Tallinna Ülikool
Informaatika Instituut

Vigade märgendus Eesti vahekeele korpuse tekstides

Seminaritöö

Autor: Harry Nõmmann
Juhendaja: Jaagup Kippar

Tallinn 2014

Autorideklaratsioon

Deklareerin, et käesolev seminaritöö 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.

Sisukord

Sissejuhatus 4

1. Eesti vahekeele korpus 5

2. Tehnilised võimalused 7

2.1 Kasutatud tehnoloogiad 7

2.2 jsTree puukujulise andmestruktuuri loomisel 8

2.3 TreeView 12

2.4 Dynatree 13

3. Muudatuste kirjeldus 14

3.1 Makro codesTreePrinterDocument 14

3.2 jsTree rakendamine 16

3.3 Märkimine andmepuus ja informandi sektsiooni peitmine 18

Kokkuvõte 20

Kasutatud allikad 21

Sissejuhatus

Seminaritöö teema valikul lähtusin soovist teha midagi praktilist ning vajalikku ja seejuures oma programmeerimisalaseid oskusi parendada. Teema pakkus välja juhendaja ning oluliseks ja otsustavaks sai see, et teema on aktuaalne ja lahendusest ollakse huvitatud, lahendusest tuleks kasu.

Seminaritöö eesmärgiks on tõsta olemasoleva vigade märgendusmooduli kasutamismugavust Eesti vahekeele korpuse tekstides. Eesmärki põhjendab vajadus kasutajasõbralikuma lahenduse järele.

Ülesanneteks on tutvustada Eesti vahekeele korpust ja korpuseid keeleteaduses üldiselt.
Välja tuua praeguse vigade esituse positiivsed ja negatiivsed pooled ning kavandada mugavam lahendus. Tutvustada jsTreed ja põhjendada selle valikut ja kasutamist veebilehel vigade puu kuvamiseks ning välja tuua mõned alternatiivid. Kirjeldada arendustööd ning tulemust sihtgrupi juures katsetada.

Seminaritöö teemast tulenevalt on meetodiks praktilise lahenduse loomine, mugavdades Eesti vahekeele korpuse tekstides olevate vigade esitust veebilehel. Töötavat tulemust saab vaadata aadressil
http://greeny.cs.tlu.ee:18186/korpus/korpus/Documents/doc_897182025872_item/Siim.html.

1. Eesti vahekeele korpus

Eesti vahekeele korpuse tutvustus

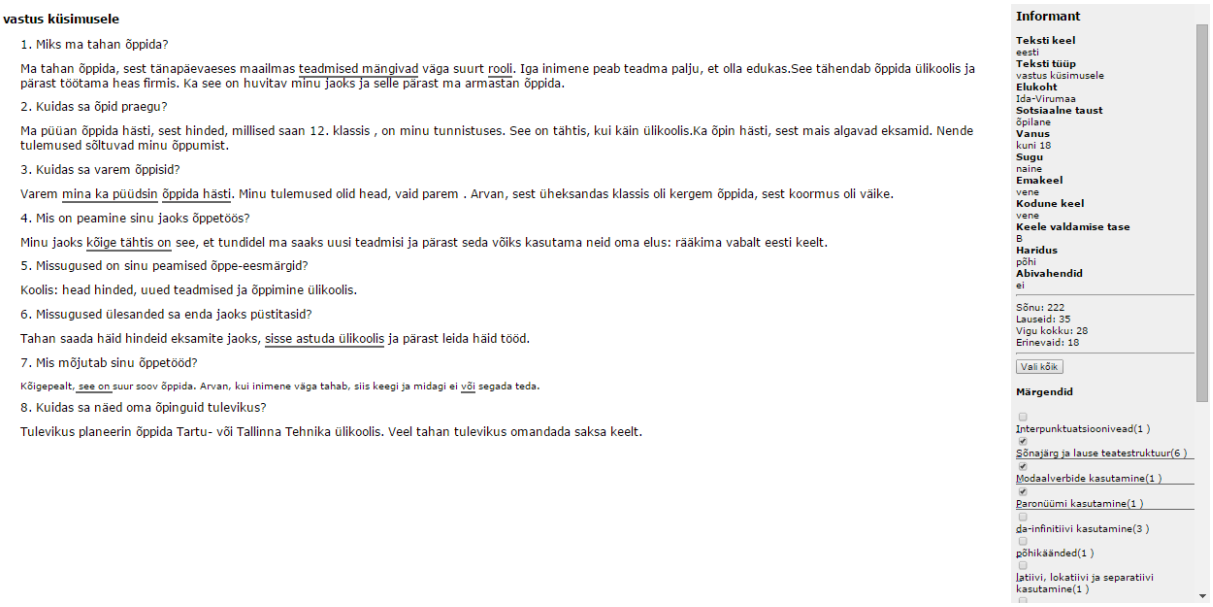
Eesti keele seletava sõnaraamatu kohaselt on korpus teatud tunnuste järgi süstematiseeritud andmekogu veebis. Keeleteaduses on sõna korpus all tavaliselt mõeldud keeleainese kogumikku, mida kasutatakse uurimistöös materjalina. Korpustesse kuuluvad tekstid on valitud eesmärgipäraselt, nii, et nendest koosnev tervik annaks tõepärase pildi kogu keelest (Muischnek, kuupäev puudub).

Tallinna Ülikooli eesti vahekeele korpus on õppurite kirjalike tööde kogu. See on avatud ehk monitorkorpus, mis tähendab, et sellesse saab tekste pidevalt lisada. Praegune versioon on valminud Tallinna Ülikooli filoloogide, haridustehnoloogide ja informaatikute koostööna. Jälgida saab ka mingi sõna vigaseid esinemisnäiteid ning sama vealiiki ühes või kõigis korpuses olevates tekstides. On võimalik seostada infot keelevigade kohta õppuri sotsiaalse päritolu, staatuse, hariduse, soo, vanuse ja keele valdamise tasemega.
Korpust saab kasutada:

- Empiirilist ja rakenduslikku laadi uurimistöös
- Tulevaste õpetajate ja lingvistide koolitamisel
- Tegevõpetajate täiendõppes

Eriõigused antakse registreeritud kasutajale, andmehaldurile ning programmeerijale, aga korpuse funktsioone saavad kasutada kõik. (Eslon, 2014, 438)

Olemasolev



Joonis 1. Olemasolev vigade esitus teksti vaatelehel

Eelised

Olemasolevat märendusmoodulit on lihtne ja kerge kasutada. Laadimine ja kasutaja tegevustele reageerimine on kiire.

Puudused

Kui tekstis esineb rohkem vigu, muutub vigade loetelu pikaks, mis teeb vigade vaatamise, märkimise lehel ebamugavaks, sest tegevuste vahepeal tuleb lehte üles-alla kerida.

Vead on järjestatud sellises järjekorras nagu nad tekstis märgiti. See tähendab, et vead pole organiseeritud ja võib juhtuda, et konkreetse veatüübi leidmiseks tuleb kogu nimekiri läbi vaadata.

2. Tehnilised võimalused

Selles peatükis tutvustan ja võrdlen seminaritöö käigus ja vigade märgendusmoodulis kasutatavaid ja kasutamiseks sobivaid tehnoloogiaid. Tutvustan andmepuu esitamiseks kasutatud jsTreed, mõnda selle alternatiivi ning põhjendan oma valikut.

2.1 Kasutatud tehnoloogiad

PHPTAL

PHPTAL on XML/XHTML malli teek. See on idee Zope'i kogukonna poolt, mis seisneb esitustegevuste ümberpaigutamises XHTML atribuutide sisse, selle asemel, et kasutada lihtmärgendeid või -elemente. -koduleht

JavaScript

Netscape'i poolt välja töötatud skriptikeel, mis võimaldab veebiautoritel luua interaktiivseid veebisaite. JavaSctipt suudab suhelda HTML-keeles kirjutatud lähtekoodiga ja võimaldab muuta veebilehed dünaamiliseks. JavaScript on avatud keel, mille kasutamiseks pole vaja osta litsentsi (Vallaste).

jQuery

jQuery on kompaktne ja paljude võimalustega JavaScripti teek, mis lihtsustab JavaScriptis programmeerimist.

JSON

JSON (JavaScript Object Notation) on süntaks andmete talletamiseks ja vahetamiseks, mida on inimesel kerge lugeda ja kirjutada. JSON on lihtsamini kasutatav XMLi alternatiiv.

2.2 jsTree puukujulise andmestruktuuri loomisel

Informaatikas nimetatakse puuks sellist andmestruktuuri, kus iga element on ühendatud ühe või mitme temast allpool asuva elemendiga. Sageli kasutatakse väljendit “pööratud puu” - puu, mis hargneb ülevalt alla (Vallaste).

Puus enamesinevad elemendid:

- juur - kõige ülemine element puus
- vanem - mingi elemendi otsene ülemelement
- laps - mingi elemendi otsene alamelement
- leht - lasteta element

Ülesande lahendamiseks otsustasin jsTree kasuks, sest olen sellega eelnevalt koolitundides kokkupuutunud. jsTree eelis mõne teise alternatiivi ees on see, et tema arendustöö jätkub endiselt ning Google jsTree grupis antakse pidevalt probleemide ja küsimuste puhul nõu, abi, mis näitab aktiivset kogukonda.
jsTree on avatud lähtekoodiga tasuta jQuery plugin, mis võimaldab luua interaktiivseid puusid. Seda on lihtne konfigureerida, kasutada ning teatud osasid omale meelepärasemaks muuta. Kuna kasutatakse jQuery sündmuste süteemi on erinevate tegevuste sidumine puuga tuttav ja lihtne.

Andmepuule jsTree lisamiseks tuleb jsTree selle kodulehelt allalaadida. Kõik vajalikud failid asuvad vaikimisi dist kaustas. Selleks, et jsTree töötaks ja andmepuu omandaks jsTree kujunduse, peab fail sisaldama viiteid jQueryle ning jsTree CSSi ja JavaScripti failidele (vt Koodinäide 1).

```
<link rel="stylesheet"
href="dist/themes/default/style.min.css"/>
<script src="dist/libs/jquery.js"></script>
<script src="dist/jstree.min.js"></script>
```

Koodinäide 1. jsTree lisamine

jsTreele on võimalik andmeid kahel moel ette anda:

1. kasutades allikana HTMLi
2. kasutades allikana JSONi

Mõlemal juhul tuleb kasutada konteinerit, kuhu hiljem valitud meetodiga andmepuu tekitada ning sellele id määrata. Kasutades esimest meetodit luuakse andmepuu <li> ja <ul> märgenditega (vt Koodinäide 2).

```
<div id="puu_ID">
   <ul>
      <li>juur Y
         <ul>
            <li>laps B</li>
            <li>laps A</li>
         </ul>
      </li>
      <li>juur X</li>
   </ul>
</div>
```

Koodinäide 2. Andmepuu loomine, kasutades allikana HTML andmeid

Ning viimaks tuleb lisada skript (vt Koodinäide 3), mis olemasolevale andmepuule jsTree rakendab ning selle nähtavaks teeb (vt Joonis 2).

```
$('#puu_ID').jstree();
```

Koodinäide 3. Skript jsTree rakendamiseks

Soovi korral on lihtne puu funktsionaalsust erinevate pluginatega tõsta (vt Koodinäide 4). Täiendades mõnevõrra algset skripti, saame puule lisada mitmetest võimalustest näiteks märkeruudud, *context-menu* ning automaatse sorteerimise (vt Joonis 3).

```
$('#puu_ID').jstree({
   "core" : {"check_callback" : true},
   "plugins" : ["checkbox", "contextmenu", "sort"]
});
```

Koodinäide 4. Pluginate lisamine esimesele variandile

Töötavat näidet saab vaadata ja proovida aadressil http://www.tlu.ee/~harry/jsTree/HTML/

Kasutades teist meetodit – võttes andmed JSONi kaudu jääbki HTMLi ossa vaid konteiner. Samasuguse andmepuu loomiseks tuleb aga lisada skript (vt Koodinäide 5).

```
$('#puu_ID').jstree({ 'core' : {
   'data' : [
      'juur X',{
         'text' : 'juur Y',
         'children' : ['laps B','laps A']}
]}});
```

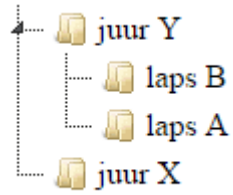
Koodinäide 5. Andmed JSONist.

Ning sarnaselt esimesele meetodile on samasuguse tulemuse saamiseks pluginate lisamine üsnagi lihtne (vt Koodinäide 6).

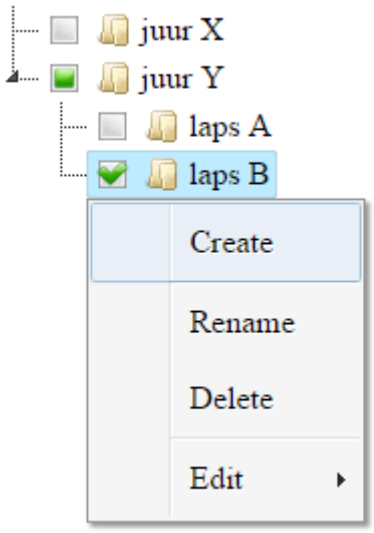
```
$('#puu_ID').jstree({ 'core' : {
   'data' : [
      'juur X',{
        'text' : 'juur Y',
        'children' : ['laps B','laps A']}
   ],
   "check_callback" : true},
   "plugins" : ["checkbox", "contextmenu", "sort"]
});
```

Koodinäide 6. Pluginate lisamine teisele variandile

Töötavat näidet saab vaadata ja proovida aadressil http://www.tlu.ee/~harry/jsTree/JSON/



Joonis 2. Vaikimis funktsionaalsustega puu



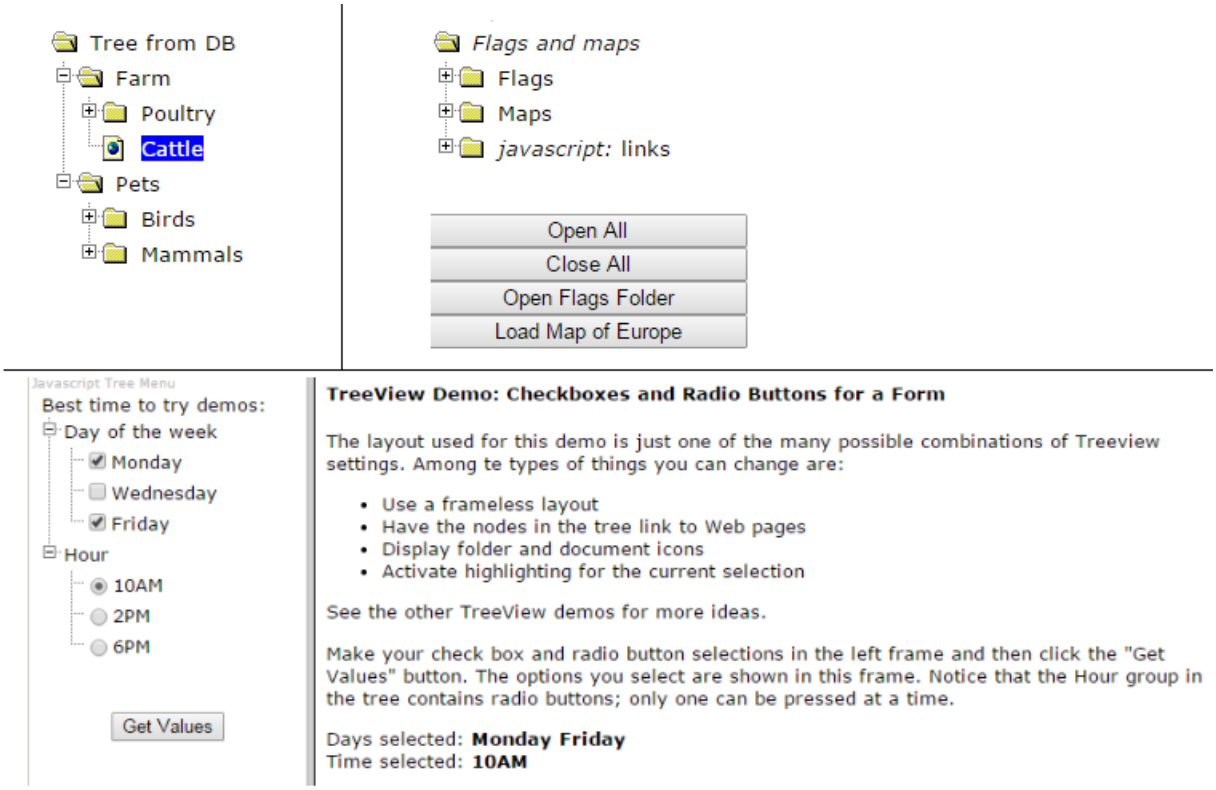
Joonis 3. Puu, mille funktsionaalsust on pluginatega tõstetud

2.3 TreeView

TreeView võimaldab, nagu jsTree, andmeid hierarhiliselt esitada (vt Joonis 4). TreeView'd on arendatud alates selle avalikustamisest aastal 1997. Kodulehel on välja toodud mõningad Treeview head omadused, näiteks:

- Kiirus – TreeView tuleb toime ka suuremate andmepuudega
- Vanade brauserite toetus
- Lihtne konfigureerimine

Miinusena näen TreeView puhul seda, et kasutajatugi on vaid neile, kes ei kasuta tasuta varianti vaid on maksnud (umbes 30 eurot), seega enamus kasutajaid või katsetajaid peavad unikaalsete või vähemalt ebatavaliste probleemide puhul ise lahendust otsima hakkama. Lisaks leian, et konfigureerimine pole niivõrd lihtne nagu esialgu, lugedes kodulehte, tunduda võib.



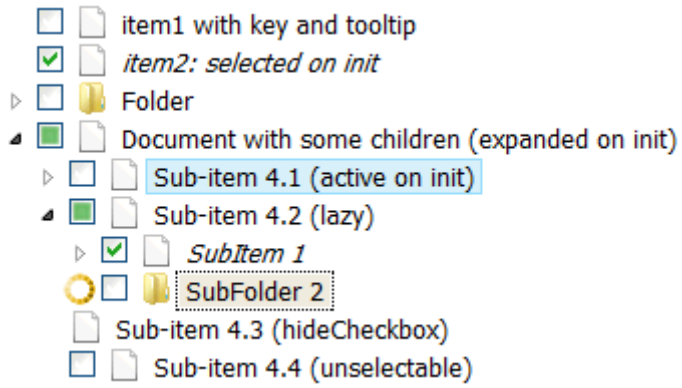
Joonis 4. Mõned näited TreeView kodulehelt.

2.4 Dynatree

Dynatree on, nagu jsTree, jQuery plugin, mis võimaldab JavaScripti kasutades luua HTML puu (vt Joonis 5). Dynatree on optimeeritud suurte dünaamiliste puude jaoks.Hetkel on väljas viimane versioon, mida enam ei täiendata. Tulevikus saab edasiarendatavaks variandiks Fancytree. Mõned kodulehel välja toodud head omadused:

- Toetab laiska laadimist ning Ajaxit
- Märkeruudud ja hierarhiline selektimine
- Toetab lohistamist
- Lähtestub HTML koodist, JSONi või JavaScripti objektidest

jsTree eeliseks on aktiivsem kogukond ning parem dokumentatsioon.



Joonis 5. Dynatree näide kodulehelt.

3. Muudatuste kirjeldus

Eesti vahekeele korpust kasutatakse aktiivselt ja seetõttu peab arendustöö toimuma testkeskkonnas. Testkeskkonnaks sai Informaatika instituudi testserveris greeny.cs.tlu.ee asuv Zope server, mille kasutamiseks mulle õigused anti.

Eelnevalt on Eesti vahekeele korpuse vigade märgendusmooduliga tegelenud ja arendanud teiste seas Siim Medijainen. Seetõttu soovitas juhendaja keskkonnaga tutvumiseks lugeda tema diplomitööd „Eesti vahekeele korpuse tekstide märgendusmooduli arendamine“. Esialgu läks suur osa ajast koodi ja keskkonnaga üldiselt tutvumiseks. Lugesin diplomitööd ja koodi ning püüdsin mõista kuidas erinevad funktsioonid omavahel seotud on.

Arendustöö võib üldistades või kokkuvõttes jagada kaheks osaks. Esialgu oli vaja luua olemasoleva vigade nimekirja asemele puukujuline ja vaid tekstis esinevatest vigadest koosnev nimekiri. Seejärel loodud veapuule jsTree rakendada ning siduda tekstiga – veapuule käsklusi andes peab tekstiosa vastavalt muutuma.

3.1 Makro codesTreePrinterDocument

Kõigist tekstis olevatest vigadest koosneva andmepuu loomiseks sai macros.pt faili lisatud makro nimega **codesTreePrinterDocument**, mis oli juba varem olemas ja kasutuses luues andmepuu leheküljel http://evkk.tlu.ee/Marks/global_marks/marks_public.html. See oli aga esialgu ebasobilik, sisaldades ja kuvades kõiki Eesti vahekeele korpuses leiduvaid vigu, luues iga teksti kõrvale väga pikk nimekiri kõigist vigadest, sõltumata sellest, kas neid vigu ka tekstis esineb.

Ülesande lahendamiseks sai algset, suurt nimekirja genereerivat varianti koos juhendajaga kohendatud. Lisatud sai PHPTAL tingimus, millega sorteeritakse makros kogu andmepuud nii, et alles jääksid vaid need vealiigid, mis tekstis esinevad või millel on vähemalt üks alamvealiik, mis tekstis esineb.

Teine variant oleks olnud muuta Document.py faili – lisada funktsioon või muuta vana, mis juba eelnevalt leiaks ja eemaldaks sellised vead ja liigid, mis tekstis mingil moel ei esine.

Makro käib läbi kogu vigade massiivi käsuga `tal:repeat="code cont/getCodes"`. Rakendatakse konditsioon, milleks on, et `context.countUsedCodeTree(code, request)` ei tohi võrduda nulliga, mis tähendaks, et tekst ei sisalda antud vealiiki või selle mõnda alamvealiiki. Nulli korral võetakse kõikide vigade massiivist järgmine viga ja korratakse, nullist suurema puhul lisatakse see aga andmepuusse ning lisatakse atribuudid, mille järel võetakse samuti massiivist ette järgmine viga ning korratakse kogu protsessi.

Vealiigi nime, esinemiste arvu tekstis ning selle vealiigi alamvigade esinemise arvu tekstis saab kätte ning esitab andmepuus järgnev osa koodist:

```
tal:content="python: code.getTitle()+' - '+
str(context.countUsedCode(code.getCodeId(), request)) +' , '+
str(context.countUsedCodeTree(code, request))"
```

Koodinäide 7. Vea nime ja esinemiste arvude saamine.

Igale veale lisatakse ka atribuudid “style”, “href”, “name”, “id”, “docolor” järgnevaga:

```
tal:attributes="style python:view.test(dele, 'font-
style:italic' );
   href string:${code/absolute_url}/markdown.html;
   name python:code.getCodeId(); id python:code.getCodeId();
   docolor python:dUsed[2][2]"
```

Koodinäide 8. Atribuutide lisamine veale.

3.2 jsTree rakendamine

Lihtsam oli olemasolevale jsTree külge liita, seega andmed HTMList saada. Lisaks olin varem jsTreed just sel moel enam kasutanud. Tuli välja, et selleks, et jsTree andmepuu peal töötaks, ei tohi andmepuu sisaldada märgendeid ning märgendeid lihtsalt ei näidata. Aga kuna esialgne makro just neist märgenditest koosneva andmepuu lõigi, siis tuli ülesehitust muuta.

Üldiselt piisas vastavates kohtades ja märgendite asendamisest <li> ja <ul> märgenditega. Kasuks tuli ka PHPTAL, millega sai mõned märgendid, mille olemasolu segas jsTree töötamist andmepuul, ära peita, kasutades meetodit `tal:omit-tag=""`.

Kui jsTree andmepuu peal tööle hakkas, siis ilmnes, et puus olevale veatüübile vajutades ei avane sellele määratud hüpertekstiviide. See oli aga jällegi probleem, kuna algne variant seda funktsionaalsust võimaldas.

Lahenduseks sai funktsiooni lisamine JavaScripti faili, mis muudab aktiivse akna hüpertekstiviiteks selektitud veatüübi hüpertekstiviite (vt Koodinäide 9). Tulemusena avatakse klõpsamise peale soovitud lehekülg.

```
jQuery("#puuvaade0").on("select_node.jstree",function(e,data){
     window.location.href = data.node.a_attr.href;
});
```

Koodinäide 9. Hüpertekstiviite avamine

Eelneva probleemi lahendamisega tuli välja järgmine. Nimelt märkeruudu märkimisel loetakse selleks tehtavat hiireklõpsu justkui see oleks tehtud nii märkeruudule kui ka sellele järgnevale veatüübi tekstile. Sellel tekstil on aga lisaks ka ülaltoodud funktsioon - avada temale peale klõpsates leht, kus on välja toodud kõik antud veatüübi esinemised erinevates tekstides. Järelikult oli tekkinud probleem, kus märkeruudu märkimisel mindi kohe teisele lehele, muutes märkeruutude olemasolu mõttetuks.

Lahenduseks sellele probleemile sai kahe funktsionaalsuse eemaldamine. Piisas kui märkida `"whole_node" : false`, mis kontrollib seda, et vealiigile klõpsates ei märgitaks tema

eesolevat märkeruutu. Ning `"tie_selection" : false`, mis tähendab, et märkeruudu märkimine pole enam seotud üldise puuga vaid puu seesmise massiiviga. Tulemuseks on, et veatüübi lehele minnakse vaid siis, kui vajutatakse veatüübi peale ning märkeruutu saab märkida vaid siis, kui selle peale klõpsata. Eemaldatud sai ka *three-state* loogika ning märkeruutude lisamiseks vajaminev kood näeb välja selline:

```
jQuery("#puuvaade0").jstree({
        "checkbox" : {
            "whole_node" : false,
            "tie_selection" : false,
            "three_state" : false
        },
        "plugins" : [ "checkbox" ]
    });
```

Koodinäide 10. Märkeruutude loomine

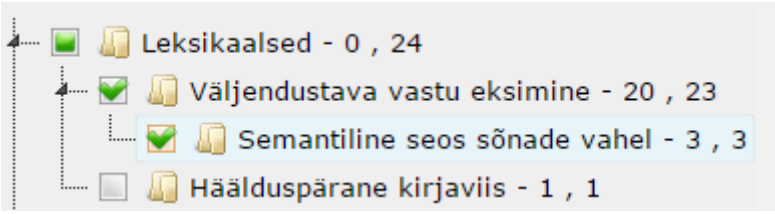
Järgmiseks oli vaja siduda märkeruudud tekstiga nii, et kui märkeruut märkida, siis tuleks tekstis vastavate vigade alla joon. Hetkel märkeruutude märkimisele mingit tegevust ei järgnenud. Selleks, et märkeruutudelt reaktsioon saada pidi javascripti faili lisama kaks lühikest funktsiooni (vt Koodinäide 11). Esimene, mis jälgiks märkeruutudesse märke tegemist ja teine, mis jälgiks märke mahavõtmist. Mõlemad funktsioonid saavad kätte märgitud veatüübi atribuudi „name“ väärtuse ning saadavad selle vastava käsuga (märkimisel 1 ja märke eemaldamisel 11) funktsiooni **updateMarkupSchema**, mis varem juba olemas oli, aga uue ja muutunud süsteemi tõttu kohandamist vajas.

```
jQuery("#puuvaade0").on("check_node.jstree", function(e,
data){
   updateMarkupSchema(data.node.a_attr.name, 1);
});
jQuery("#puuvaade0").on("uncheck_node.jstree", function(e,
data){
   updateMarkupSchema(data.node.a_attr.name, 11);
});
```

Koodinäide 11. Märkeruutude interaktiivseks tegemine

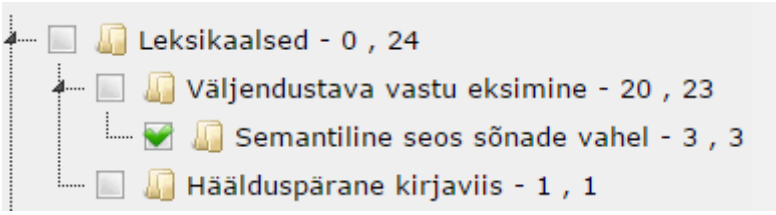
3.3 Märkimine andmepuus ja informandi sektsiooni peitmine

jsTreel on valik `three_state` vaikimisi tõene. See loob olukorra, kus puu juure märkimisel märgitakse automaatselt ka kõik selle juure alamelemendid. Sõltuvalt alamelementide arvudest märgitakse elemendi märkimisel tema vanem kas pool- või täismärkega (vt Joonis 6).



Joonis 6. Märkeruudu märkimine, kui three_state on lubatud

Kuna `three_state` ei muuda automaatselt märgitud vigade olekut, siis ei tooda neid ka tekstis esile ning kui soovida antud näites märkida "Väljendustava vastu eksimine", siis peab esmalt selle eest märke ära võtma ning siis uuesti märke tegema. Selline lahendus oleks aga segadust tekitav, mistõttu otsustasin selle hetkel eemaldada (vt Joonis 7). Koodi pidi selleks lisama järgneva: `"three_state" : false.`

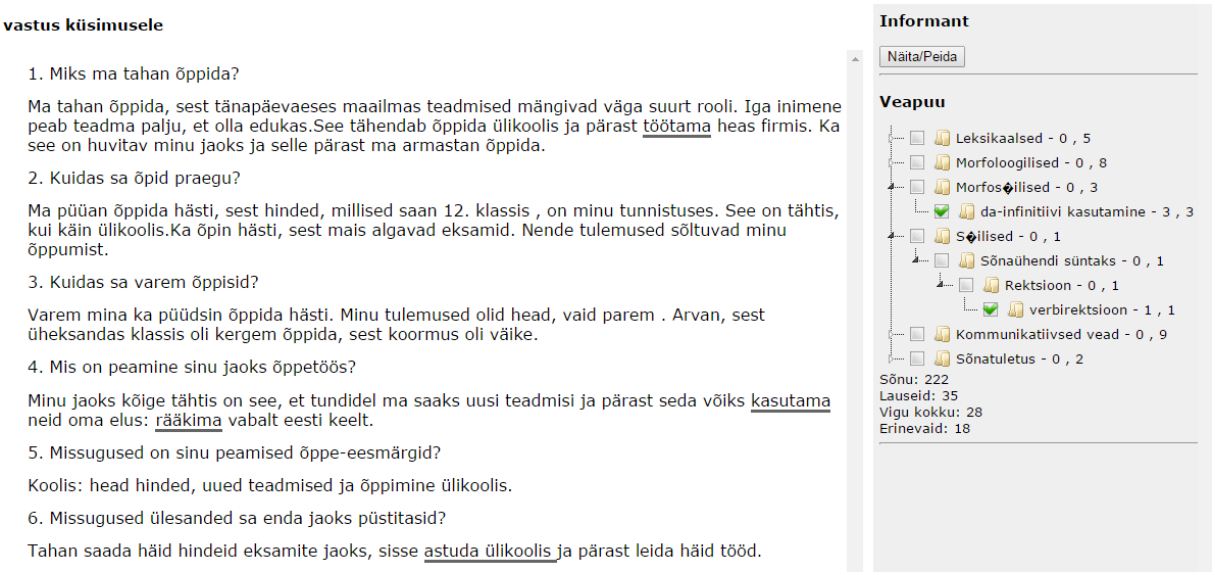


Joonis 7. Märkeruudu märkimine, kui three_state on keelatud

Lisasin nupu, millega saab informandi sektsiooni näidata ja peita, eesmärgiga veelgi ruumi kokku hoida. Kuna seda informatsiooni kogu aeg näha pole vaja ja nähtaval olles võtab see palju ruumi, siis on muudatus põhjendatud.

```
function toggle_visibility(id) {
   var e = document.getElementById(id);
   if(e.style.display == 'block')
      e.style.display = 'none';
   else
      e.style.display = 'block';
}
```

Koodinäide 12. Koodiosa, mis muudab nupu interaktiivseks



Joonis 8. Uus vigade märkimine ja puukujuline esitamine

Kokkuvõte

Seminaritöö peamised eesmärgid ja ülesanded said täidetud. Õnnestus viia tekstis esinevad veanimetused puu kujule ja seejuures eelnevalt olemasolev funktsionaalsus säilitada ning rakendada jsTree veelgi mugavamaks kasutamiseks.

Praktilise lahendusena loodud uus vigade esitusviis muudab vigade märkimise ja esitamise mugavamaks ja loogilisemaks. Muudatuste tõttu on lehe parem osa, kus asuvad informandi sektsioon ja veapuu, lühem ja kompaktsem, hoides kokku ruumi ja tõstes seeläbi mugavust.

Töö käigus sain uusi kogemusi ja teadmisi veebipõhise rakenduse aredamisest eelkõige JavaScripti ja selle jQuery teeki kasutades. Tutvusin põhjalikumalt pluginaga jsTree ning sain kogemuse selle oskuslikuks kasutamiseks.

Kasutatud allikad

PHPTAL. (2005). *Introduction.* Loetud 16. 11. 2014 aadressil http://phptal.org/.

Vallaste, H. (2000). *e-Teatmik.* Loetud 16. 11. 2014 aadressil http://www.vallaste.ee/.

jQuery. (kuupäev puudub). *jQuery*. Loetud 16. 11. 2014 aadressil http://jquery.com/.

JSON. (kuupäev puudub). *JSON.* Loetud 16. 11. 2014 aadressil http://www.json.org/.

jsTree. (kuupäev puudub). *jsTree*. Loetud 16. 11. 2014 aadressil http://www.jstree.com/.

GubuSoft. (2006). *TreeView.* Loetud 16. 11. 2014 aadressil http://www.treeview.net/.

dynatree. (kuupäev puudub). *dynatree.* Loetud 16. 11. 2014 aadressil https://code.google.com/p/dynatree/

Muischnek, K. (kuupäev puudub). *Korpuslingvistika kursus: 1*. Loetud 16. 11. 2014 aadressil http://www.cl.ut.ee/kursused/korp_ling01

Eesti Keele Instituut. (kuupäev puudub). *Eesti keele seletav sõnaraamat.* Loetud 16. 11. 2014 aadressil http://www.eki.ee/dict/ekss/.

Medijainen, S. (2011). *Eesti vahekeele korpuse tekstide märgendusmooduli arendamine* (diplomitöö). Haapsalu, Eesti: Tallinna Ülikooli Haapsalu kolledž.

Eslon, P. (2014, juuni). Eesti vahekeele korpus. *Keel ja kirjandus* 6, 438. Tallinn, Eesti: SA Kultuurileht.