekülje kujunduse.

HTML – *HyperText Markup Language*, veebilehekülgede standardne märgenduskeel.

JavaScript – skriptikeel. Vahend dünaamiliste veebilehtede loomiseks. Oskab suhelda ülalmainitud märgenduskeeltega.

Korpus – elektroonne keeleressurss, sisaldab kirjalikke tekste ja/või suulise kõne näiteid, koostatud kindlatel uurimis- ja rakenduslikel eesmärkidel, varustatud kasutajaliidesega.

Plone – Avatud lähtekoodiga sisuhaldussüsteem. Sisaldab Zope'i.

Python – populaarne kõrgtaseme objektorienteeritud programmeerimiskeel, mis töötab kõikides enamlevinud operatsioonisüsteemides. Töö kirjutamise hetkel leiavad laialdast kasutust keele kaks põhiversiooni, Python 2 ja 3. Käesoleva töö raames kirjutatud programmikood on kirjutatud Python 2.4.4-s.

Sisend – süsteemi sisestatud infohulk, millest tavaliselt oleneb süsteemi väljund.

Sõne – sõna tekstikasutuse ühik.

TAL – *Template Attribute Language*, mallkeel. Zope'is kasutusel olev keel dünaamiliste veebilehekülgede loomiseks. Kasutab spetsiaalseid atribuute, võimaldavad töödelda ja genereerida erinevaid kasutatava märgenduskeele elemente ja atribuute ning nende väärtuseid.

TEI – kasumitaotluseta ettevõtete grupp Text Encoding Initiative. Tegeleb rahvusvaheline digitaalsete tekstide esitamise standardite arendamisega.

Väljund – süsteemi poolt väljastatud mõõdetav infohulk, mis võib mõjutada teisti süsteeme.

Õppijakeel ehk vahekeel – termin on kasutusel seoses teise keele/võõrkeele omandamisega; tähistab keelevarianti, mida õppijad sihtkeelt kasutades loovad.

Õppijakeele- ehk vahekeelekorpus – keeleõppijate loodud kirjalike tekstide või suulise kõne elektroonne kogu, millel on oma kasutajaliides.

XHTML – *Extensible HyperText Markup Language*, laiendatav veebilehekülgede standardne märgenduskeel. Loodud HTML 4 ja XML 1.0 baasil.

XML – *Extensible Markup Language*, laiendatav standardne märgenduskeel. Erinevalt HTML-ist pole XML piiratud vaid fikseeritud elementide kasutamisega.

XPath – *XML Path Language*, XML-i päringkeel. Kasutatakse XML faili elementide ja atribuutide sees navigeerimiseks. Võimaldab teha lihtsamaid tehteid elementide ja nende atribuutidega seotud väärtustega.

Zope – Pythonis kirjutatud veebirakenduste server ja veebiraamistik, mõeldud veebipõhiste rakenduste loomiseks ja jooksutamiseks. Töö kirjutamise hetkel on kasutusel serveri kaks põhiversiooni Zope 2 ja BlueBream, varem tuntud kui Zope 3. Käesoleva töö raames kirjutatud kood jookseb Zope 2 serveris.

Sissejuhatus

Käesoleva bakalaureusetöö eesmärk on arendada Eesti vahekeele korpuse (EVKK) märgendusliidest, täiendades olemasolevaid funktsioone ning lisades uusi. Liidese näol on tegemist rakendusega, mis võimaldab filoloogidel käsitsi korrigeerida korpuses süntaksanalüsaatori märgendeid.

Bakalaureusetöö raames sooritatava arendusprotsessi eesmärgiks on luua praktiline ja töökorras rakendus, mille järele on reaalne vajadus. EVKK-sse implementeeritud eesti keele süntaksianalüsaator analüüsib ja märgendab tekste automaatselt. Kuna aga korpuses leiduvad tekstid pole tihtipeale keeleliselt korrektsed ja ka süntaksianalüsaatori töös esineb mitmesust, siis on reaalne vajadus analüsaatori märgendust käsitsi korrigeerida. Seda võimaldavat rakendust töö alustamise hetkel EVKK-s ei olnud; eraldi liides on olemas vaid vealiikide käsitsi märgendamise ja korrigeerimise jaoks.

Otseseks sihtrühmaks on EVKK märgendamisega seotud filoloogid ja keeletehnoloogid, kes saavad märgendusliidet kasutades korrigeerida süntaksianalüsaatori tööd ning treenida analüsaatori rakendamist eesti keele grammatikakorrektorina.

Töös on lahti seletatud kasutatud mõistestik räägitud pikemalt korpustest, sealhulgas Eesti vahekeele korpusest. Ühtlasi on kirjeldatud märgendusliidese arendusprotsessi käigus kirjutatud koodi, sellega seotud probleeme ning põhjendatud tehtud valikuid. Samuti on kirjas mõtted märgendusliidese edasiste arendusvõimaluste kohta.

1. Korpus

Korpus on elektroonne keeleressurss, mis loodud kindlatel eesmärkidel. Korpust eristavad tavalistest tekstikogudest reeglina kolm olulist punkti:

1. Valim ja esindavus. Korpuse loomisel valitakse tekstid, mis iseloomustavad analüüsitavat keelt kõige paremini ning annavad tervikpildi kõigist keelekasutuse valdkondadest. See tähendab, et tekstid peavad olema võimalikult mitmekesised ja eesmärgipäraselt valitud. Tekstide maht on korpustes tavaliselt piiratud (500 – 1500 sõnet, keskmiselt 1000 sõnet). Sellele vaatamata esineb korpustes ka väga erineva suurusega tekste.
2. Piiratud maht. Korpused võivad olla kas avatud või suletud ehk standardkorpused. Avatud korpustesse saab pidevalt tekste lisada (nt Eesti Keele Instituudi eesti kirjakeele korpus), suletud korpustele mitte (nt Browni korpus, mis sisaldab standardkorpusele kohaselt 1 miljon sõnet või British National Corpus). Mahukad korpused jagunevad tihti väiksemateks alamkorpusteks (nt International Corpus of Learner English).
3. Masinloetav kuju. Tänapäeva korpused on elektroonsed ning neis leiduv info on arvutite abil töödeldav. Algselt tähendas korpus ainult füüsiliselt kujul eksisteerivate tekstide kogu, mis nüüdseks on harv nähtus. Masinloetava korpusel on selged eelised: sellise korpuse andmed on otsitavad ja töödeldavad kujul, mis paberkandjal poleks lihtsalt võimalik; andmetele on võimalik lisada erinevat lisainformatsiooni, rikkumata originaalandmeid; lisaks kaasnevad sellega inimlike eksimuste vähenemine ning korralik ajasääst. (vt McEnery & Wilson 2001: 29-32)

1.1. Korpuste liigitamine

Korpuseid saab liigitada vastavalt nende koostamisprintsiipidele ja kasutusvõimalustele:

1. Kirjutatud tekstide või suulise kõne korpus. Põhiline erinevus nende kahe korpuseliigi vahel on see, et suuline kõne on korpusesse salvestatud helifailina ja transkribeeritud

tekstina, samas kui kirjutatud tekstid on sisestatud kas *online*-variandina (originaalkujul) või käsikirjalise teksti alusel digitaliseerituna.

2. Avatud või suletud korpus (vt eespool piiratud maht). Avatud korpuse tekstide hulk pole lõplik. Neid saab sinna vajadusel lisada või sealt välja võtta. Avatud korpus on ühtlasi tuntud ka monitorkorpuse nime all. Piiratud mahuga ehk suletud korpust nimetatakse ka standardkorpuseks, mille maht on tavaliselt 1 miljon sõnet, korpuse tekstide hulk ei muutu, kuna on muutmisele suletud.
3. Märgendamata või märgendatud korpus. Märgendamata ehk nn puhta teksti korpus sisaldab ainult originaalvariandis tekste. Märgendatud korpuses on tekstidele lisatud informatsiooni, mis sõltuvalt korpuse kasutamise eesmärkidest võib olla igas korpuses erinev. Reeglina saab märgendatud tekstist soovi korral alati kätte nn puhta teksti. Mõlemat tüüpi korpustest on palju kasu, kuid korralik märgendus annab korpusele palju funktsionaalsust juurde. (vt McEnery & Wilson 2001: 30–32)

2. Teksti märgendamine

Märgendamine tähendab teksti varustamist lisainfoga, mis ei tule tingimata teksti lugemisel esile. Selline info aitab nii inimestel keelt tundma õppida kui ka märgatavalt lihtsustada arvutiga teksti töötlemist.

Geoffrey Leech (1993) on toonud välja seitse olulist punkti, mis peaksid kehtima iga märgendatud korpuse kohta.

1. Märgendatud tekstist peab olema võimalik eemaldada märgendused ja saada kätte teksti originaalkuju.
2. Märgendatud tekstist peab olema võimalik kätte saada ainult märgendused ning salvestada need kuskile mujale.
3. Märgendusskeem peab olema kättesaadav tavakasutajale. Enamikel korpustel on olemas juhis, mis selgitab kasutusel oleva märgendusskeemi ehitust.
4. Peab olema selge, kes ja kuidas on teksti märgendanud. Näiteks võivad korpuse tekste märgendada mitu inimest käsitsi või on märgendus arvuti abil automatiseeritud.
5. Lõppkasutajale peab olema selge, et märgendus ei pruugi olla vigadeta.
6. Märgendusskeemid peaksid nii palju kui võimalik vastama laiemalt aktsepteeritud põhimõtetele.
7. Ühelgi märgendusskeemil pole eesõigust võtta ennast kui standardit. Standardid tekivad läbi praktiliste lahenduste nende integreerimise alusel (vt Leech 1993: 275–281)

Tekstide märgendamise juures on tänapäeval väga paljudel juhtudel hakatud järgima TEI juhiseid masinloetavate tekstide esitamiseks.

2.1. TEI

Digitaalsete tekstide esitamise standardite arendamise on enda eesmärgiks võtnud kasumitaotlusteta ettevõtete koondis TEI (*Text Encoding Initiative*). Selle koondise

põhiväljundiks on juhised, mis määratlevad masinloetavate tekstide kodeerimismeetodeid, seda peamiselt humanitaar-, sotsiaalteaduste ning keelevaldkonnas. Lisaks pakutakse mitmeid tugimaterjale, näiteks materjale TEI kodeeringureeglite õppimiseks või tarkvara selle kasutamiseks. Aastast 1994 on TEI juhised leidnud laialdast kasutust nii raamatukogudes, muuseumides kui ka kirjastustes.

Lühendit TEI kasutatakse tihti mitteametlikult kui sünonüümi TEI direktiivides määratletud kodeerimismeetodile. TEI direktiivide eesmärgiks on defineerida kodeerimismeetodeid formaalses märgenduskeeles. (vt http://www.tei-c.org/Support/Learn/intro.xml, viimati vaadatud 28.12.2011). Esimene stabiilne versioon P4 väljastati aastal 2002. 2007-ndal aastal tuldi välja versiooniga P5, mis pole eelmise versiooni suhtes täielikult tagurpidi ühilduv. (vt http://www.tei-c.org/index.xml, viimati vaadatud 28.12.2011).

3. Eesti vahekeele korpus

EVKK on loodud Tallinna Ülikooli eesti keele ja kultuuri instituudis. Korpus sisaldab eesti keelt teise või võõrkeelena õppijate kirjalikke loovtöid (peamiselt esseed). Tegu on avatud korpusega, mis koostatud uurimistöö eesmärgil ning avatud kõikidele huvilistele. (vt Eslon & Metslang 2007: 105).

2011. aasta detsembri seisuga on EVKK osaliselt märgendatud tekstikogu, mis sisaldab 1 226 132 sõnet. Märgendatud sõnesid on 514 464, neis kokku 56 086 viga.

Lisaks tekstidele sisaldab Eesti vahekeele korpus ka metateavet teksti autori kohta (nt sugu, vanus, haridus, emakeel jm), teksti liigi (nt essee, isiklik kiri jm) ja teksti kirjutamise tingimuste kohta (klassiruumis kirjutatud või varem kodus ette valmistatud tekst).

Tekstikogust saab soovitud andmeid kätte kasutajaliidese abil, valides sobiva päringu (nt kõik esseed, mis on kirjutanud soome emakeelega inimene vanuses kuni 25 aastat jne). Korpuse kasutajaliides kuvab metaandmeid teksti (teksti maht sõnades, vigade hulk tekstis vealiigiti jne) ja teksti autori kohta (emakeel, kodune keel, sugu jne). Märgendatud vigu on võimalik kuvada vealiikide kaupa. Kasutajaliidese abil saab leida kindla sõna või sõnavormi kõik kasutuskontekstid ehk konkordantsiread. Korpuse materjalid on statistiliselt töödeldavad, korpuses on sõna- ning vormisageduse statistika.

Korpusest saab tekste kätte nii märgendatud kui ka märgendamata kujul. Märgendus toob välja lingvistiliselt tõlgendatud keelevead. Andmeid kuvatakse märgenduskeele XHTML abil, päringute jaoks on kasutusel XPath päringukeel.

3.1. Eesti vahekeele korpuse veaklassifikatsioon

EVKKs tähendab keeleviga grammatikareeglile või kommunikatiivsele eesmärgile mittevastavat keelekasutust, mis ei hõlma väsimusest ja hooletusest põhjustatud eksimusi. Samuti ei peeta antud kontekstis vigadeks sõnade variatsioone, mille on põhjustanud soov mõtet väljendusrikkamalt edastada.

EVKK lingvistiline veaklassifikatsioon on hierarhiline: koosneb veaklaasidest, -liikidest, alamliikidest jne.

Keelesüsteemi süntagmaatilise (sõnade sidusus semantika, grammatika ja pragmaatika aspektis) ja paradigmaatilise (keeletasandite hierarhia) telje ristumisel saab eristada 18 veaklassi (vt tabel 1), mis sisaldavad vigade liike ja alamliike. EVKK veaklassifikatsioonis on kokku 170 vealiiki. (vt Eslon & Metslang 2007: 106–107)

Süntagmaatika / Paradigmaatika	Semantika	Grammatika	Pragmaatika
Tekst	1	2	3
Lause	4	5	6
Sõnaühend	7	8	9
Sõna	10	11	12
Morfeem	13	14	15
Grafeem	16	17	18

Tabel 1. Keelevigade lingvistiline klassifikatsioon

3.2. Eesti vahekeele korpuse märgendus

Eesti vahekeele korpuses mõistetakse märgendamise all veamärgendust, mille tulemusena on vigane ja korrektne õppijakeel teineteisest selgelt eristatavad. Tavaliselt saab üks keeleviga mitu märgendit, kuna vead on enamjaolt erinevalt tõlgendatavad. Tegemist on mitmetasandilise süsteemiga, kus kõrgema tasandi märgend sisaldab endas kõiki alamtasandite märgendeid.

EVKK üle miljonilisest kogumahust on käsitsi märgendatud 514 464 sõnet, järk-järgult liigutakse poolautomaatse märgenduse suunas.

Hetkeseisuga on loodud ja korpusesse integreeritud sõnajärje vealeidja prototüüp, mida tuleb edasi arendada. Selle aluseks on eesti keele süntaksianalüsaator, mis on samuti korpusesse implementeeritud. Prototüüp on loodud kasutades programmeerimiskeelt Python. Tegemist on

statistikapõhise programmiga, mis testib oluliste lauseliikmete järgnevusi esimeses osalauses ja lihtlauses, lähtudes sõnade järjekorrast. Nt: lauseliikmete järjend *Internetis* (@ADVL) *on* (@FMV) *võimalik* (@PRD) *kasutada* (@SUBJ) *mitmeid* (@NN>) *teenuseid* (@OBJ) ja sellele vastav sõnajärjemall [‘@ADVL’, ‘@FMV’, ‘@PRD’, ‘@SUBJ’, ‘@OBJ’]. Sama algusmärgendiga korduvad sõnajärjemallid moodustavad erineva sagedusega ilmnevaid sõnajärjemustreid, mis paigutatakse andmepuusse. Eesti keelele omaseim andmepuu sisaldab sõnajärjemustreid, mis algavad märgendiga @SUBJ: '@SUBJ' '@FMV' '@ADVL' '@ADVL jne. Sõnajärje vealeidja prototüübi töö tulemuslikkus on esialgu 87,72%. (http://www.keeletehnoloogia.ee/projektid/vako, viimati vaadatud 30.12.2011) Detailsem ülevaade sõnajärje vealeidja loomisest vt Metslang & Matsak 2010; Matsak & Metslang & Kippar 2010 ja Matsak & Eslon & Kippar 2010.

Teine pooleliolev arendus on oletaja-lemmatiseerija ehk veakindel lemmatiseerija, mis peab automaatselt ära tundma vigased sõnad ja vormid ning need lemmaga siduma. Oletaja-lemmatiseerija töö baseerub Levenshteini kaugusel (nt: kantsid vs kandsid > t asendada d-ga > kaugus = 1; igasugulased vs igasugused > kustutada l ja a > kaugus = 2) ning foneetilisel algoritmil Metaphone, mis moodustab iga sõna jaoks hääldusest ainult kõige olulisemat sisaldava kuju. Referentssõnastik on moodustatud Eesti Ekspressi korpuse alusel (7,2 miljonit sõnavormi), mida töödeldi Metaphone algoritmiga, et jagada sõnad hääldussarnasuse järgi klassidesse. Õppijakeele vigastele sõnakasutustele leiti samuti hääldusku ju, arvutati Levenshteini kaugused vigase sõna ja sama hääldusku jug a sõnade vahel referentssõnastikus ning leiti suurima tõenäosusega kandidaat nende sõnade hulgast, mille kaugus on maksimaalselt 2. Oletaja-lemmatiseerija teeb valikud otsustuspuude abil kirjavahemärkide, mittesõnade, pärisnimede ja ühetähenduslike sõnade jaoks. Mitmesuste lahendamisel tugineb tõenäosuste arvutamisele. (http://www.keeletehnoloogia.ee/projektid/vako/, viimati vaadatud 30.12.2011)

Käesoleva töö seisukohalt on oluline ära märkida, et nii süntaktaksi- ja morfoanalüsaatori kui ka oletaja-lemmatiseerija töös esineb mitmesusi ning vale märgendust. See tähendab, tuleb ette olukordi, kus näiteks süntaktilise analüüsi puhul antakse ühele sõnale mitu morfosüntaktilist tõlgendust. Õige variandi filoloog, kes teeb selle töö käsitsi. Et vähegi korrigeerimist hõlbustada, selleks ongi vaja vastavat märgendusmoodulit.

4. Arendusprojekti kavandamine

Käesolev peatükk räägib täpsemalt bakalaureusetöö raames tehtud arendusprotsessile eelnenud etappidest. Pikemalt seletatakse lahti juba olemasolevad rakendused, projekti eesmärgid ning arendusprotsessi käigus kasutatavad vahendid.

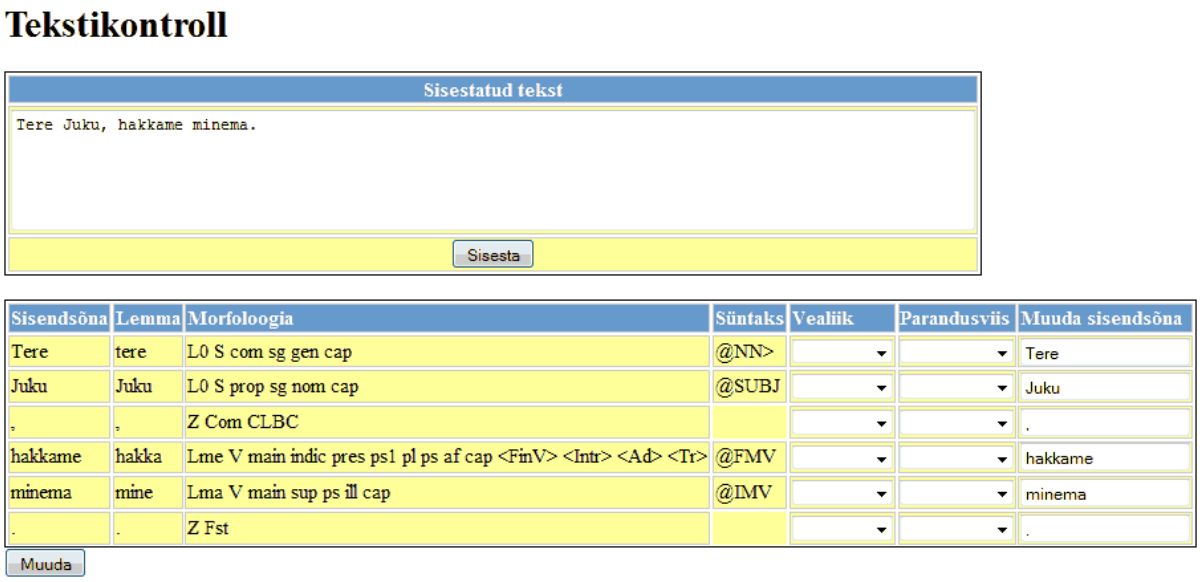
4.1. Olemasolevad rakendused

Käesoleva töö autor alustas märgendusliidese arendamist seminaritöö raames. Tulemuseks oli liidese esmaversioon (vt joonis 1), mis on oma võimalustelt praegusest oluliselt piiratum.

Algse liidese põhifunktsioonid:

1. Võimalus saata kasutaja sisestatud tekst EVKK-sse implementeeritud süntaksianalüsaatorisse, mille väljund on automaatselt analüüsitud tekst.
2. Oskus lugeda, töödelda ja kuvada morfosüntaktiliselt märgendatud teksti.

Märgendusliides on kirjutatud Zope 2 veebirakenduste serveris, kasutades programmeerimiskeelt Pythoni ja mallkeelt TAL. Rakendus vajab kaasaegset veebibrauserit ning Interneti ühendust.



Joonis 1. Esialgse märgendusliidese ekraanipilt: vastus päringule

Süntaksianalüsaatorisse saadetavat teksti võis esialgse liidese abil esitada kahel viisil:

1. Aadressiriba kaudu, kasutades muutujanime „tekst“.
2. Mitmerealise tekstivälja kaudu.

Olenevalt sellest, kuidas tekst liidesele edastati, tuli teksti erinevalt töödelda, et saada analüsaatori jaoks sobiv kuju.

Esialgses märgendusliideses puudus võimalus redigeerida keele grammatilisi andmeid. Selle asemel on kasutatud lahendust, mis võimaldab muuta sõnesid ja saata neid uuesti analüsaatorisse märgendamiseks. Ühtlasi puudub funktsioon, mis salvestaks päringute andmeid mingil kujul liideseväliselt ning sedasi, et need andmed ei kaoks peale liidesega töö lõpetamist. Detailsema ülevaate leiab selle kohta autori seminaritööst (vt Sultsing 2010).

4.2. Arendusprojekti eesmärgid

Käesoleva arendusprojekti eesmärk on luua EVKK esialgse märgendusliidese alusel niisugune liides, mis võimaldaks filoloogidel korrigeerida süntaksianalüsaatorist läbi käinud tekstide märgendusi ja teha vajalikke parandusi.

Arendusprotsessi käigus täiendatud märgendusliides peab vastama järgmistele nõuetele:

1. Oskus lugeda, töödelda ja kuvada EVKK-s süntaksianalüsaatori märgendatud teksti, sh töötlemisega seoses oskus eristada, kas ühe sõne kohta antakse analüsaatoris vastuseks vaid üks või mitu gruppi andmeid.
2. Iga tekstis oleva sõne kohta käivate andmete korrigeerimise võimalus, sh vääralt kokku kirjutatud sõnade lahku kirjutamise võimalus.
3. Vastavalt TEI P4 kodeeringureeglitele märgendatud teksti ja XML märgenduskeeles vormindatud tekstifaili salvestamine nii originaal- kui ka korrigeeritud kujul.
4. Olemasolevate funktsioonide viimistlemine, sh EVKK-sse implementeeritud süntaksianalüsaatori märgendatud teksti lugemise koodi täiustamine.
5. Brauseripõhine kasutajaliidese täiendamine, et vastu tulla eespool mainitud uuendustele.

4.3. Arendusprotsessis kasutatavad vahendid

Arendusprotsessi käigus kasutatavad vahendid on enamjaolt paika pandud juba olemasoleva süsteemi poolt, kuhu kuuluvad EVKK-sse implementeeritud süntaksianalüsaator ja autori varem loodud märgendusliidese esialgne versioon – kirjutatud Zope 2 veebiraamistikus ja kasutatakse ka antud arendusprotsessi puhul. Samuti on seoses eelneva rakenduse arendamisega ning ühtlasi kasutatava keskkonnaga kasutusel TAL mallkeel, mis vastutab kasutajaliidese dünaamilise genereerimise eest, et seda brauseris korrektses XHTML märgenduskeeles kuvada.

Tagaplaanil toimivad Pythoni programmeerimiskeeles kirjutatud vajalikud funktsioonid, mis käivitatakse TALi poolt koos edastatud sisendiga. Väljund edastatakse uuesti mallkeelele, mis otsustab, mida sellega edasi teha.

Vähemal määral on kasutusel ka JavaScriptis loodud funktsioonid ning kujunduse eest vastutav CSS.

5. Arendusprotsess

Käesolevas peatükis on välja toodud ja põhjendatud olulisemaid arendusprotsessi käigus tehtud valikuid. Ühtlasi on kirja pandud, millise tulemuseni arenduse tulemusena jõuti ning millised on edasised plaanid.

5.1. Analüütilised põhjendused arendamise käigus tehtud valikutele

Arendusprotsessi käigus tehtud valikud on seotud teksti sisestamise, regulaaravaldiste, märgendatud tekstimassiivi ja faili salvestamisega TEI kodeeringus, samuti märgendusliidese täiustamisega.

5.1.1. Teksti sisestamine

Autori seminaritöö raames kirjutatud rakendus sai süntaksianalüsaatorisse saadetavad andmed kätte kas liidese tekstiväljas olevast tekstist või brauseri aadressiriba kaudu sisestatud muutuja väärtusest. Need on käesoleva arendusprotsessi käigus asendatud teksti lugemisega failist.

See muutus on põhjustatud soovist vähendada liideseväliste tegurite mõju tekstile (nt brauseri kasutatav kodeering) ning soovist minna üle vaid ühele viisile teksti sisestamiseks – sellisele, mis oleks ühtlasi võrreldav korpuses oleva tekstiga. Selline lahendus lihtsustab testimist ja võimaldab tulevikus kergemini otse korpusega ühendada.

Oluline on siinkohal märkida, et sisendiks pole mitte märgendatud, vaid märgendamata tekst, mis Pythoni funktsiooni vahendusel alles saadetakse süntaksianalüsaatorisse märgendamiseks. Selle lahenduse põhjuseks on asjaolu, et töö tegemise hetkel polnud autoril otsest ligipääsu korpuse tekstiallikatele, küll aga analüsaatorile.

5.1.2.Regulaaravaldised

Üheks esimeseks arendusprotsessi eesmärgiks on täiustada viisi, kuidas süntaksianalüsaatorist tulnud teksti töödeldakse, kuna see lihtsustaks järgnevate eesmärkide täitmist. Seminaritöö käigus valmistatud esialgne rakendus töötles süntaksianalüsaatorile saadetud päringut ja sealt tulnud vastust lihtsate tekstimärgendite asendamistega, mis küll toimis, kuid nõudis liigsel hulgal koodiridu ning oli rohkem avatud vigade tekkimisele. Sisendi suhtes sisse viidud muutustega, mis tehtud praeguse arendusprotsessi käigus, kadus ära vajadus analüsaatori sisendteksti samal kujul töödelda, kuid töötlus on siiski oluline märgendatud teksti jaoks, mis süntaksianalüsaatorist vastuseks tuleb. Parema töötlusviisi otsimise käigus jõudis autor kiirelt järelduseni, et mõistlik on kasutada regulaaravaldisi (*regular expressions*).

Regulaaravaldis tähendab reeglina erimärgendust sisaldavat tekstirida, mille tõlgendatud sisu vastab kas osaliselt või täielikult ühele või mitmele tekstile, kusjuures need tekstid võivad olla väga erinevad ning mitte vastata mingitele ühistele reeglitele. Paljud programmeerimiskeeled ja tekstiredaktorid toetavad regulaaravaldiste kasutamist, sh ka Python.

Kuigi regulaaravaldis võib olla ka ainult näiteks tähemärk „a“, ei erine selle avaldisega sooritatud otsing kuidagi tavaliselt otsingust. Regulaaravaldise oluliseks komponendiks on metakarakterid, nende täielik nimekiri on näha koodinäites 1. Kui tavalise Pythonis sooritatud otsingu puhul vastaks punktile kõik juhud, kus tekstist leitakse sama tähemärk, siis regulaaravaldiste puhul antakse vastuseks peaaegu kõik karakterid, k.a punkt. Tõeline regulaaravaldiste jõud seisneb aga metakarakterite ja tavaliste tähemärkide ühises kasutamises, et sooritada tekstide peal erinevaid päringuid ja operatsioone.

Koodinäide 1. Regulaaravaldiste metakarakterid

```
. ^ $ * + ? { } [ ] \ | ( )
```

Pythonis saab sooritada toiminguid regulaaravaldiste abil, kaasates mooduli *re*. Koodinäide 2 seletab, kuidas seda moodulit kasutades jupitatakse kõik kohad tekstis, kust leitakse valgetühiku (*whitespace*) tähemärk (koodis tähistatuna \s) ning paigutatakse need massiivi

(*array*). Keerulisema teksti puhul nõuab see ilma regulaaravaldisi kasutamata sama tulemuse saavutamiseks oluliselt rohkem etappe, näites antud koodi aga täiendama ei peaks.

Koodinäide 2. Päring regulaaravaldisega

Väljundiks on kõik väiketähed vahemikus a ja z (inglise keele põhjal).

```
>>> import re
>>> text = 'Hello, world!'
>>> print re.split('\s',text)
['Hello,', 'world!']
```

Kasutades käesoleva arendusprojekti raames regulaaravaldiste funktsioone *re.sub()* (teksti asendamine), *re.split()* (teksti jupitamine massiivi) ning *re.findall()* (tekstist otsimine koos vastuse salvestamisega massiivi) on saadud tulemuseks lühem, puhtam ja veakindlam kood. Regulaaravaldiste kasutamise tulemuseks on väljundina kõiki märgendatud teksti andmeid sisaldav mitmetasemeline massiiv.

5.1.3.Massiiv

Märgendatud ja regulaaravaldistega töödeldud teksti andmete salvestamine mitmetasemelisse massiivi tähendab seda, et kasutusel on üks ülemmassiiv, mille sisse on paigutatud alammassiivid. Massiiv on kasutusel seetõttu, et sinna on väga lihtne märgendusliidese tagaplaanil rakendatavate päringute abil andmeid salvestada ning sealt kätte saada. Antud andmekogum kasutab sarnast hierarhiat (vt koodinäide 3) korpusesisese märgenduse jaoks (vt koodinäide 4), st kogu märgendatud teksti alammassiivides sisalduva ülemmassiivi otseseks alamaks on lausemassiivid, mis omakorda sisaldavad sõnemassiive. Ühtlasi on igas sõnemassiivis nii mitu massiivi, kui palju on selle sõne kohta erinevaid morfosüntaktiliste andmete gruppe. Üheks olulisemaks erinevuseks, võrreldes korpusesisese märgendusega, on

ühe sõna kohta käivate erinevat tüüpi andmete selgem eristatavus, kuna iga andmetüüp on salvestatud eraldi väärtusena.

Koodinäide 3. Mitmeastmeline massiiv

Genereeritud massiiv on tehtud morfosüntaktiliselt märgendatud tekstile „Tere, maailm! Kuidas läheb?“ Täpitähed on automaatselt salvestatud 16-süsteemis ja UTF-8 kodeeringus.

```
 [[[['Tere', 'tere', 'L0 I cap', '@B', '', '', '', ''], ['', 'tere', 'L0 S
com sg gen cap', '@NN>', '', '', '', '']], [[',', ',', 'Z', 'Com', '', '',
'', '']], [['maailm', 'maa_ilm', 'L0 S com sg nom cap', '@SUBJ', '', '',
'', '']], [['!', '!', 'Z', 'Exc', '', '', '', '']]], [[['Kuidas', 'kuidas',
'L0 D cap', '@ADVL', '', '', '', '']], [['l\xc3\xa4heb', 'mine', 'Lb V main
indic pres ps3 sg ps af cap <FinV>', '@FMV', '', '', '', '']], [['?', '?',
'Z', 'Int', '', '', '', '']]]]
```

Koodinäide 4. Süntaksianalüsaatori vastus sisendtekstile „Tere, maailm! Kuidas läheb?“

```
"<s>"
"<Tere>"
      "tere" L0 I cap @B #1->1
      "tere" L0 S com sg gen cap @NN> #1->1
"<,>"
      "," Z Com #2->2
"<maailm>"
      "maa_ilm" L0 S com sg nom cap @SUBJ #3->3
"<!>"
      "!" Z Exc #4->4
"</s>"
"<s>"
```

```
"<Kuidas>"
      "kuidas" L0 D cap @ADVL #1->2
"<läheb>"
      "mine" Lb V main indic pres ps3 sg ps af cap <FinV> @FMV #2->2
"<?>"
      "?" Z Int #3->3
"</s>"
```

Astmelise hierarhia kasutamine võimaldab massiivi peal tsükleid „jooksutada“ ja vajadusel mõne kindla lause väga lihtsalt kätte saada. Nendesse andmekogudesse sisestatud andmeid kasutatakse kahel otstarbel: tekstiandmete salvestamiseks TEI kodeerimisreeglite järgi ning märgendusliideses kuvamiseks.

5.1.1.Massiivi väärtuste TEI kodeeringus faili salvestamine

Massiivis leiduvate andmete TEI kodeeringus XML märgendusega faili salvestamine on üks olulisemaid uusi funktsioone, mis antud arendusprojekti käigus märgendusliidesele lisati.

Vaheetapina tuli töö autor välja lahendusega, kus kõik andmed võeti massiivist tsükleid kasutades ja liideti need tekstijadaga (*text string*), mille lõpptulemusena tekstifail salvestati. Väljundiks oli korrektse hierarhiaga XML märgendusega fail.

Selline kood ei vastanud siiski autori soovitud tulemusele, kuna polnud eriti selgelt loetav, vajas erinevat tekstijada nii originaal- kui ka korrigeeritud märgendatud teksti jaoks. Samuti ei kasutanud selline lahendus Pythonisse sisse ehitatud funktsioone, mis sama töö efektiivsemalt ära teevad. Seetõttu sai üle mindud tõhusamale lahendusele.

Sarnaselt esialgse lahendusega kasutab ka uus versioon tsükleid, et andmemassiivist andmeid kätte saada. Erinevus tuleb sisse aga Pythoni mooduli *lxml.etree* kasutuselevõtuga, mida eelistati sarnase funktsiooniga moodulile *ElementTree*. Määravaks said seejuures *lxml.etree*

paremad võimalused ning tugi kasutusel olevas Pythoni versioonis. Mõlemad moodulid võimaldavad genereerida ja lugeda XML hierarhiat, kuid käesoleva töö raames on oluline just tekstimärgendusi sisaldava XML puu loomine.

Antud moodul genereerib kirjutatud koodis järk-järgult puu elemendid ning sõne tasandile jõudes hakkab ühtlasi ka sõna kohta käivaid andmeid neile vastavatesse elementidesse ja atribuutidesse sisestama (vt koodinäide 5). Peale kogu elemendipuu loomist salvestatakse see uude muutujasse XML hierarhiat esindava teksti kujul (*method='xml'*), kus iga element asub omaette reas ning on hierarhiliselt paigutatud (*pretty_print='True'*). See muutuja salvestatakse tekstifailina. Olenevalt sellest, kas tekstiandmete seas on täiendusi tehtud või mitte, võidakse ühe teksti kohta luua kuni kaks faili, üks parandamata ja teine parandatud teksti kohta.

Koodinäide 5. Märgendatud teksti andmeid sisaldava XML puu loomine ning kuvamine esitamiseks sobival kujule

Antud näites on tekst väljastatud ekraanil, arendusprojekti käigus loodud rakenduses salvestatakse see tekstifailina.

```
sentence_array = [[[['Tere', 'tere', 'L0 I cap', '@B', '', '', '', ''],
['', 'tere', 'L0 S com sg gen cap', '@NN>', '', '', '', '']], [[',', ',',
'Z', 'Com', '', '', '', '']], [['maailm', 'maa_ilm', 'L0 S com sg nom cap',
'@SUBJ', '', '', '', '']], [['!', '!', 'Z', 'Exc', '', '', '', '']]],
[[['Kuidas', 'kuidas', 'L0 D cap', '@ADVL', '', '', '', '']],
[['l\xc3\xa4heb', 'mine', 'Lb V main indic pres ps3 sg ps af cap <FinV>',
'@FMV', '', '', '', '']], [['?', '?', 'Z', 'Int', '', '', '', '']]]]

from lxml.etree import Element, SubElement, tostring

TEI = Element('TEI',xmlns = 'http://www.tei-c.org/ns/1.0')

text = SubElement(TEI, 'text')

body = SubElement(text, 'body')

entryFree = SubElement(body, 'entryFree')

sentences = sentence_array

for sentence in sentences:
```

```
    s = SubElement(entryFree, 's')
    for words in sentence:
        w = SubElement(s, 'w')
        for word in words:
            w2 = SubElement(w, 'w', lemma=word[1].decode('utf-8'))
            if word[0] != ' ':
                w2.text = word[0].decode('utf-8')
            m = SubElement(w, 'm')
            m.text = word[2].decode('utf-8')
            syntax = SubElement(w, 'syntax')
            syntax.text = word[3].decode('utf-8')
            W_ERR = SubElement(w, 'W_ERR')
            W_ERR.text = word[4].decode('utf-8')
            LEMMA_ERR = SubElement(w, 'LEMMA_ERR')
            LEMMA_ERR.text = word[5].decode('utf-8')
            ERR_M = SubElement(w, 'ERR_M')
            ERR_M.text = word[6].decode('utf-8')
            ERR_S = SubElement(w, 'ERR_S')
            ERR_S.text = word[7].decode('utf-8')
TEI_string = tostring(TEI, encoding='utf-8', method='xml',
pretty_print='True')
print TEI_string
```

Kuigi genereeritud XML vastab TEI kodeeringureeglitele, pole antud väljundi puhul tegemist siiski täieliku TEI vormindusega. Puudu on mõningad TEI vorminduse elemendid, mis iseloomustavad faili sisu. Nende elementide lisamine on üks tulevikuplaanidest, mis

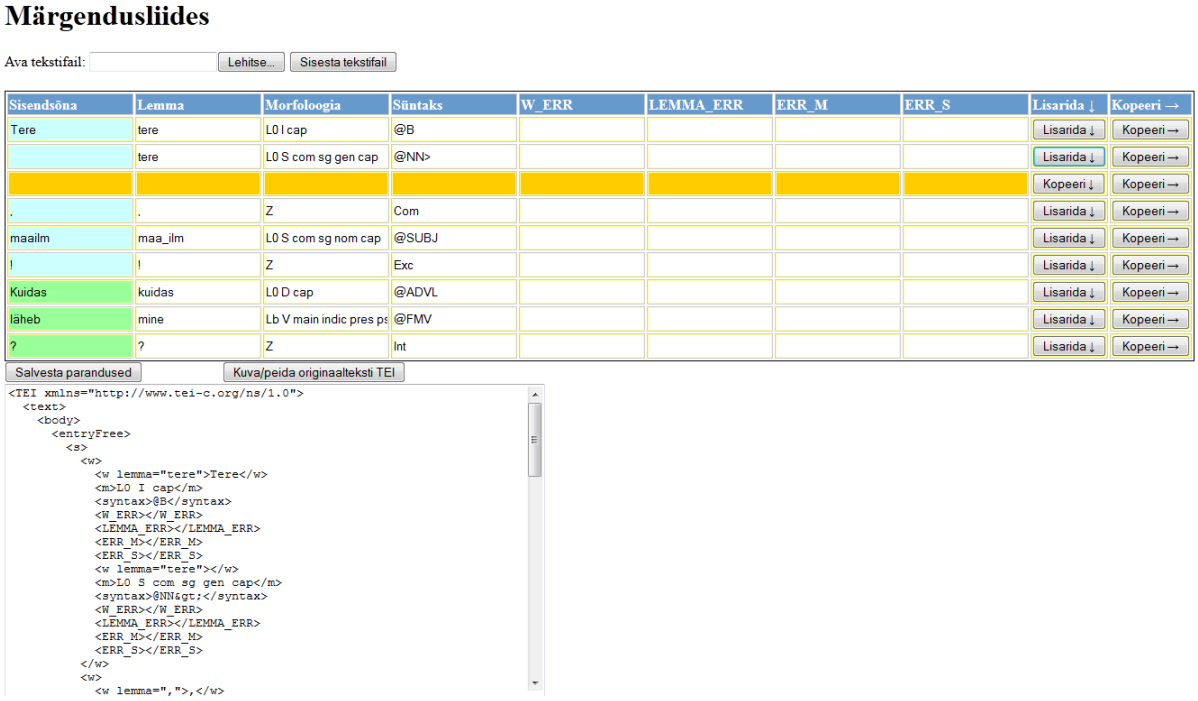
tõenäoliselt rakendatakse samaaegselt koos Eesti vahekeele korpuse märgendatud tekstide juurde kuuluvate metaandmete XML faili lisamisega.

5.1.4.Märgendusliidese täiendused

Seoses arendusprotsessi käigus tehtud täienduste ning uute võimaluste lisamisega oli vajadus täiendada ka olemasolevat märgendusliidest.

Olulisemad muudatused, mis on sisse viidud märgendusliidese uude versiooni (vt joonis 2) võrreldes seminaritöö käigus valminud esialgse versiooniga (vt joonis 1):

1. Teksti sisestamiseks kasutusel olnud tekstilahter on asendatud faili avamise ja laadimise väljadega. See muutus on otseselt seotud alapeatükis 5.1.1. mainitud põhjustega.
2. Märgendatud teksti kuvavas tabelis on iga märgendatud sõne andmete kohta rohkem lahtreid, sest ka erinevaid andmeid võib olla rohkem. Kasutusel on ainult tekstilahtrid. Uued sai lisatud, kuna oli vaja võimaldada märgendusliidese kasutajal märgendatud tekstis parandusi sisse viia, ilma et originaalandmeid muutuks.
3. Iga sõne kohta saab lisada ühe uue lisarea. Seda võimalus sai lisatud olukordade jaoks, kus on vaja valesti kokku kirjutatud sõnad lahku kirjutada.
4. Igasse lisaritta saab kopeerida eelneva rea andmeid. See lahendus lihtsustab oluliselt kasutaja poolt korduvate andmete sisestamist. Kopeerimine ei toimu automaatselt, kuna see pole paljudes olukordades otstarbekas.
5. Laused on üksteisest eristatavad erinevate taustavärvide kasutamise tõttu. Lihtsustab eelkõige just pikemate tekstide puhul järje leidmist.
6. Tabeli all oleva nupu „Salvesta parandused“ võimaldab muudatuste sisse viimist nii ekraanil kui ka tekstifailis. Vajalik muudetud andmete salvestamiseks.



Joonis 2. Märgendusliidese uus versioon

Märgendusliides kuvab tabelis märgendatud teksti ja selle all TEI P4 kodeerimisreeglitele vastavat märgendust. Kollast värvi kastid tähistavad lisarida, kuhu filoloog saab vajadusel sisestada uue sõne ja sellega seotud morfosüntasktilised andmed.

5.2. Testimine ja täiendamine

Loodud märgendusliidest on testitud koos juhendajate ning filoloog Pille Esloniga. Selle tulemusena jõuti järeldusele, et on vaja sisse viia mõned täiendused:

1. TEI P4 kodeeringureeglite järgi ja XML märgenduskeeles salvestatud teksti märgendusliideses kuvamise võimalus. Annab kasutajale kiire ülevaate, kuidas näeb märgendusliidese lõpptulemus välja.
2. TEI kodeeringus XML failide salvestamise võimalus kasutaja valikul soovitud asukohta.
3. Võimaldab kasutajal kiirelt ja mugavalt märgendusi sisaldavast failist endale koopia teha.

4. Olemasolevate tekstiandmete korrigeeritud andmete lahtritesse kopeerimise võimalus. Sekundaarne viis andmeid kopeerida, kus kopeeritakse rea nelja esimese lahtri sisu neile vastavatesse järgmistesse lahtritesse, mis on mõeldud muudatuste salvestamiseks.
5. Piiramata hulgal lisaridade tekitamise võimalus iga sõne alla. Varem oli võimalus lisada vaid üks rida. Täiendus on vajalik, et olla valmis olukordadeks, kus valesti kirjutatud sõna peab jagunema enam kui kaheks.

Faili salvestatud XML märgenduse kuvamiseks sai välja tuldud lahendusega, kus vastavalt salvestatud failide hulgale kuvatakse 0 kuni 2 nuppu, mille vajutamisel edastatakse nupuga seotud mitmerealisse tekstivälja vastav TEI märgendus. Neid tekstivälju koos märgendusega ei kuvata automaatselt, et vältida lehekülge põhjuseta pikemaks tegemist, kuna suure hulga tekstiandmete tabelis kuvamine võib lehe üpris pikaks venitada.

Selleks, et kuvada nupud vaid siis, kui nende järele vajadus tekib, on kasutatud TAL atribuuti *tal:condition*, mille abil saab kirja panna, mis tingimustel neid nuppe kuvatakse. Nupud laetakse ja kuvatakse vaid siis, kui nende nuppude vajutamisel kuvatav märgendus eksisteerib.

Tekstiväljade sisse õige teksti paigutamise eest hoolitsevad Pythoni funktsioonid ja TAL mallkeel. Nende väljade kuvamiseks olevad nupud kasutavad JavaScripti funktsiooni, mis toob vastavalt antud sisendile nähtavale kas parandamata või korrigeeritud teksti XML märgendust sisaldava mitmerealise tekstivälja. JavaScript on siinkohal kasutusel just seetõttu, kuna sellised ülesanded on kõige lihtsamini teostatavad just seda skriptikeelt kasutades.

Võimaldamiseks kasutajal märgendust sisaldavat XML faili mugavalt oma valitud asukohta salvestada, sai kasutusele võetud lihtne lahendus, kus kasutajaliideses kuvatakse viit vastava XML faili aadressil.

Teise kopeerimismeetodi sisse viimiseks sai aluseks võetud esimese meetodi kood ning arendatud seda sobivale kujule, et kopeerida andmeid ainult ühe rea piires. Seoses selle muudatusega sai kasutajaliidest ühtlasi täiendatud ühe lisanupuga, mis tähistab vastavat kopeerimisviisi. Erinevaid kopeerimisviise tähistavatele nuppudele sai lisatud kopeerimissuunda iseloomustavad nooled.

5.3. Arendusprotsessi tulemus

Arendusprotsessi tulemuseks on märgendusliides, mis suudab lugeda, töödelda ja kuvada EVKK morfosüntaktiliselt märgendatud tekste. Loodud rakendus võimaldab filoloogidel korrigeerida eesti keele süntaksianalüsaatori tööd, parandada vigast ning mitmesusi sisaldavat märgendust. Korrigeeritud märgendusega tekst salvestatakse peale töötlemist nii originaal- kui ka parandatud versioonis välisesse tekstifaili.

Arenduse käigus täiendatud märgendusliideses kuvatakse vaid seda informatsiooni, mis on oluline. Näiteks tabel märgenduste jaoks laetakse ja esitatakse kasutajale vaid siis, kui viimane on laadinud teksti ning see on edukalt süntaksanalüsaatorist läbi käinud. Liides on tahtlikult minimaalse kujundusega, tõstes esile vaid olulisemad funktsioonid.

Märgendusliides koosneb kolmest tekstifailist ja kümnest välisest meetodist (*External Method*) Zope'i keskkonnas. Kõige olulisemaks failiks on markup_interface.tal, mis vastutab kogu märgendusliidese kuvamise eest, sh selleks vajalike funktsioonide käivitamise eest. Teistest failidest sisaldab üks JavaScripti koodi ning teine lehe kujundust CSS-is. Iga väline meetod vastab ühele Pythoni funktsioonile. Mahukamad neist on:

1. text_to_word_data. Vastutab süntaksanalüsaatoriga suhtlemise ning analüüsitud teksti andmete massiivi paigutamise eest.
2. word_data_to_tei. Võtab massiivis paiknevad andmed, genereerib neist XML puu ja salvestab selle tekstifaili.

Loodud rakendus on plaanis avalikustada veebiaadressil http://evkk.tlu.ee/Search/search_margendid.html.

5.4. Edasised plaanid

Üks olulisemaid plaane märgendusliidese arendamisel seisneb EVKK-s märgendatud tekstide importimise-eksportimise võimaluses. Selle tulemusena saab liides täiendavaid funktsionaalsusi juurde ning on kasutatav ka väljastpoolt imporditavate tekstide morfosüntaktiliseks analüüsiks, märgendite korrigeerimiseks ja TEI vormingus

salvestamiseks. Metaandmete lisamisega seoses on ühtlasi plaan täiustada TEI P4 standardite järgimist, mis tähendab TEI skeemide kaasamist ning uute elementide XML failide hierarhiasse lisamist. See on oluline mitte ainult eesti keele süntaksianalüsaatori testimise ja veelgi efektiivsemaks muutmise eesmärgil, vaid perspektiivis rakendamiseks ka automaatselt töötava õigekirjakorrektorina.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli arendada EVKK märgendusliidest niikaugele, et see võimaldaks korpuse veebikeskkonnas töötava süntaksanalüsaatori märgendatud teksti korrigeerimist ja TEI kodeeringus salvestamist. See täindab korpuse kasutajaliidest uute võimaluste lisamisega, võttes nende rakendamisel arvesse nii funktsionaalsuste avardumist kui ka kasutajamugavust.

Töös on avatud rida teemaga seotud mõisteid nagu korpused ja nende liigid. Pikemalt on peatutud Eesti vahekeele korpusel, mille jaoks märgendusliides on loodud ning plaanitakse tulevikus ühendada. Lühidalt on selgitatud ka EVKK veaklassifikatsiooni ning märgenduse olemust.

Eraldi peatükk on pühendatud märgendusliidese arendusprotsessi käigus sooritatud valikutele, sh nende valikute põhjuste analüüsimisele. Eraldi on välja toodud testimise käigus vajalikuks osutunud täiendused ning nende rakendamine.

Valminud märgendusliidese kood koos vajamineva tarkvara ning paigaldusõpetusega on bakalaureusetööle lisatud DVD plaadil.

Tänuavaldused

Autor soovib tänada Pille Eslonit, kes oli suureks abiks töös kasutatud materjalide leidmisel ning autori mõtteavalduste keelendamisel. Siirad tänusõnad kuuluvad juhendajatele Erika Matsakule ja Jaagup Kipparile suunavate nõuannete ja heatahtlike märkuste eest. Suur aitäh Gerli Rohile, kes on autorit innustanud ja igati motiveerinud – temata oleks see töö olemata.

Summary

Development and Possible Applications of Estonian Intermediate Language Corpora Markup Interface

This bachelor's thesis focuses on the development process of Estonian Intermediate Language Corpora markup interface. It is a continuation of authors work on the interface started as a project for his previous thesis.

As the texts in the corpora are analyzed by a syntax analyzer, that's still being improved and which does not always produce annotated text with correct grammatical data, there is need for a interface that allows manually fixing made errors.

The primary goal of this thesis is to produce a working markup interface that allows users to manually correct the grammatical data from the text annotated by syntax analyzer integrated to Estonian Intermediate Language Corpora. The interface also has to be able to generate XML tree containing data that follows TEI guidelines and save it to an external file.

Final result of this bachelor's thesis is a working markup interface that is capable of reading, processing and displaying annotated text created by the syntax analyzer. It is also capable of producing XML files that contain the grammatical data from these texts. Users are able to correct the data, if needed, with changes being saved in a separate XML file.

Kasutatud kirjandus

1. Eslon, P. & Metslang, H. (2007). Õppijakeel ja eesti vahekeele korpus. – Eesti Rakenduslingvistika Ühingu Aastaraamat 3, 99–116.
2. Eslon, P. (2007). Õppijakeelekorpused ja keeleõpe. – Tallinna Ülikooli keelekorpuste optimaalsus, töötlemine ja kasutamine, 87–120.
3. Goyvaerts, J. (2009). Regular Expression Tutorial. Viimati vaadatud 30.12.2011, aadressil http://www.regular-expressions.info/tutorial.html
4. Leech, G (1993). Literary and Linguistic Computing 8.
5. McEnery, T. & Wilson, A. (2001). Corpus Linguistics (2nd Edition).
6. Python Software Foundation (2011). Regular Expression HOWTO. Viimati vaadatud 29.12.2011, aadressil http://docs.python.org/howto/regex.html
7. Python Software Foundation (2011). 19.13. xml.etree.ElementTree – The Elementree API. Viimati vaadatud 29.12.2011, aadressil http://docs.python.org/library/xml.etree.elementtree.html
8. Richter, S. (n.d.). lxml – XML and HTML with Python. Viimati vaadatud 29.12.2011, aadressil http://lxml.de/index.html
9. Sultsing, M. (2010). Tekstianalüsaatori automaatmärgendusele kasutajaliidese loomine.
10. Text Encoding Initiative (n.d.). TEI: Text Encoding Initiative. Viimati vaadatud 28.12.2011, aadressil http://www.tei-c.org/index.xml
11. Text Encoding Initiative (n.d.). TEI: Text Encoding Initiative. Viimati vaadatud 28.12.2011, aadressil http://www.tei-c.org/Support/Learn/index.xml
12. VAKO (n.d.). VAKO - Eesti vahekeele korpuse keeletarkvara ja keeletehnoloogilise ressursi arendamine. Viimati vaadatud 30.12.2011, aadressil http://www.keeletehnoloogia.ee/projektid/vako/
13. Vallaste, H. (n.d.). e-teatmik. Viimati vaadatud 02.01.2011, aadressil http://vallaste.ee/