TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Katrin Valdson

Mustripõhine informatsiooni eraldamine Eesti kohtulahenditest

Magistritöö (30 EAP)

Juhendaja(d): Raul Sirel
Aleksandr Tkatšenko

Tartu 2016

Mustripõhine informatsiooni eraldamine Eesti kohtulahenditest

Lühikokkuvõte:

Käesolevas töös rakendatakse mustripõhise informatsiooni eraldamise meetodeid Eesti kohtulahenditele, mis on senimaani vähekasutatud eestikeelne ressurss. Töö eesmärgiks on luua andmebaas, mille põhjal saaks arendada tarkvara, mis võimaldaks efektiivselt relevantseid kohtulahendeid otsida ning otsingute tulemusi analüüsida. Töö käigus loodud andmebaasis on kohtulahenditest eraldatud huvipakkuvat informatsiooni sisaldavad tekstiosad, mille põhjal on kohtuotsuste tulemusi võimalik analüüsida näiteks maakondade ja kohtute lõikes.

Võtmesõnad:

Tekstikaeve, informatsiooni eraldamine, dokumentide segmenteerimine, kohtulahendid.

CERCS: P175

Pattern based information extraction from Estonian court documents

Abstract:

This thesis describes how pattern based information extraction methods were applied to Estonian court documents, which is an Estonian language resource that until now has remained mostly unused. The goal of the thesis was to create a database, which would enable developing an application that would allow effective searching of relevant court documents and analyzing the results. The result of this work is a database containing facts of interest that were extracted from the court documents. The database enables analyzing decisions written down in the court documents, for example based on counties and the different Estonian courts.

Keywords:

Text mining, information extraction, document segmentation, court documents.

CERCS: P175

Sisukord

1. Sissejuhatus

Interneti laialdase leviku ning järjest kasvava arvu teenuste internetis kättesaadavaks tegemise tõttu suureneb informatsiooni hulk kiiremini kui kunagi varem. Kiire kasvuga on ka vabatekstiliste andmete hulk, milleks on näiteks uudised, sotsiaalmeedia postitused ning kõikvõimalikud avalikuks tehtud dokumendid [1]. Seetõttu on oluliseks saanud ka tekstikaeve valdkond, mis tegeleb selliste andmete töötlemise ja analüüsiga.

Käesolevas töös rakendatakse informatsiooni eraldamise meetodeid, mis on tekstikaeve üheks haruks [2], eestikeelsetele kohtulahenditele, mis on avalikult kättesaadavad Riigi Teataja veebilehelt. Andmekoguks valiti just kohtulahendid, kuna see on tänaseni suurelt jaolt kasutamata eestikeelne ressurss.

Informatsiooni eraldamiseks on kasutatud Python'i EstNLTK teeki [3], mida ei ole informatsiooni eraldamise jaoks eelnevalt piisavalt testitud, seega on käesolev töö selle üks esimesi põhjalikumaid rakendusi. Töö üheks eesmärgiks on uurida EstNLTK teegi grammatikate mooduli praktilisust ning tuvastada selle puudujääke.

Töö peamiseks eesmärgiks on luua andmebaas, mille toel oleks võimalik arendada tarkvara, mis võimaldaks juristidel tunduvalt mugavamalt ja efektiivsemalt otsida enda jaoks relevantseid kohtulahendeid. Samuti oleks selle abil võimalik koostada õiguspraktika analüüse, mis tähendab kohtute langetatud otsuste uurimist. Neid on kasulik arvesse võtta näiteks õigusaktide loomisel või poliitika kohandamisel.

Tulemusena valmis andmebaas, mis sisaldab igast kohtulahendist eraldatud huvipakkuvat informatsiooni. Seda kasutades on võimalik analüüsida kohtuotsuste tulemusi näiteks kohtute ja maakondade lõikes. Samuti saab moodustatud andmebaasi abil otsida kohtulahendeid, mis vastavad mingitele teatud kriteeriumitele.

Töö teises peatükis antakse ülevaade tekstikaevest ja informatsiooni eraldamisest ning nende erinevatest ülesannetest. Kolmandas peatükis kirjeldatakse töös kasutatavaid andmeid. Neljandas peatükis tuuakse välja töö käik, milles kirjeldatakse andmete eeltöötlust ning seejärel informatsiooni eraldamist, mille alla kuulub teksti segmenteerimine ning faktituvastus. Viimasena antakse ülevaade töö käigus tekkinud probleemidest. Viiendas peatükis kirjeldatakse segmenteerimise ja faktituvastuse tulemusi ning tuuakse näiteid analüüsist, mida on võimalik saadud andmebaasi abil teha. Kuuendas peatükis tuuakse välja edaspidise töö plaanid.

2. Tekstikaeve

Tänu interneti laialdasele levikule tekivad tänapäeval andmehulgad kiiremini kui neid analüüsitakse. Selle tagajärjel on väga oluliseks alaks muutunud **andmekaeve** (*data mining*), mis tegeleb andmebaasis olevatele struktureeritud andmetele algoritmide ja meetodite rakendamisega, millele järgneb tulemustest mittetriviaalsete ja peidetud mustrite otsimine ehk **teadmiste omandamine** (*knowledge discovery in databases, KDD*). Struktureeritud andmehulkadest veel kiiremini tekib struktureerimata andmeid, mistõttu on oluliseks erialaks saanud **tekstikaeve** (*text mining* ja *knowledge discovery in text, KDT*), mis tegeleb struktureerimata tekstile algoritmide ja meetodite rakendamisega, eesmärgiga tekstist peidetud mustreid leida. [2]

Tekstikaeves on tähtsal kohal **loomuliku keele töötlemine** (*natural language processing*), mille eesmärgiks on arendada meetodeid, mis võimaldaksid arvutitel loomulikust keelest aru saada. Lause semantikast aru saamist on raske teostada eriti sellepärast, et tihtipeale esineb lauses informatsiooni, mida otseselt kirjutatud ei ole ehk antud lause põhjal tuleb teha järeldusi. Järelduste tegemist omakorda raskendab otsustamine, kas selleks on piisavalt informatsiooni. [4]

2.1. Tekstikaeve ülesanded

Tekstikaevega lahendatakse mitmeid erinevaid probleeme. **Informatsiooni eraldamine** (*information extraction*) on nii eraldiseisev ülesanne kui ka paljude tekstikaeve projektide alguspunkt, kuna see võimaldab tekstist eraldada tähtsamaid tekstilõike. Selle tulemusena väheneb ka teksti maht, sest uurijate jaoks ebaolulised tekstiosad võib vaatluse alt välja jätta. [5]

Tekstikaeve meetoditega saab moodustada ka **tekstikokkuvõtteid** (*text summarization*) pikkadele tekstidele või tekstikogumikele lühikeste kokkuvõtete moodustamisega. Tekstikokkuvõtteid võidakse moodustada kahel viisil:

- kasutades informatsiooni eraldamise meetodeid
- sünteesides informatsiooni dokumentides olevate andmete põhjal, mille tulemusena moodustub abstraktne kokkuvõte, milles olev tekst ei pruugi otseselt dokumentides esineda

Samuti tegeletakse dokumentide organiseerimisega mitmel erineval viisil. Esiteks võib kasutada **klasterdamise** (*clustering*) meetodeid, mis jagab dokumendid gruppidesse ehk klastritesse vastavalt nende sisule. Teiseks on võimalik rakendada dokumentide **klassifitseerimise** (*classification*) meetodeid, mis jagab dokumendid gruppidesse, kuid erinevalt klasterdamisele on vaja defineerida, millistesse klassidesse on võimalik dokumente jagada. Kolmandaks dokumentide grupeerimise meetodiks on **teema modelleerimine** (*topic modeling*), mis on klasterdamisele väga sarnane, kuid dokumendi ühte klastrisse kuuluvaks lugemise asemel antakse igale dokumendile tõenäosuslikud väärtused, mis näitavad, kui suure tõenäosusega vastav dokument erinevatesse klastritesse kuulub. [5]

Informatsiooni otsimiseks (*information retrieval*) uuritakse kontekste, milles dokumentides leiduvad sõnad esinevad ning hinnatakse selle põhjal nende sarnasust, näiteks sünonüümid esinevad väga sarnases kontekstis. Selle abil saab otsida lisaks dokumentidele, mis vastavad antud sõnale või fraasile, ka dokumente, mis vastavad antud sõnaga sarnases kontekstis esinevatele sõnadele. Informatsiooni otsimise eesmärgiks ei ole anda küsimusele vastust, vaid esitada dokumendid, kust võib vastuseid leida. [2]

Tekstikavet võib rakendada ka **tekstivoogudel** (*mining text streams*), mis tähendab pideva tekstivoo töötlemist, mida esineb näiteks sotsiaalmeedias ja uudiste veebilehtedel. Tekstivoogude kaevet raskendab see, et andmeid on raske salvestada hilisemaks töötlemiseks, kuna teksti hulk kasvab pidevalt. Seega toimub ühekordne ja järjepidev tekstikaeve meetodite rakendamine, mis muudab töötlemise keerulisemaks. [5]

Sotsiaalmeedia väga laialdase leviku tõttu on mõeldav rakendada tekstikaevet (*text mining in social media*) ka selles valdkonnas. Tekstikaeve eesmärgiks sotsiaalmeedias on dünaamilise ja mürase kirjakeelega tekstides kaevamine, mida esineb peamiselt sotsiaalmeedia veebilehtedel nagu näiteks Facebook ja Twitter. [5]

Ärimaailmas on eriti oluliseks tekstikaeve valdkonnaks **arvamuskaeve** ehk meelsusanalüüsi (*opinion mining*) teostamine, mille eesmärgiks on internetist leida teksti, milles kirjutaja avaldab oma arvamust huvipakkuva teema kohta, näiteks mingi uue toote kohta. Pärast leitud arvamuste kohta kokkuvõtte tegemist on võimalik neile toetudes näiteks firma juhtkonnal paremaid otsuseid langetada. [5]

Biomeditsiini andmetes (*text mining in biomedical data*) rakendatakse tekstikaevet peamiselt kirjutatud artiklitest informatsiooni kogumisega, kuna avaldatud kirjandust on liiga palju, et inimesed seda ise suudaks kokku võtta. [5]

2.2. Informatsiooni eraldamine

Tekstikaeve üheks ülesandeks on informatsiooni eraldamine, mis on käesoleva töö põhifookuseks. Informatsiooni eraldamise eesmärgiks on struktureerimata tekstist tuletada struktureeritud andmeid. [6] Olgu antud loomulikus keeles järgmine lause:

> *“Esmaspäeval puhkes Riia tänaval suur tulekahju.”*

Sellest lausest on võimalik saada näiteks järgmised struktureeritud andmed:

> *Aeg: "esmaspäev"*
> *Koht: "Riia tänav"*
> *Sündmus: "tulekahju"*

2.2.1. Informatsiooni eraldamise ülesanded

Informatsiooni eraldamise ülesanded jagunevad järgmistesse kategooriatesse:

- **Nimega üksuste tuvastamine** (*named entity recognition)*, mis tähendab nimede tuvastamist ja klassifitseerimist [6]. Nimedeks loetakse näiteks järgmisi sõnu ja fraase:
  - inimeste nimed, näiteks *"Katrin"*
  - kohanimed, näiteks *"Eesti Vabariik"*
  - ettevõtete ja asutuste nimed, näiteks *"Eesti Energia"* ja *"Tartu Ülikool"*
  - aega ja raha väljendavad fraasid, näiteks *"10 tundi"* ja *"30 eurot"*
- **Elementide konstrueerimine** (*template element construction)* tähendab tekstist nimeüksuste kohta lisainformatsiooni otsimist, näiteks kui on tegemist inimesega, siis neile vastav lisainformatsioon võib olla tema rahvus, sugu, töökoht ja muud. Seda kategooriat loetakse vahel ka nimeüksuste tuvastamise alamülesandeks. [7]
- **Anafooride lahendamine** (*coreference resolution*) on ühele ja samale entiteedile viitavate sõnade tuvastamine, milleks võivad olla näiteks järgmised sõnad ja fraasid [6]:

    - ettevõtte või asutuse täisnimi ja lühend, näiteks *"Tartu Ülikool"* ja *"TÜ"*
    - inimese nimi ja asesõna *"tema"*, näiteks lausetes
      “Katrin sööb õuna. Talle maitsevad õunad”,
      või firma nimi ja sõna *"firma"*, näiteks
      “AS Selver tõstis puuviljade hindu. Firma kasum langes kolmekordselt.”
    - Anafooride lahendamise alla käivad ka juhtumid, kus lause alus puudub ning kirjeldatud on ainult tegevust, kuid tegijat on vaid eelnevates lausetes mainitud, näiteks dialoogis, kus küsitakse
      “Mida Miisu teeb?”
      ja vastatakse
      “Magab voodi peal.”
- **Seoste eraldamise** (*relation extraction)* eesmärgiks on identifitseerida, mis suhtes leitud nimeüksused on. Vastavaks suhteks võib olla töösuhe, näiteks kes töötab kelle jaoks, või asukoht, näiteks mis asub kus kohas. [6]
- **Sündmuste eraldamine** (*event extraction)* kasutab kõikide eelnevate informatsiooni eraldamise meetodite tulemusi, et tuvastada tekstist sündmuseid, proovides vastata küsimustele kes, kellele, millele, millal, kus, mida, miks, millega ja kuidas. [6]

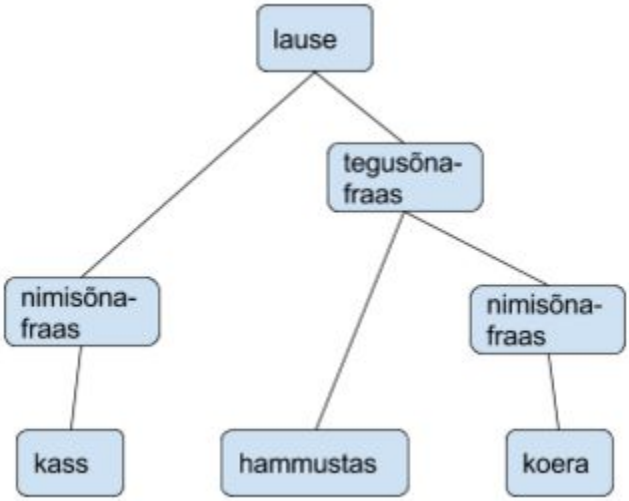
2.2.2. Informatsiooni eraldamise süsteemide ülesehitus

Esimesed informatsiooni eraldamise süsteemid olid väga erinevad. Leidus süsteeme, millega analüüsiti tekstid põhjalikult läbi, kasutades loomuliku keele töötlemise meetodeid, mille alla kuulus teksti süntaktiline ja semantiline analüüs. Samuti leidus süsteeme, mis kasutasid väga vähesel määral tekstianalüüsi ning informatsiooni eraldamine toetus pigem tekstist võtmesõnade leidmisele [8]. Informatsiooni eraldamise vastu hakati huvi tundma 1980ndatel ning esimeseks süsteemiks oli ATRANS, mille eesmärgiks oli eraldada informatsiooni pankadevaheliste ülekannete sõnumitest. Sellega umbes samal ajal arendati välja ka JASPER, mille eesmärgiks oli ettevõtte aruannetest eraldada informatsiooni tulude kohta.

Vaatamata sellele, et informatsiooni eraldamise süsteemides esineb ka tänapäeval suuri variatsioone, on üldtunnustatud töövoog järgmine [6]:

- **Metaandmete analüüs**, mis hõlmab dokumendi pealkirja, sisu, ülesehituse ja kuupäeva eraldamist.
- **Tokeniseerimine** ehk teksti jagamine sõnedeks, kus sõne on kas sõna, kirjavahemärk või mingi muu iseseisev ühik.
- **Morfoloogiline analüüs**, mis tähendab sõnade algvormi ehk lemma, struktuuri ja morfoloogilise informatsiooni tuvastamist. Morfoloogiliseks informatsiooniks eesti keeles on näiteks sõnaliik, kääne, arv, komparatsioon ehk alg-, kesk- või ülivõrre, isik, aeg, tegumood jms. Lisaks võib läbi viia ka morfoloogilise ühestamise, mis tähendab kõigist võimalikest morfoloogilistest märgenditest õige valimist. [9]
- **Lausepiiride tuvastamine**.
- Laialtlevinud **nimeüksuste tuvastamine**, mille alla kuuluvad valdkonnast sõltumatud fraasid, mis väljendavad näiteks aega, raha või riigi nime.
- **Fraaside tuvastamine**, mis tähendab järgmiste üksuste leidmist:
    - Nimisõnafraasid, mis koosnevad nimisõnadest, näiteks *"tüdruku lilled"*
    - Tegusõnafraasid, mis väljendavat tegevust või mingit seisundit, näiteks *"on ilus"*
    - Eessõnafraasid, mida eesti keeles ei esine

    - Lühendid
- **Süntaksianalüüs**, mille eesmärgiks on koostada leitud fraaside struktuuripuu, mis näitab, kuidas fraasid omavahel seotud on. Eesti keele jaoks on süntaksianalüsaatori loonud Kaili Müürisep. [9] Näiteks kui lause on
  "Kass hammustas koera.",
  siis selle struktuuripuu on järgmine: [joonis 1]



Joonis 1. Lause struktuuripuu.

Pärast eelnevalt loetletud sammude teostamist jätkatakse valdkonnast sõltuvate tegevustega. Sellega võivad abiks olla vastava valdkonna jaoks moodustatud sõnastikud, näiteks sõnastikud, mis sisaldavad majandustermineid või meditsiinitermineid.

Valdkonnast sõltuvate tegevuste alla käivad järgmised sammud [6]:

- **Nimeüksuste tuvastamine**, kus nimeüksusteks on nüüd kõik mida eelnevalt ei otsitud
- Defineeritud **mustrite abil tekstist informatsiooni eraldamine**, kus mustriks võivad kõige lihtsamal juhul olla näiteks regulaaravaldised. Mustrite abil võidakse eraldada kas tekstilõike, mis sisaldavad soovitud informatsiooni, või täpsemaid fragmente, millest hiljem moodustada struktureeritud andmeid
- **Anafooride lahendamine**
- **Informatsiooni kokkuviimine**, kus kasutatakse mustrite abil leitud andmeid ning pannakse samale entiteedile viitavate üksuste andmed kokku.

Tihtipeale tehakse ka valdkonnast sõltuvat ja sõltumatut informatsiooni eraldamist samaaegselt. Edaspidi jääb järgi vaid saadud struktureeritud andmeid analüüsida.

2.2.3. Informatsiooni eraldamise rakendused

Informatsiooni eraldamist rakendatakse mitmes valdkonnas. **Finantsanalüüsi** jaoks otsitakse internetist teksti, mis viitab huvi pakkuva firma teenustega rahulolule või rahulolematusele, näiteks blogid, kommentaarid, uudised. Uurides näiteks, kui palju negatiivseid ja positiivseid arvustusi firma kohta on ja kuidas firma reputatsioon viimase aja jooksul muutunud on, on võimalik ennustada, kuidas ettevõtte sissetulek lähitulevikus muutuma hakkab. Informatsiooni eraldamisega on võimalik ka jälgida vastava firma teenuse või toodete kohta internetis kirjeldatud kaebuseid, et neile võimalikult kiiresti lahendus leida. [7]

Turustusstrateegiate ehk kampaaniate ja reklaamide mõju on raske hinnata puhtalt kasumeid vaadeldes, seega on võimalik kasutada informatsiooni eraldamist, et näiteks koostada statistikat selle kohta, kui palju on vastavat toodet või teenust meedias mainitud, mille põhjal saab hinnata kampaania mõju. [7]

Bioinformaatikas arendatakse informatsiooni eraldamise süsteeme, mis suudaksid teadusartiklitest näiteks eraldada geenide ja proteiinide nimesid, et leida neile vastavad standardiseeritud identifitseerijad, ja samuti leida tekstist vastavate geenide või proteiinide funktsionaalsuste kirjeldused [10]. Lisaks otsitakse tekstidest ka andmeid proteiinidevaheliste suhtlusvõrgustike kohta. [11]

Sotsiaalmeedias on informatsiooni eraldamine eriti keerukas ettevõtmine, kuna ühe sündmuse kohta käiv info võib olla jagunenud mitme sissekande vahel ning neis olev kirjakeel sisaldab palju müra. Selles valdkonnas on informatsiooni eraldamisel siiski suur potentsiaal, kuna tihtipeale on sotsiaalmeedias kõige värskem informatsioon vastupidiselt uudistele ja muule meediale, kus informatsiooni avaldamine on ajakulukam. [6]

2.2.4. Informatsiooni eraldamine käesolevas töös

Informatsiooni eraldamise grammatikad jagunevad mustripõhisteks ja statistilisteks. Mustripõhised grammatikad toetuvad enamasti regulaaravaldistele. Kõigepealt defineeritakse sõnastikud regulaaravaldistena iga otsitava informatsiooni elemendi jaoks. Seejärel kombineeritakse neid vastavalt vajadusele, et moodustada reeglid, mille abil tuvastada huvipakkuvaid tekstiosi [12]. Mustripõhist lähenemist on kasutanud Timo Petmanson oma magistritöös mustripõhisest faktituletusest eestikeelsetest tekstidest. [13]

Statistiliste grammatikate puhul kasutatakse informatsiooni tuvastavate reeglite moodustamiseks peamiselt juhendatud masinõppe meetodeid, mis tähendab et reeglite moodustamine toimub automaatselt, kui programmile ette anda vastavalt märgendatud treeningkorpus. Reeglite moodustamiseks kasutatakse mudeli treenimisel erinevat informatsiooni, näiteks sõnadele eelnevaid ja järgnevaid sõnu, sõnade prefikseid ja sufikseid ja muud. Statistilist lähenemist on kasutanud Aleksandr Tkatšenko oma magistritöös nimeüksuste tuvastamiseks. [14]

Käesolevas töös on kasutatud mustripõhist lähenemist.

Grammatikate koostamine EstNLTK's

EstNLTK võimaldab paindlikumalt ja selgema koodiga leida tekstimustreid, kui üksnes Python'i regulaaravaldiste kasutamine. Selleks kasutatakse EstNLTK objekte *Regex*, *IRegex*, *Lemmas*, *Union*, *Concatenation* ja *Gaps*.

Näiteks olgu tekstis, millest informatsiooni otsitakse, järgmine lõik:

> *"Gigabyte Z170X-Gaming G1 emaplaadid, 500 eurot.*
> *Protsessoriks Intel Core i5-6600K, 230 eurot.*
> *120 Eurot AMD Radeon R7 360 graafikakaardi eest."*

ning sellest on vaja otsida arvutikomponentide hindu.

Esimeseks sammuks on defineerida järgmised *Regex*, *IRegex* ja *Lemmas* objektid:

```
arvutikomponent = Lemmas('emaplaat', 'protsessor', 'graafikakaart',
                  name='arvutikomponent')
number = Regex('\d+')
ühik = IRegex('euro')
tühik = Regex('\s')
```

Lemmas objektiga *arvutikomponent* defineeritakse kõigi sõnade algvormid, mis vastavad huvipakkuvatele arvutikomponentidele. Selle objekti abil tuvastatakse tekstist valitud sõnad olenemata sellest, mis vormis nad esinevad. Käesolevas näites on iga otsitav arvutikomponent erinevas vormis: *"emaplaadid", "Protsessoriks"* ja *"graafikakaardi"*. Lisaks on võimalik igale defineeritud objektile anda nimi, mille abil saab hiljem võimalik näha, mis tekstiosi vastav objekt tuvastas.

Regex objektiga *number* tuvastatakse tekstis esinevaid mistahes numbreid. *IRegex* objekti *ühik* abil leitakse sõna *"euro"* esinemisi, kusjuures erinevalt *Regex* objektist ei ole *IRegex* objekti puhul oluline, kas antud sõna on tekstis suur- või väiketähtedega. See on oluline, kuna antud tekstis esineb sõna *"euro"* nii suure kui ka väikse algustähega. Samuti defineeritakse *Regex* objekt mis vastab tühikule.

Järgmisena moodustatakse tervet komponendi hinda tuvastav objekt:

```
hind = Concatenation(number, tühik, ühik, name='hind')
```

Concatenation objekt *hind* abil tuvastatakse tekstiosi, milles objektile antud argumendid esinevad vahetult üksteise järel. Kuna hinna numbrile järgneb antud tekstis alati tühik ning seejärel rahaühik, siis sobis see objekt vastava mustri tuvastamiseks.

Lõpuks on vaja defineerida objekt, mis leiab nii arvutikomponendi kui ka sellele vastava hinna:

```
komponent_ja_hind = Union(
   Gaps(hind, arvutikomponent),
   Gaps(arvutikomponent, hind),
)
```

Gaps objekt on sarnane *Concatenation* objektiga, kuna selle abil tuvastatakse tekstiosi, milles objektile antud argumendid esinevad vastavas järjekorras. Erinevalt *Concatenation* objektist võib *Gaps* objekti argumentide vahel olla mistahes arv muid sümboleid. Antud tekstis esineb arvutikomponenti määrav sõna nii lause alguses kui ka lõpupoole, seega ei ole võimalik kasutada *Concatenation* objekti, kuna ei ole teada, mis sümbolid komponendi ja hinna vahel võivad esineda. Gaps objekte moodustati kaks, kuna hind esineb nii komponendi ees kui ka järel.

Union objekt *komponent_ja_hind* leiab tekstist kõik tekstiosad, mis vastavad vähemalt ühele objekti argumendiks antud *Gaps* objektile.

Grammatika rakendamist tekstile on näha järgmises koodilõigus:

```
tekst = Text('Gigabyte Z170X-Gaming G1 emaplaadid, 500 eurot.'
           'Protsessoriga Intel Core i5-6600K, 230 eurot.'
           'AMD Radeon R7 360 graafikakaart, 120 eurot.')

for lause in tekst.split_by('sentences'):
     komponent_ja_hind.get_matches(lause)
```

Kõigepealt tuleb antud tekstist teha *Text* objekt ning jagada see funktsiooniga *split_by* kas lauseteks või paragrahvideks. Seejärel on vaja vaid koostatud grammatikale *komponent_ja_hind* rakendada funktsiooni *get_matches* andes sellele argumendiks iga lause eraldi.

Tulemusena tuvastab grammatika esimesest lausest *“500 euro”* ja *“emaplaadid”*, teisest lausest *“230 euro”* ja *“Protsessoriks”* ning viimasest lausest *“120 Euro”* ja *“graafikakaardi”*.

3. Andmete kirjeldus

Töö jaoks vajalikud andmed olid kohtulahendid, mida on võimalik alla laadida Riigi Teataja veebilehelt [15]. Lahendid pärinesid maa-, haldus- ja ringkonnakohtutest, mis tähendab, et andmete hulgas riigikohtu lahendeid ei esinenud. Dokumendid on kättesaadavad vaid pdf-formaadis. Töö alguses sai alla laetud 379 844 kohtulahendit, kuid pärast töötlemist ja filtreerimist jäi alles 177 453 dokumenti.

Kohtulahendid jagunevad kahte gruppi: kohtuotsused, milles on välja toodud kohtumenetluse lõppotsused, ning kohtumäärused, millega kinnitatakse näiteks pooltevahelist kokkulepet, menetluse lõpetamist või lahendatakse muid menetluse käigus tekkinud küsimusi.

Käesolevas töös keskendutakse vaid kohtuotsustele, kuna kõige huvipakkuvamaks osaks on menetluse lõppotsus, mida leidub vaid kohtuotsustes.

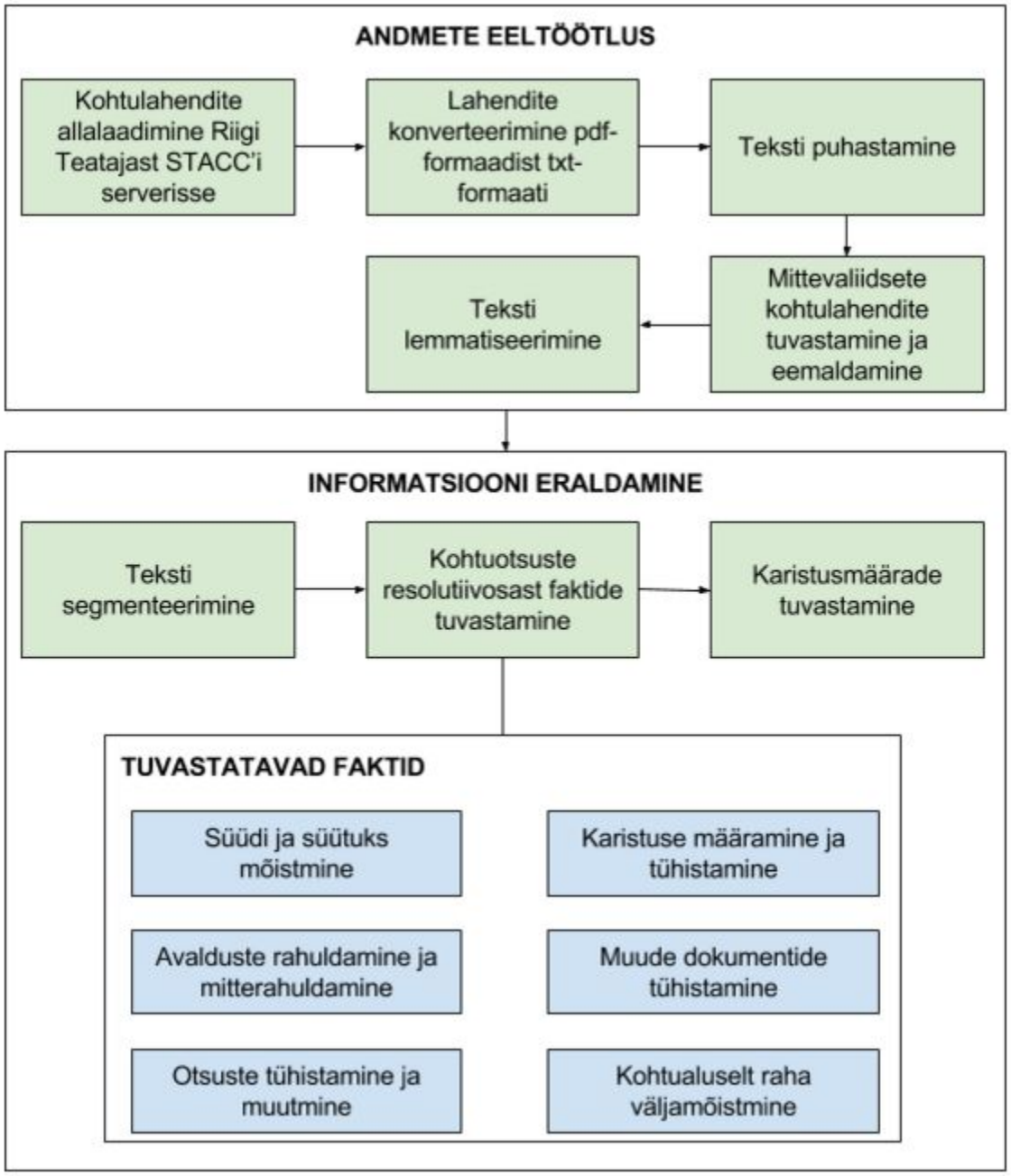
Kohtuotsustes leidub mitmesugust informatsiooni, näiteks:

- kas tegemist on väärteoasja, tsiviilasja, kriminaalasja või haldusasjaga
- kohus ja kohtunik
- hageja ja kostja esindajad
- dokumendi koostamise kuupäev
- kohtuasja number
- kohtusse pöördumise põhjus ja asjaolud
- kohtu ja asjaosaliste seisukohad
- lõppotsus ehk resolutsioon
- kelle kanda jäävad menetluskulud
- edasikaebamise kord

Kohtulahendid on tihti anonümiseeritud, mis tähendab, et dokumendist on eemaldatud isiku identifitseerimist võimaldavad andmed, näiteks nimi, isikukood, sünnikuupäev, töökoht jms.
Lisaks võeti Riigi Teataja veebilehelt eraldi lisainformatsiooni iga lahendi kohta, milleks oli kohtuasja number, menetlusliik, kohus, lahendi liik, lahendi alaliik, kohtunik, kohtuasja algus, menetluse liik, menetluse algus, asja kategooria, lahendi aeg ja lahendi jõustumised.

4. Töövoog

Andmete hoidmiseks kasutati PostgreSQL [16] ja Elasticsearch [17] andmebaase, skriptide kirjutamiseks Python'i versiooni 3.4, loomuliku keele töötluseks Python'i teeki EstNLTK ning andmetes päringute koostamiseks ja tulemuste vaatamiseks Elasticsearch andmebaasil põhinevat tööriista TEXTA. Käesoleva töö sammud on näha järgmisel skeemil: [joonis 2]



Joonis 2. Töövoo skeem

4.1. Andmestiku eeltöötlus

Esimese sammuna said pdf-formaadis olevad kohtulahendid paigutatud serverisse. Kuna andmetega töötamiseks pdf-formaat ei sobi, oli vajadus konverteerida kohtulahendid txt-formaati. Formaadi muutmise käigus oli soov säilitada dokumendi ülesehitust, näiteks lõigud ja loendid, ning selleks sobis tarkvara teegi Poppler [18] käsurea funktsioon *"pdftotext"*.

Käsu struktuur, mida rakendati Unix operatsioonisüsteemi käsureal igale pdf-formaadis kohtulahendile oli järgmine:

```
pdftotext -layout -nopgbrk pdf_fail txt_fail,
```

kus pdftotext on funktsiooni nimi, millele järgnevad argumendid:

- *-layout*, mis määrab, et konverteerimisel säilitatakse pdf-faili ülesehitus, kuna see on vajalik edaspidisel töötlemisel
- *-nopgbrk* määrab, et ei lisataks lehekülje lõpetamise sümboleid, kuna käesoleva töö puhul ei ole vaja lehekülgi eristada
- *pdf_fail* - pdf-formaadis oleva kohtulahendi failinimi
- *txt_fail* - failinimi, kuhu konverteerimise txt-formaadis tulemus salvestatakse

Konverteerimine ebaõnnestus täielikult 817 kohtulahendil, mis jäid edaspidisest töötlemisest ja analüüsist välja. Pärast kohtulahendite konverteerimist txt-formaati, paigutati andmed PostgreSQL andmebaasi, millega loodi ühendus Python'i teegi psycopg2 [19] abil. Tulemusena tekkis tabel ***kohtulahendid_raw***, mille tunnused olid järgmised:

- *raw_text*, mis sisaldab kogu teksti, mis saadi pdf-formaadi konverteerimisel txt-formaati
- *case_id*, mis on Riigi Teatajast alla laadimisel failile nimeks pandud id number.

Samuti paigutati andmebaasi Riigi Teatajast võetud lisainformatsioon ning loodi tabel nimega **kohtulahendid_detailid**, mille tunnused olid järgmised:

- *kohtuasja_number* - kohtumenetluse number
- *menetlusliik* - tsiviil-, haldus-, kriminaal- või väärteoasi
- *kohus*
- *lahendi_liik* - kohtuotsus või kohtumäärus
- *lahendi_alaliik* - täpsustab lahendi otsuse liiki, nt tagaseljaotsus või otsuse resolutiivosa
- *kohtunik*
- *kohtuasja_algus* - kuupäev, millal kohtuasi algas
- *menetluse_liik*
- *menetluse_algus* - kuupäev, millal menetlus algas
- *asja_kategooria* - mis kategooria alla kohtumenetlus kuulub, nt liiklusväärteod, võlaõigus või perekonnaõigus
- *lahendi_aeg* - kohtulahendi vormistamise kuupäev
- *lahendi_joustumised* - kuupäev, mil kohtulahendis tehtud otsus jõustub
- *rt_id* - Riigi Teatajas kohtulahendile antud id number
- *id* - andmebaasis antud automaatne id

Kuigi eelnevas nimekirjas leidub tunnuseid, mis olid ka kohtulahendite teksti sees, oli seda tunduvalt lihtsam kätte saada veebilehelt.

Seejärel olid kohtulahendite andmed kahes eraldi tabelis, kuid edasise töötluse jaoks oli vaja, et kõik tunnused oleks ühes tabelis. Seetõttu oli järgmiseks sammuks tabelite *kohtulahendid_raw* ja

kohtulahendid_details ühendamine ühtsesse tabelisse ***kohtulahendid_full***. Kui andmed olid edukalt PostgreSQL andmebaasi paigutatud ning ühte tabelisse ühendatud, algas andmete olemasolevatest tunnustest uute tunnuste moodustamine.

4.1.1. Teksti puhastamine

Kuna pdf-failide konverteerimise tulemusena saadud txt-failidel olid mõned puudujäägid, oli vajadus neid korrigeerida. Kuna konverteerimisel valiti, et säilitataks dokumendi ülesehitus, siis lisati näiteks txt-formaadis olevasse faili lausete keskele reavahetused, kui pdf-formaadis olevas failis jätkus lause järgmiselt realt. Näiteks:

> *"Harju Maavalitsuse ÕVVTK*
> *komisjon jättis oma 30.04.99 otsusega H.K.u kaebuse Tallinna Linnavalitsus korralduse nr 1333-k seadusevastaseks tunnistamise nõudes rahuldamata."*

Selle parandamata jätmine oleks edaspidises töös takistanud teksti lausestamist. Teksti ülesehituse parandamiseks sobisid Python'i regulaaravaldised, mis on kättesaadavad teegina re. Iga kohtulahendi väljale *raw_text* rakendati regulaaravaldisi, mis tegid tekstis järgmisi muudatusi:

- Tühikute eemaldamine rea algusest ja lõpust
- Mitmekordsete tühikute asendamine ühekordse tühikuga
- Numbri eemaldamine, kui see oli ainuke sümbol real, mis tähendab, et tegemist on leheküljenumbriga
- Rohkem kui kahe reavahetuse asendamine kahe reavahetusega, kuna kahest reavahetusest piisab, et eraldada tekstis lõike
- Reavahetuse eemaldamine, kui sellele eelneb üksik sümbol, mille mõlemal pool on punkt, ja järgneb väike täht. See tähendab, et rida lõppeb lühendiga ning jätkub järgmisel real.
- Reavahetuse eemaldamine, kui sellele eelnevad järjest tühik, suur täht ning punkt. See tähendab, et rida lõppeb lühendatud nimega
- Reavahetuse eemaldamine, kui sellele ei eelne lauselõpumärk,mis tähendab, et lause jätkub järgmisel real. Reavahetus eemaldati vaid siis, kui sellele ei eelnenud teine reavahetuse sümbol, et vältida lõikude kaotamist.

Nende muudatuste tulemusena saadud tekstist moodustati uus tunnus *clean_text*.

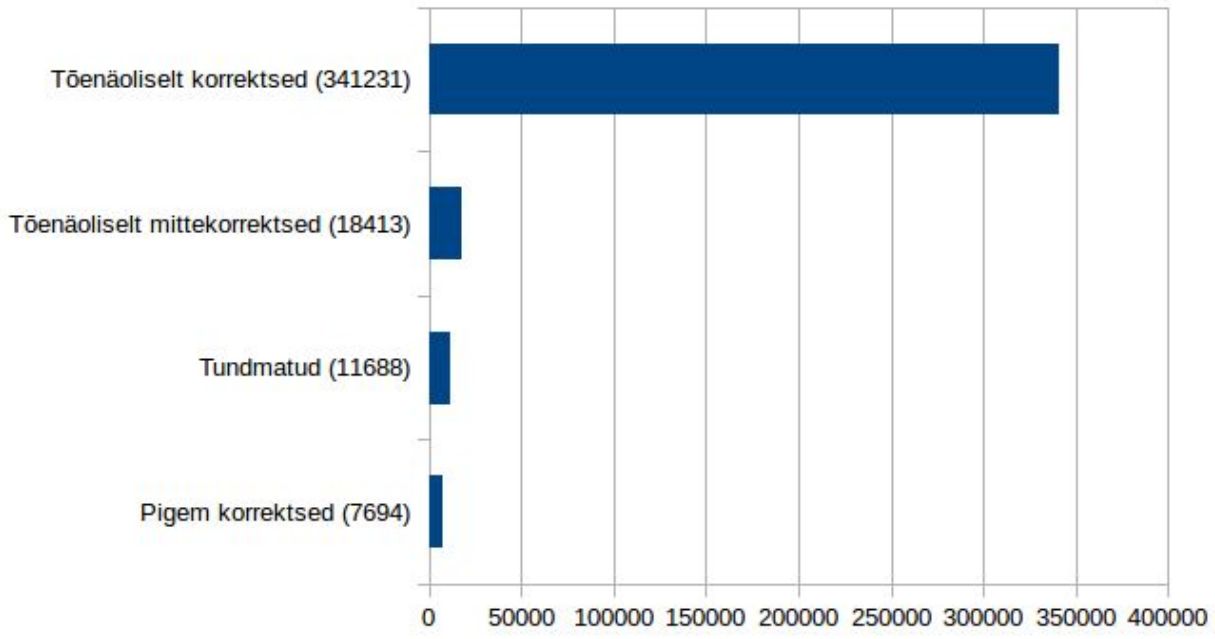
4.1.2. Sobimatute kohtulahendite tuvastamine

Teksti puhastamise käigus selgus, et andmebaasis leidub kirjeid, milles kohtulahendi tekst on teadmata põhjustel konverteerimisel muutunud loetamatuks pealtnäha suvaliste sümbolite järjendiks. Kuna selliste lahenditega ei ole võimalik tööd teha, oli vajadus need tuvastada ja eemaldada. Loetamatute dokumentide tuvastamist raskendavateks teguriteks olid kohtulahendite rohkus ning lahendite sisu ja vormistuse erinevused, mille tõttu ei olnud seda võimalik teha manuaalselt. Sobimatute lahendite tuvastamise viis läbi töörühmakaaslane, kes oli juba oma projektis sarnast tööd teinud, kasutades selleks hierarhilist klasterdamist.

Lahendid klasterdati vastavalt sellele, kui tõenäoline oli, et nende tekst on loetav. Esimesse gruppi jagati kohtulahendid, mis olid suure tõenäosusega korrektsed ning teise kohtulahendid, mis olid pigem korrektsed kui mittekorrektsed. Kolmas grupp koosnes lahenditest, mis olid suure tõenäosusega mittekorrektsed ning neljandasse jäid lahendid, mida ei suudetud teistesse gruppidesse jagada. Kohtulahendite loetavuse hindamiseks kontrolliti näiteks sõna *"kohtu"*

esinemist tekstis ja maksimaalset järjestikust sümbolite jada, mis ei koosne numbritest, tähtedest ega tühikutest.

Andmebaasi lisati uus tunnus nimega symbols_clustering, mille väärtuseks oli grupp, millesse vastav lahend jagati. Lahendite jagunemine loetletud klastrisse on välja toodud järgmisel joonisel: [joonis 3]



Joonis 3. Lahendite jagunemine klastritesse

4.1.3. Teksti lemmatiseerimine

Kohtulahendites päringute tegemise lihtsustamiseks viidi tekst lemmade kujule ehk iga tekstis olev sõna viidi tema algvormi. Näiteks oleks lause

“Kohus ei anna luba edasikaebamiseks”

lemmatiseeritud kuju

“Kohus ei andma luba edasikaebamine”.

Tänu sellele on võimalik teha lahendites sõnavormist sõltumatuid otsinguid, mis tähendab, et sõna või fraasi otsimisel piisab selle algvormi päringust ning otsingusse ei pea lisama kõiki võimalikke vastava sõna või fraasi vorme. Näiteks, kui on tarvis tuvastada tekstides sõna *"abielu"* mistahes käändes esinemist, piisab lemmatiseeritud tekstist sõna *"abielu"* otsimisest ning ei ole vaja koostada päringut, mis otsiks näiteks sõnu *"abielu"*, *"abielus"*, *"abieluga"*, *"abielust"* ja nii edasi.

Kohtulahendite lemmatiseerimiseks kasutati Python’i teegi EstNLTK funktsiooni *"lemmas"*. Selguse jaoks jagati tekst enne lemmatiseerimist lauseteks, kuna oli soov vormindada tulemust nii, et igal real oleks üks lause, mille sõned on eraldatud tühikutega. Võimalik on, et ühel sõnal on mitu tähendust ja seega ka mitu analüüsitulemust. Selle probleemi lahendamiseks valiti analüüsidest esimene. Tulemusena tekkis uus tunnus *"lemmas"*.

4.2. Informatsiooni eraldamine

Pärast lemmatiseerimist tõsteti andmed PostgreSQL andmebaasist Elasticsearch andmebaasi, et võimaldada TEXTA tööriista kasutamist lahenditega töötamisel. Välja jäeti kohtulahendid, mis olid eelnevalt märgitud suure tõenäosusega loetamatuteks. Edaspidisest andmete töötlemisest võeti välja kohtumäärused ja jäeti alles vaid kohtuotsused, kuna käesoleva töö jaoks huvipakkuvat infot, milleks oli menetluste lõppotsused, leidus vaid kohtuotsustes. Kohtuotsuseid oli kokku 177 453.

4.2.1. TEXTA tööriist

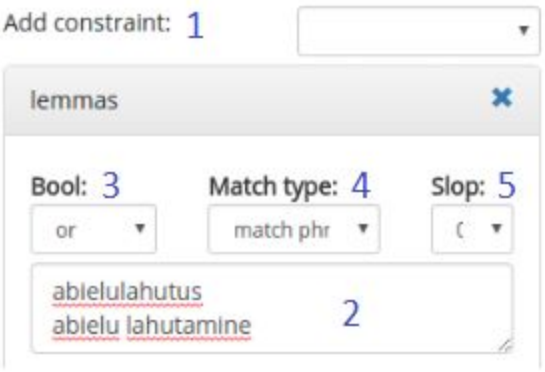
Terminology EXtraction and Text Analytics ehk TEXTA tööriist [20] on Tarkvara Tehnoloogia Arenduskeskuses loodav tarkvara, mis ei ole veel avalikult kättesaadav. TEXTA eesmärgiks on võimaldada vabatekstiliste andmete analüüsi, võttes arvesse vastava keele omadusi. Käesolevas töös on TEXTA't kasutatud andmetes kiirelt päringute tegemiseks, nende tulemuste vaatamiseks ning valitud tunnuse põhjal päringutulemuste agregeerimiseks. Samuti kasutati võimalust luua fikseeritud korpusel leksikone. TEXTA kasutamine on mugavaks tehtud graafilise kasutajaliidesega.

Päringute koostamine

Päringu koostamine toimub TEXTA's järgmiselt:

1. Tunnuse valimine, mille järgi tulemusi filtreerida
2. Kui tunnusel on lõpmatu arv võimalikke vääruseid, kirjutada soovitud väärtus(ed) selleks ettenähtud tekstivälja. Kui tunnusel on piiratud arv võimalikke väärtuseid, on neid võimalik tekstivälja sisestada ka rippmenüüst valides.
3. Kui tekstivälja sisestati väärtused $x_1, x_2, \ldots, x_n$, siis järgmisena valida, kas nendest väärtustest peavad tulemusena antud kirjetes esinema kõik, mis on loogiliste operaatorite kaudu väljendatav kui $x_1 \,\&\, x_2 \,\&\, \ldots \,\&\, x_n$, vähemalt üks neist ehk $x_1 \vee x_2 \vee \ldots \vee x_n$ või mitte ükski neist ehk $\neg x_1 \,\&\, \neg x_2 \,\&\, \ldots \,\&\, \neg x_n$.
4. Valida, kas sisestatud väärtused võivad esineda eraldi, fraasina või fraasi prefiksina. Näiteks, kui otsitavaks väärtuseks on sisestatud *"määrates karistus"*, siis lubades sõnade tekstis eraldi esinemine, antakse tulemusena kirjed, mis sisaldavad sõnu *"määrates"* ja *"karistus"* mistahes järjekorras ning üksteisest mistahes kaugusel. Kui lubada sõnade esinemist vaid fraasina, antakse tulemusena kirjed, milles fraas *"määrates karistus"* esineb täpselt etteantud kujul. Valides antud sõnade esinemise fraasi prefiksina, võib fraasi viimasele sõnale järgneda veel muid tähemärke, näiteks *"määrates karistusena"* ja *"määrates karistuseks"*.
5. Kui otsitakse fraasi, valida, mitu sõna võib sisestatud fraasis olevate sõnade vahel olla (0 kuni 5). Näiteks, kui fraas on *"määrates karistus"* ja väärtuseks valitakse 1, saadakse tulemusena kirjed, kus esinevad näiteks fraas *"määrates karistus"* või *"määrates talle karistus"*.

Järgnevalt on näha päringu koostamise vormi, kus otsitavateks väärtusteks lemmatiseeritud tekstist on *"abielu lahutamine"* ja *"abielulahutus"*, millest tulemusena saadud kirjetes peab esinema vaid üks. Fraaside esinemist otsitakse täpselt etteantud kujul ning nende vahel muidu sõnu ei lubata. Eelmise loetelu punktidele on viidatud joonisel vastava numbriga. [joonis 4]



Joonis 4. Päringu koostamine.

Päringu tulemused antakse tabelina, milles on võimalik andmete tunnuseid peita, et võimaldada paremat ülevaadet. Samuti tõstetakse tuvastatud otsingufraas(id) värvitud taustaga esile. Joonisel 5 on näha kuidas esitatakse joonisel 4 moodustatud päringu tulemused. Kõik tunnused peale *"lemmas"* on peidetud kujul ning otsingule vastavaid kirjeid on kokku 3289.

Joonis 5. Päringu tulemused.

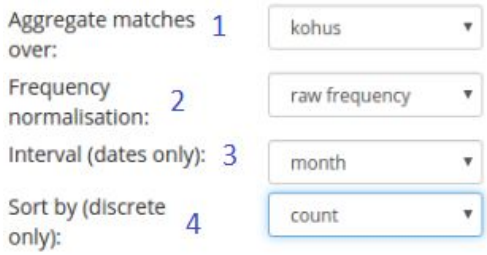
Tulemuste agregeerimine

Pärast päringu koostamist on võimalik selle tulemusena saadud kirjeid agregeerida ehk kokku lugeda mingi tunnuse põhjal järgmiselt:

1. Valida tunnus, mille põhjal agregeerida
2. Valida, kas agregeerimise tulemus on päringu tulemuse kirjete toorarv või suhtarv

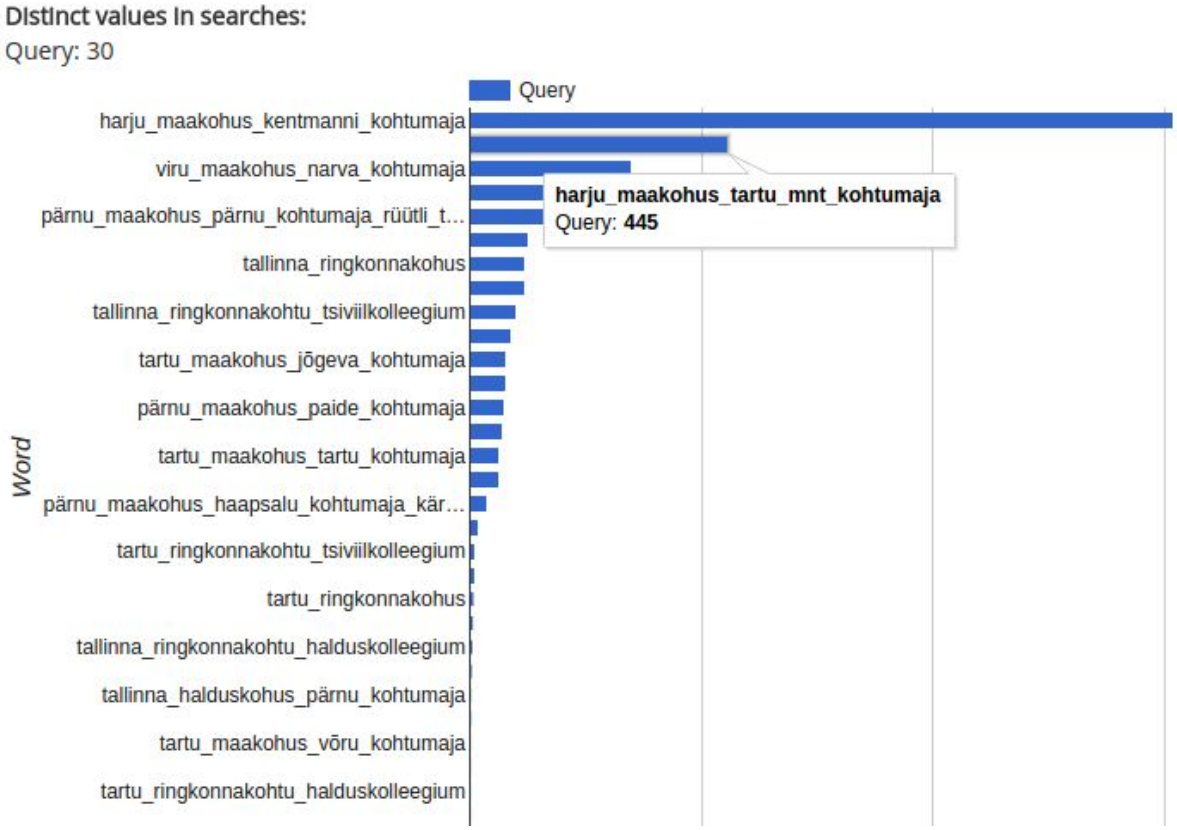
3. Ajatelje agregatsioonide puhul valida millise intervalli sees olevaid kuupäevi vaadeldakse ühe grupina, kus intervalliks võib olla päev, kuu, veerand või aasta
4. Diskreetsete agregatsioonide puhul valida, mis järjekorras agregeerimise tulemusi näidatakse, kas kirjete (suht)arvu või tähtsuse (*significance*) järjekorras

Järgmisel joonisel on näha agregatsiooni moodustamist, kus eelmise loetelu punktidele on viidatud joonisel vastava numbriga. [joonis 6]

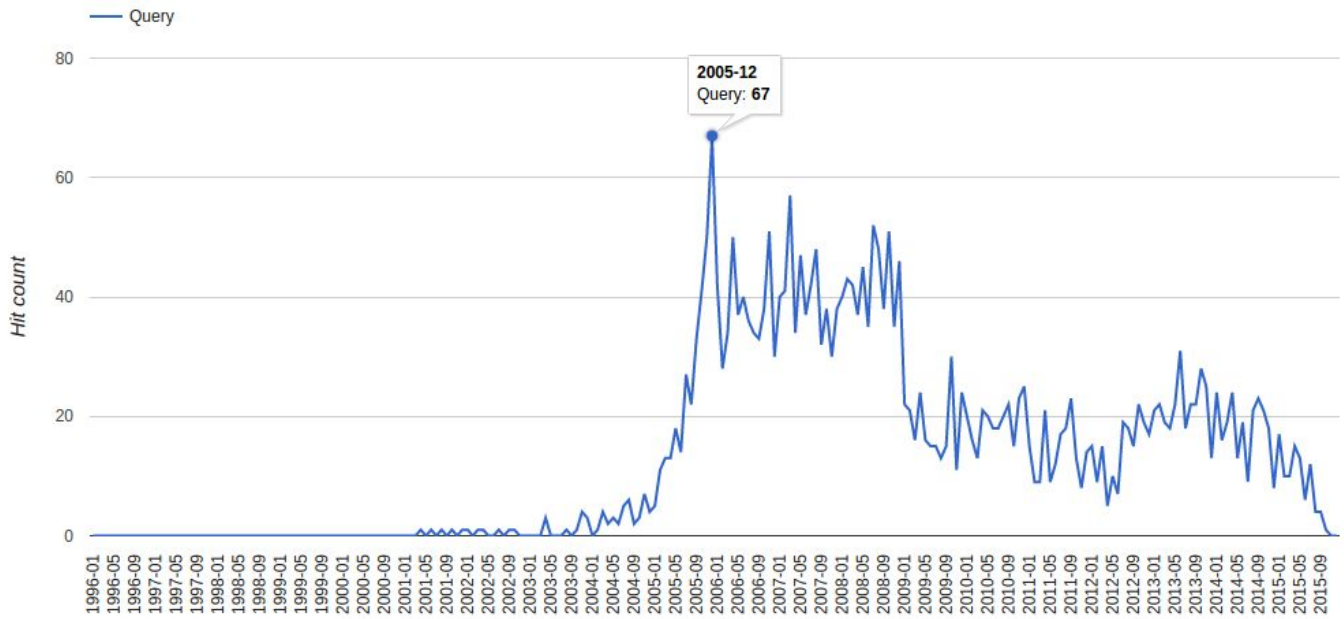


Joonis 6. Agregatsiooni koostamine.

Joonisel 7 on näha diskreetse agregatsiooni tulemust, kus tunnuseks, mille põhjal eelnevalt koostatud päringu tulemusi agregeeriti, oli *kohus*. Joonisel 8 on näha sama tulemuse ajatelje agregatsiooni tulemust, kus ajaliseks tunnuseks, mille põhjal agregeeriti oli *menetluse_algus*. Täpsete kirjete arvu saamiseks tuleb hiir liigutada huvipakkuva tulba või ajapunkti kohale.



Joonis 7. Diskreetne agregatsioon.



Joonis 8. Ajatelje agregatsioon.

TEXTA's päringute tegemine ning tulemuste agregeerimine lihtsustasid tunduvalt töö käigus saadud vahetulemuste kontrollimist ja jälgimist. Näiteks oli selle graafilise kasutajaliidese tõttu päringuid tehes mugav kontrollida, kas tuvastatud faktid vastasid ootustele ja kui ei vastanud, siis oli kerge uurida, miks vastav lause valesti selle faktina tuvastati. Agregeerimisest oli kasu näiteks andmete korrektsuse kontrollimisel. Andmete jagunemise ebaloogilisus võis tähendada viga andmete töötlemisel.

4.2.2. Segmenteerimine

Suurem osa kohtuotsuste tekstist sisaldas kas ühte või mitut järgnevatest informatsioonigruppidest:

- otsuse resolutsioon, milles kirjeldati menetluse lõpptulemust. Selleks võis olla näiteks kohtualuse süüdi- või süütuksmõistmine, kaebuste ja muude avalduste rahuldamine või rahuldamata jätmine, eelnevalt tehtud otsuste tühistamine, muutmine ja muutmata jätmine, karistuse määramine ja tühistamine, abielu lahutamine või pankroti kuulutamine.
- menetluse asjaolud, kus kirjeldati põhjuseid, mille tõttu menetlus alustati. Näiteks narkootiliste ainete tarbimine või kaebuse esitamine.
- asjaosaliste seisukohad, milles võis välja tuua osapoolte kirjeldused toimunud sündmustest, nende põhjendused ning esitatavad nõuded.
- kohtu seisukoht, kus põhjendati tehtud kohtuotsust.
- menetluskulud, milles toodi välja, kes kohtumenetluse kulusid katab ja kui suures summas.
- edasikaebamise kord, kus seletati, tehtud kohtuotsuse edasikaebamise tingimusi, näiteks tähtaeg.

Üldiselt eelnesid nendele gruppidele tekstis sektsioonipealkirjad, näiteks enne resolutsiooniosa esines tihti sõna *"resolutsioon"*. Seetõttu oli mõeldav kohtuotsused vastavalt tekstilõikude sisule segmentideks jagada.

Sektsioonipealkirjade erinevate variantide leidmiseks kasutati TEXTA tööriista leksikonide loomise võimalust. Kui iga informatsioonigrupi jaoks olid sektsioonipealkirjade sõnastikud moodustatud, alustati grammatikate koostamisega kasutades Python'i teeki EstNLTK.

Esimese sammuna defineeriti iga informatsioonigrupi jaoks *Regex* objekt, millega oli võimalik tuvastada kõiki vastavale grupile kuuluvaid sektsioonipealkirju. Pealkirjadeks loeti leitud leksikonide sõnu vaid siis, kui need olid kas esisuurtähega või läbiva suurtähega. Seejärel moodustati kõigist kuuest *Regex* objektist *Gaps* objekt, millele rakendati funktsiooni *get_matches* iga kohtulahendi teksti lõigu puhul. Ühe segmendina arvestati teksti, mis jäi kahe sektsioonipealkirja vahele.

Elasticsearch andmebaasis olevasse kohtuotsuste tabelisse lisati uus tunnus *"segments"*, mille väärtuseks oli segmenteeritud tekst. Igale segmendile anti pealkiri vastavalt sellele, millise *Regex* objekti abil vastav segment tuvastati. Kui tekstist ei leitud ühtegi segmenti, pandi välja väärtuseks *"NONE"*.

Järgmisena on näha ühe teksti segmenteerimise tulemust, kus segmendi pealkiri on märgitud rasvases kirjas:

> ***edasikaebamise_kord***
> *Edasikaebamise kord*
> *Kohtuotsuse peale võib edasi kaevata kassatsiooni korras Riigikohtule, teatades sellest seitsme päeva jooksul Harju Maakohtule arvates otsuse lõpposa kuulutamisest ning esitades advokaadi vahendusel kassatsioonkaebuse Riigikohtule Harju Maakohtu kaudu kolmekümne päeva jooksul arvates otsuse tutvumiseks kättesaadavusest.*
> *Kassatsioonkaebuse teate esitamise korral on kohtuotsus tervikuna menetlusosalistele kättesaadav 21.05.2014.a.*
> *Kohus juhindub Väärteomenetluse seadustiku § 111; §132 p 2; § 133-135.*
>
> *Piret Liivo Kohtunik*
>
> ***resolutsioon***
> *RESOLUTSIOON*
> *Tühistada Politsei- ja Piirivalveameti Põhja Prefektuuri Korrakaitsebüroo liiklusjärelevalvekeskuse liiklusmenetlustalituse 21.03.2014.a. otsus väärteoasjas nr. 2302,13,013898 lisakaristuse määramise osas.*
>
> *Liiklusseaduse § 223 lg.1 alusel määratud karistus -rahatrahv 130 trahviühikut, s.o. 520 eurot- jätta muutmata.*
> *Täpsustada otsust ning määrata, et KarS§ 66 lg.3 alusel kuulub rahatrahv kokku summas 520 eurot tasumisele osiiti igakuiselt 10 kuu jooksul, s.o. 52 eurot kuus alates kohtuotsuse jõustumisest.*

4.2.3. Faktituvastus

Eraldatud segmentidest oli kõige huvipakkuvam resolutsiooniosa, mis oli käesoleva töö põhifookuseks. Kohtuotsuste resolutsioonist võib leida infot süüdi ja süütuks mõistmiste, avalduste ja kaebuste rahuldamiste ja rahuldamata jätmiste, kohtulahendite muutmise ja muutmata jätmise ja lahendite või muude dokumentide tühistamise kohta. Samuti on resolutsiooniosas tihti kirjeldatud, kas ja milline karistus mõisteti või tühistati ja kui palju kostjalt raha välja mõisteti.

Sellest lähtuvalt oli järgmiseks sammuks resolutsioonist lausete tuvastamine vastavalt sellele, millist informatsiooni nad sisaldasid. See on saavutatud, nagu segmentide leidmiselgi, grammatikatega.

Faktituvastuse grammatikate moodustamisel kasutati enamasti regulaaravaldiste klassi *IRegex*, kuna vaadeldavate sõnade puhul ei olnud teada, kas need esinesid lause alguses või keskel, seega oli vaja tekstist vasteid otsida suur- ja väiketähtedest sõltumatult. Samuti ei olnud kohtulahendite tekstid korrapäraselt vormistatud, mis tähendas, et lausete keskel esines ka suurtähega sõnu.

Käesolevas töös on toodud näidetes isikuandmed asendatud *"X"*-ga, kui neid juba eelnevalt pole anonümiseeritud.

Süüdi- ja süütuksmõistmise tuvastamine

Süütegusid puudutavates kohtuotsustes leidub tihti lause selle kohta, kas kohtualune tunnistati süüdi või mitte. **Süüdimõistmist** väljendavate lausete tuvastamiseks moodustati kõigepealt objektid, mida on näha järgmises koodilõigus:

```
space = Regex('\s')
syydi = IRegex(u'süüdi(?=[ \.,;:])')
tunnistada_moista = IRegex(u'tunnistada|mõista')
```

Kõigepealt defineeriti tühikut tuvastav *Regex* objekt *space*. Seejärel sõna *"süüdi"* tuvastav *IRegex* objekt *syydi*, kusjuures sellele võis vahetult järgneda vaid tühik või kirjavahemärk. Keelates sõnale sümbolite, mis ei ole tühik ega kirjavahemärk, järgnemist, on võimalik vältida vastava sõna tuvastamist mingis teises vormis. Viimasena defineeriti *IRegex* objekt *tunnistada_moista*, mis tuvastab sõnu *"tunnistada"* ja *"mõista"*.

Järgnevalt moodustati süüdimõistmist väljendavaid lauseid tuvastav *Union* objekt nimega *"tunnistada_süüdi"*, mida on näha järgmises koodilõigus:

```
guilty_matches = Union(
        Concatenation(syydi, space, tunnistada_moista),
        Gaps(tunnistada_moista, syydi),
        name='tunnistada_süüdi'
)
```

Grammatika koostati kahest objektist, millest esimene oli *Concatenation* objekt. See tuvastas tekstist lauseid, kus sõnale *"süüdi"* järgnes vahetult tühik ja seejärel sõna *"tunnistada"* või *"mõista"*. Kuna nende vahel muid sõnu ei saanud esineda, oli võimalik valida *Gaps* asemel *Concatenation* objekt, mille tulemusena leiti näiteks lause

"Süüdi tunnistada X KarS § 121 järgi.".

Teiseks objektiks oli *Gaps*, mis tuvastas lauseid, kus sõnale *"tunnistada"* või *"mõista"* järgnes sõna *"süüdi"*, kuid nende vahel võis olla veel piiramatu hulk teisi sõnu. Tihti oli nendeks sõnadeks näiteks kohtualuse nimi nagu on näha selle objekti abil tuvastatud lauses

"Tunnistada X süüdi KarS § 424 järgi ja karistada vangistusega 6 kuud.".

Süütuks mõistmist väljendavate lausete tuvastamiseks moodustati kõigepealt IRegex objektid, mis on välja toodud järgmises koodilõigus:

```
moista = IRegex(u'mõista')
oigeks = IRegex(u'õigeks')
```

Sellega defineeriti *IRegex* objektid *moista*, mis tuvastas sõna *"mõista"*, ning *oigeks*, mis tuvastas sõna *"õigeks"*. Kuna tühikut tuvastav objekt *space* oli juba eelnevalt defineeritud, siis ei olnud vajadust seda uuesti teha. Sõna *"tunnistada"* see kord ei otsitud, kuna väljendit *"õigeks tunnistada"* ei kasutatud.

Seejärel koostati süütuks mõistvaid lauseid tuvastav Union objekt nimega *"tunnistada_süütuks"*, mida on näha järgmises koodilõigus:

```
notguilty_matches = Union(
        Concatenation(oigeks, space, moista),
        Gaps(moista, oigeks),
        name='tunnistada_süütuks'
)
```

Grammatika moodustati kahest objektist, millest esimene oli *Concatenation* objekt. Selle abil leiti lauseid, kus sõnale *"õigeks"* järgnes vahetult tühik ja seejärel sõna *"mõista"*. Kuna nende sõnade vahel ei saanud esineda muid sümboleid eelistati *Concatenation* objekti *Gaps* objektile. Selle abil tuvastati näiteks lause

"Õigeks mõista X KarS § 169 järgi.".

Teisena defineeriti *Gaps* objekt, mis tuvastas lauseid, kus sõnad "mõista" ja "õigeks" esinesid vastavas järjekorras ning nende vahel võis leiduda veel muid sõnu, näiteks

"Mõista X KarS § 289 ja 380 järgi õigeks.".

Avalduste rahuldamise ja rahuldamata jätmise tuvastamine

Mistahes põhjusel kohtusse pöördumiseks tuleb koostada avaldus, milles on kirjeldatud kohtusse pöördumise põhjused ning mis lahendust kohtusse pöörduja vastavale probleemile soovib. Pärast kohtumenetlust teeb kohtunik otsuse kas avaldus rahuldada, osaliselt rahuldada või üldse mitte rahuldada.

Avalduste rahuldamist ja osaliselt rahuldamist väljendavate lausete tuvastamiseks moodustati kõigepealt objektid, mida on näha järgmises koodilõigus:

```
avaldus = Union(IRegex(u'hagid*(?=[ \.,:;])'),
IRegex(u'apellatsioon(id)*'),
IRegex(u'(määrus|apellatsioon|kassatsiooni)*kaebus(ed)*(?=[ \.,:;])'),
IRegex(u'taotlus(ed)*(?=[ \.,:;])'),
IRegex(u'avaldus(ed)*(?=[ \.,:;])'))
rahuldada = IRegex(u'rahuldada')
```

Esimesena defineeriti *Union* objekt *avaldus*, mis koosnes mitmest *IRegex* objektist. Selle abil tuvastati sõnu *"hagi(d)"*, *"apellatsioon(id)"*, *"kaebus(ed)"*, *"taotlus(ed)" ja "avaldus(ed)"*. Samuti leiti selle objektiga liitsõnu, mille esimene pool oli kas *"määrus", "apellatsioon"* või *"kassatsiooni"* ning teine pool *"kaebus(ed)"*. Kõigile otsitavatele sõnadele, välja arvatud sõnale *"apellatsioon",* võis vahetult järgneda vaid tühik või kirjavahemärk. Teisena moodustati *IRegex* objekt *rahuldada*, mis tuvastas sõna *"rahuldada"*.

Seejärel moodustati avalduste rahuldamist väljendavaid lauseid tuvastav Union objekt nimega *"avaldus_rahuldada"*, mida on näha järgmises koodilõigus:

```
accepted_matches = Union(
       Gaps(rahuldada, avaldus),
       Gaps(avaldus, rahuldada),
       name='avaldus_rahuldada'
)
```

Grammatika moodustati kahest *Gaps* objektist. Neist esimene tuvastas lauseid, kus sõnad *"rahuldada"* ja üks objekti *avaldus* poolt tuvastatav sõna esinesid vastavas järjekorras ning nende vahel võis olla piiramatu arv muid sümboleid, näiteks

"Rahuldada Narva Linna avaldus.".

Teine objekt tuvastas lauseid, kus sõnad üks objekti *avaldus* poolt tuvastatav sõna ja *"rahuldada"* esinesid vastavas järjekorras ning nende vahel esines mistahes arv sümboleid, näiteks

"Hagiavaldus elatise nõudes rahuldada.".

Avalduste rahuldamata jätmist väljendavate lausete tuvastamiseks moodustati kõigepealt IRegex objektid, mida on näha järgmises koodilõigus:

```
jatta = IRegex(u'jätta')
rahuldamata = IRegex(u'rahuldamata')
```

Esimesena defineeriti objekt *jatta*, mis tuvastab sõna *"jätta"*, ning teisena objekt *rahuldamata*, mis tuvastas sõna *"rahuldamata"*. Seejärel koostati avalduste rahuldamata jätmist tuvastav *Gaps* objekt nimega *"avaldus_mitterahuldada"*, mis on välja toodud järgmisel koodireal:

```
rejected_matches = Gaps(jatta, rahuldamata,
name='avaldus_mitterahuldada')
```

Selle grammatika abil tuvastati lauseid, kus sõnad *"jätta"* ja *"rahuldamata"* esinesid vastavas järjekorras, näiteks lauses

"Jätta prokuröri taotlus X süüdimõistmiseks KarS § 323 järgi rahuldamata.".

Erinevalt grammatikast *"avaldus_rahuldada"*, ei otsitud lausest kõiki avalduse tüüpe, mida on võimalik rahuldamata jätta, kuna tuvastatud lausete arv oleks selle lisamisel muutunud piisavalt vähe, et olla oluline. Kasutades erinevaid avalduse tüüpe tuvastavat *IRegex* objekti, oleks käesolev grammatika koosnenud kolmest *Gaps* objektist. Nendest oleks kõik omakorda koosnenud kolmest *IRegex* objektist, kuna avaldust märkiv sõna võib olla sõnade *"jätta"* ja *"rahuldada"* ees, keskel kui ka järel. Programmi tööaeg suureneb tunduvalt, kui objekt *Gaps* koosneb rohkem kui kahest objektist, seega otsustati avaldust märkivat sõna mitte otsida.

Dokumentide muutmise, muutmata jätmise ja tühistamise tuvastamine

Avalduse rahuldamise tulemusena võib olla vajalik eelnevalt tehtud otsuseid ja määruseid muuta või tühistada. **Otsuste ja määruste muutmist** väljendavate lausete tuvastamiseks moodustati kõigepealt *IRegex* objektid, mida on näha järgmises koodilõigus:

```
maarus_otsus = IRegex(u'määrus|(kohtu)*otsus(?=[ \.,:;])')
muuta = IRegex(u'muuta')
```

Kõigepealt moodustati *IRegex* objekt *maarus_otsus*, mille abil leiti sõnu *"määrus"* ja *"(kohtu)otsus"*, kusjuures sõnale *"otsus"* võis vahetult järgneda vaid tühik või kirjavahemärk. Järgmisena defineeriti *IRegex* objekt *muuta*, mille abil tuvastati sõna *"muuta"*.

Seejärel moodustati otsuste ja määruste muutmist väljendavaid lauseid tuvastav *Union* objekt nimega *"lahend_muuta"*, mis on välja toodud järgmises koodilõigus:

```
changed_matches = Union(
        Gaps(muuta, otsus_maarus),
        Gaps(otsus_maarus, muuta),
        name='lahend_muuta'
)
```

Grammatika koosnes kahest *Gaps* objektist. Neist esimene tuvastas lauseid, kus sõnad *"muuta"* ja *"määrus"*, *"kohtuotsus"* või *"otsus"* esinesid vastavas järjekorras ning nende vahel võis leiduda piiramatu arv teisi sümboleid. Näiteks nagu lauses

"Muuta Pärnu Maakohtu tsiviilasjas nr 2-08-8215 21.aprillil 2008 tehtud kohtumäarust.".

Teine objekt tuvastas lauseid, milles sõna *"otsus"*, *"kohtuotsus"* või *"määrus"* ja sõna *"muuta"* esinesid vastavas järjekorras, näiteks

"Otsus menetluskulude jaotuse osas muuta.".

Kohtuotsuse resolutsiooniks võib olla ka eelnevalt tehtud otsuse või määruse muutmata jätmine, kui edasikaebamisega ei ole saavutatud mingeid tulemusi. **Otsuste ja määruste** (osaliselt) **muutmata jätmist** väljendavate lausete tuvastamiseks moodustati kõigepealt *IRegex* objektid, mida on näha järgmises koodilõigus:

```
muutmata = IRegex(u'muutmata')
jaab = IRegex(u'jääb|jätta')
```

Esimesena defineeriti *IRegex* objekt *muutmata*, mis tuvastas sõna *"muutmata"*, ning teisena *IRegex* objekt *jaab*, mis tuvastas sõnu *"jääb"* ja *"jätta"*.

Seejärel moodustati otsuste ja määruste muutmata jätmist väljendavaid lauseid tuvastav *Gaps* objekt nimega *"jätta_lahend_muutmata"*, mis on välja toodud järgmisel koodireal:

```
unchanged_matches = Gaps(jaab, muutmata, name='jätta_lahend_muutmata')
```

Käesoleva grammatika põhjal tuvastati lauseid, milles sõnad *"jääb"* või *"jätta"* ja *"muutmata"* esinesid vastavas järjekorras, näiteks nagu lauses

"Jätta Tallinna Halduskohtu 14.05.2010 otsus muutmata.".

Otsuste ja määruste tühistamist väljendavate lausete tuvastamiseks koostati kõigepealt *IRegex* objekt *tyhistada*, mis tuvastas sõna *"tühistada"*. Seda objekti on näha järgmisel koodireal:

```
tyhistada = IRegex(u'tühistada')
```

Kuna sõnu *"määrus"* ja *"(kohtu)otsus"* tuvastav objekt *maarus_otsus* oli juba eelnevalt otsuste ja määruste muutmist väljendavate lausete leidmiseks defineeritud, siis polnud seda vaja uuesti teha. Seejärel moodustati otsuste ja määruste tühistamist sisaldavaid lauseid tuvastav *Union* objekt nimega *"lahend_tühistada"*, mida on näha järgmises koodilõigus:

```
cancelled_matches = Union(
       Gaps(tyhistada, maarus_otsus),
       Gaps(maarus_otsus, tyhistada),
       name='lahend_tühistada'
)
```

Grammatika koosnes kahest *Gaps* objektist. Nendest esimene tuvastas lauseid, kus kõigepealt esines sõna *"tühistada"* ja seejärel kas *"(kohtu)otsus"* või *"määrus"*, näiteks

> *"Tühistada Tartu Halduskohtu 22. märtsi 2010 otsus."*.

Teine objekt tuvastas lauseid, milles esimesena esines sõna *"(kohtu)otsus"* või *"määrus"* ja teisena *"tühistada"*, näiteks

> *"Maakohtu otsus ülejäänud osades tühistada ja teha uus otsus."*.

Otsused ja määrused on üldiselt pigem tõsisemate kohtumenetluste kohta ning ettekirjutused, korraldused ja käskkirjad pigem triviaalsemate olukordade kohta. Seega tuvastati nende tühistamisi eraldi.

Muude dokumentide tühistamist sisaldavate lausete tuvastamiseks moodustati kõigepealt *IRegex* objekt, mida on näha järgmisel koodireal:

```
muu_dok = IRegex(u'korraldus|ettekirjutus|käskkiri')
```

Defineeritud *IRegex* objekti abil tuvastati sõnu *"korraldus"*, *"ettekirjutus"* ning *"käskkiri"*. Kuna otsuste ja määruste tühistamist väljendavate lausete leidmiseks juba defineeriti sõna *"tühistada"* tuvastav objekt *tyhistada*, siis seda uuesti ei tehtud. Seejärel koostati muude dokumentide tühistamist sisaldavaid lauseid tuvastav *Union* objekt nimega *"muu_dokument_tühistada"*, mida on näha järgmisel koodilõigul:

```
cancelled_other_matches = Union(
       Gaps(tyhistada, muu_dok),
       Gaps(muu_dok, tyhistada),
       name='muu_dokument_tühistada'
)
```

Grammatika koosnes kahest *Gaps* objektist. Neist esimene tuvastas lauseid, kus sõnad *"käskkiri"*, *"korraldus"* või *"ettekirjutus"* ja sõna *"tühistada"* esinesid vastavas järjekorras, näiteks

> *"Kuna eelnimetatud puudused korralduses võisid mõjutada asja otsustamist, tuleb Tori Vallavalitsuse 12.06.2008 korraldus nr 162 tühistada."*.

Teise objekti abil leiti lauseid, kus sõna *"tühistada"* ning sõna *"käskkiri"*, *"korraldus"* või *"ettekirjutus"* esinesid vastavas järjekorras, näiteks

> *"Tühistada Ida Prefektuuri 11.02.2011 käskkiri nr 7d."*.

Karistuse määramise ja tühistamise tuvastamine

Kohtumenetlustes, milles on leitud, et kohtualune on süüdi väärteo või kuriteo toimepanemises, määratakse üldiselt ka sellele vastav karistus. Kuna edaspidises töös oli huvi täpsete karistuste eraldamiste vastu, näiteks vangistused, arestid ja rahatrahvid, oli kõigepealt vajadus sellist informatsiooni sisaldavad laused muust tekstist eraldada.

Määratud karistust sisaldavate lausete tuvastamiseks moodustati kõigepealt hulk *IRegex* objekte, mis on välja toodud järgmise koodilõiguga:

```
maarata = IRegex(u'määrata|mõista')
karistusena = IRegex(u'karistus(eks|ena)')
karistuseks = IRegex(u'karistus(eks|ena|(?=[ \.,;:]))')
karistada = IRegex(u'karistada')
alusel = IRegex(u'alusel|järgi|eest(?=[ ;:\.,])')
teda = IRegex(u'teda')
raha_karistus = IRegex(u'raha\w*( )*karistus')
kr_eur = IRegex(u'kr\w*|eur\w*|€')
```

Kõigepealt defineeriti *IRegex* objekt *maarata*, mis tuvastas sõnu *"määrata"* ja *"mõista"*. Objektiga *karistusena* leiti sõnu *"karistuseks"* ja *"karistusena"* ning objektiga *karistuseks* otsiti lisaks eelmisele kahele sõnale ka algvormi *"karistus"*, millele võis vahetult järgneda vaid tühik või kirjavahemärk. Objektiga *karistada* tuvastati sõna *"karistada"*, objektiga *alusel* sõnu *"alusel", "järgi"* ja *"eest"*, kusjuures sõnale *"eest"* võis vahetult järgneda vaid tühik või kirjavahemärk, ning objektiga *teda* tuvastati sõna *"teda"*. Samuti moodustati objekt *raha_karistus*, mille abil tuvastati fraasi, milles esimene sõna algas tähtedega *"raha"* ja teine sõna tähtedega *"karistus"*. Selle abil oli võimalik leida rahalise karistuse määramist mistahes vormis. Sarnaselt moodustati ka objekt *kr_eur*, millega tuvastati sõna, mis algas kas tähtedega *"kr"* või *"eur"*, kuna rahaühikuid esines nii lühendatud kui ka täispikkuses vormis. Samuti tuvastas see objekt euro sümboli *"€"* esinemist.

Seejärel koostati karistust sisaldavaid lauseid tuvastav Union objekt nimega *"mõista_karistus"*, mida on näidatud järgmisel koodilõigul:

```
punishment_matches = Union(
        Gaps(maarata, karistuseks),
        Concatenation(karistusena, space, maarata),
        Gaps(karistada, alusel),
        Concatenation(karistada, space, teda),
        Gaps(raha_karistus, kr_eur),
        name='mõista_karistus'
)
```

Grammatika koosnes viiest objektist, millest esimene oli *Gaps* objekt, mis tuvastas lauseid, milles sõna *"mõista"* või *"määrata"* ja sõna *"karistus"*, *"karistuseks"* või *"karistusena"* esinesid vastavas järjekorras, näiteks

"Mõista talle karistuseks vangistus kaks (2) kuud.".

Järgmiseks objektiks oli *Concatenation* objekt, mille abil leiti lauseid, kus sõnale *"karistuseks"* või *"karistusena"* järgnes vahetult tühik ning seejärel sõna *"määrata"* või *"mõista"*, näiteks lause

"Süüdi tunnistada X KarS § 121 järgi ja karistuseks mõista 6 kuud vangistust.".

Kolmandaks objektiks oli *Gaps* objekt, mis tuvastas lauseid, kus sõnast *"karistada"* mistahes kaugusel oli sõna *"alusel", "järgi"* või *"eest"*. Selliseks lauseks oli näiteks

"Süüdi tunnistada X NPAS § 151 järgi ja karistada kahe väärteo eest arestiga 10 päeva.".

Järgmine *Concatenation* objekt tuvastas lauseid, kus sõnale *"karistada"* järgnes vahetult tühik ning seejärel sõna *"teda"*, näiteks

> *"Tunnistada X süüdi KarS § 262 ettenähtud väärteo toimepanemises ning* karistada teda *arestiga 12 (kaksteist) päeva."*

Viimaseks objektiks oli *Gaps* objekt, mille abil tuvastati lauseid, kus kõigepealt esines fraas, milles esimene sõna algas tähtedega *"raha"* ja teine sõna tähtedega *"karistus"*. Sellest fraasist mistahes kaugusel pidi esinema rahaühikut tähistav sümbol *"€"* või sõna, mis algas kas tähtedega *"kr"* või *"eur"*. Näiteks leiti selle objekti abil lause

> *"Süüdi tunnistada X KarS § 121 järgi ja karistada rahalise karistusega 240 miinimumpäevamäära, s.o. 12000 krooni."*

Kui kohtualusele määratakse kohtuotsusega mingi karistus, on tal võimalus otsust edasi kaevata, juhul kui ta ei ole sellega rahul või pärast otsuse tegemist on tekkinud lisainformatsiooni, mis tema süütust tõestaks. Kui edasikaebamisel leitakse, et kohtualune on süütu või tema sooritatud kuritegu või väärtegu on eelnevalt leitust väiksem, siis võidakse määratud karistus täielikult tühistada või tühistada ja asendada kergema karistusega. Mõlemal juhul toimub **karistuse tühistamine** ning sellist informatsiooni sisaldavate lausete tuvastamiseks koostati kõigepealt *IRegex* objekt, mida on näha järgmisel koodireal:

```
karistus = IRegex(u'rahatrahv|karistus(?=[ \.,;:])')
```

Kuna dokumentide tühistamist leidva grammatika jaoks oli juba sõna *"tühistada"* tuvastav objekt moodustatud, siis seda uuesti vaja ei olnud teha. Järgmisena moodustati karistuse tühistamist kirjeldavaid lauseid tuvastav *Union* objekt nimega *"karistus_tühistada"*, mida näitab järgmine koodilõik:

```
cancelled_punishment_matches = Union(
       Gaps(karistus, tyhistada),
       Gaps(tyhistada, karistus),
       name='karistus_tühistada'
)
```

Grammatika koosnes kahest *Gaps* objektist, millest esimesega leiti lauseid, kus kõigepealt esines sõna *"karistus"* või *"rahatrahv"* ja järgmisena sõna *"tühistada"*, näiteks

> *"LS § 224 lg 3 p 1 alusel mõistetud lisakaristus tühistada."*.

Teise objektiga tuvastati lauseid, kus sõna *"tühistada"* ja sõna *"karistus"* või *"rahatrahv"* esinesid vastavas järjekorras, näiteks

> *"Tühistada Põhja Politseprefektuuri korrakaitsebüroo avariitalituse 08.01.2010a. otsusega nr.2302,09,018781 Liiklusseaduse § 7417 lg.1 alusel X"le määratud karistus."*.

Kui kohtualuse toimepandud väärteos või kuriteos oli kannatajaid, pidi süüdlane tihti kannatanule põhjustatud kahjud hüvitama. Kohtulahendites kasutati selle väljendamiseks tihti sõnu *"mõista"* ja *"kasuks"*, kus sõna *"kasuks"* määras ära, kellele raha tuli maksta. Kohtualuselt raha välja mõistvaid lauseid oli ka selliseid, kus raha ei mõistetud kannatanule vaid riigile, näiteks trahvina või kohtukulude katmiseks. Sellistel juhtudel sõna *"kasuks"* ei esinenud, mistõttu oli selle sõna esinemine lauses järgmise fakti tuvastamiseks oluline.

Kannatanule **kahjude hüvitamise kohustust** väljendavate lausete tuvastamiseks defineeriti kõigepealt *IRegex* objekt, mida on näha järgmisel koodireal:

```
kasuks = IRegex(u'kasuks')
```

Kuna süütuks mõistmist väljendavate lausete tuvastamiseks oli juba objekt, mis tuvastas sõna *“mõista”*, moodustatud, ei olnud seda vaja uuesti teha.

Kannatanule kahjude hüvitamise kohustust väljendavaid lauseid tuvastati *Gaps* objektiga, mille nimeks oli *"välja_mõista"*. Seda objekti on näha järgmisel koodireal:

```
valja_moista_matches = Gaps(moista, kasuks, name='välja_mõista')
```

Käesoleva grammatikaga leiti lauseid, kus sõnad *“mõista”* ja *“kasuks”* esinesid lauses vastavas järjekorras, näiteks

“Välja mõista X 809.- krooni Eesti Haigekassa kasuks.”.

Eelpool loetletud grammatikaid rakendati segmenteerimisel leitud resolutsiooniosadele ja selle puudumisel kogu tekstile. Tulemusena tekkinud uue tunnuse väärtuseks olid tuvastatud laused, grupeeritud vastavalt grammatikale, mille põhjal need leiti. Igale grupile anti pealkiri, milleks oli vastavale grammatikale pandud nimi. Tuvastatud fakti muster ning kõik mustriosade vahel olevad sõnad märgiti rasvase kirjaga, näiteks:

tunnistada_süüdi
juhindudes KrMS § 248 lg 1 p 5 ***Süüdi tunnistada*** *X KarS § 424 järgi ning mõista karistuseks rahaline karistus 300 (kolmsada) päevamäära, s.o. 25 500 (kakskümmend viis tuhat viissada) krooni.*

mõista_karistus
juhindudes KrMS § 248 lg 1 p 5 Süüdi tunnistada X KarS § 424 järgi ning ***mõista karistuseks*** *rahaline karistus 300 (kolmsada) päevamäära, s.o. 25 500 (kakskümmend viis tuhat viissada) krooni.*
KarS § 50 lg 1 alusel ***mõista lisakaristus*** *ja võtta X-lt ära mootorsõiduki juhtimisõigus 8 (kaheksaks) kuuks.*

Kui kohtuotsuses ei esinenud ühtegi lauset, mis vastaks kirjeldatud grammatikatele, pandi tunnuse väärtuseks *"NONE"*.

4.2.4. Informatsiooni eraldamist raskendavad asjaolud

Kindlat informatsiooni sisaldavate lausete eraldamine on lihtsam probleem, kui nendest lausetest täpse informatsiooni eraldamine. EstNLTK objekti *Gaps* funktsiooni *get_matches* kasutamine oli selle probleemi lahendamiseks üldjuhul parim meetod, kuna see võimaldas tuvastada lauseid, kus etteantud sõnad esinesid üksteisest suvalisel kaugusel. *Gaps* objekti laialdase kasutamise tõttu olid ka selle probleemid kõige märgatavamad.

Suurimaks probleemiks faktituvastusel objektiga *Gaps* oli valepositiivsete leidmine liitlausete tõttu. See tähendab, et kui liitlauses oli kaks erinevat fakti, siis vajalike sõnade olemasolul võidi nendest kahest kokku tuvastada fakt, mida tegelikult lauses ei esinenud. Näiteks olgu lause:

“VTMS § 132 lg 2 alusel rahuldada X-i kaebus karistuse määra osas, ***tühistada Põhja Prefektuuri Korrakaitsebüroo Liiklusmenetlustalituse 02.10.2012.a otsus väärteoasjas***

nr. 2302,12,013940 ja teha uus otsus – määrata X-le LS § 205 alusel rahatrahv *8 trahviühiku ulatuses, so 32 eurot, mis tasuda 30 päeva jooksul alates kohtuotsuse kättesaamisest (kui ei esitata kassatsiooni) Rahandusministeeriumi arvele 10220034796011 SEB Pangas või arvele 221023778606 Swedbank pangas."*

Eelnevas lauses tuvastatakse valesti rahatrahvi tühistamine. Tegelikult koosnes valepositiivsena tuvastatud lauseosa kahest faktist: otsuse tühistamine ja rahatrahvi määramine.

Selle takistuse lahendamiseks oleks hea olnud näiteks võimalus defineerida sõnu, mis ei tohi otsitavate sõnade vahel esineda. Samuti oleks probleemi lahendanud lause osalauseteks jagamine.

Lisaks oli raskendavaks teguriks funktsiooni *get_matches* rakendamine, kui objektile *Gaps* antud argumente oli rohkem kui kaks. Selle tagajärjel pikenes programmi tööaeg märgatavalt ning kuna grammatikate koostamine toimus täiendamise ja kontrollimise korduvast tsüklist, siis vähenes töö efektiivsus tunduvalt.

Laialtlevinud probleemiks tekstikaeves on kirjavigade esinemine, mis võivad olla lihtsad trükivead või tuleneda halvast keeleoskusest. Näiteks kirjaviga lauses

"Jääta Harju Maakohtu 04.12.2006 otsus muutmata ja apellatsioonkaebus rahuldamata."

ja omastava käände valesti kasutamine lauses

"Süüdi tunnistada X KarS § 329 järgi kvalifitseeritava kuriteo toimepanemises ning mõista karistuseks 6 kuulise vangistuse.".

Samuti leidub ka juhuseid, kus kirjutaja ei ole olnud tähelepanelik ning jätnud mingi sõna kirjutamata, näiteks

"Lõplikuks karistuseks mõista X"ile 30 (kolmkümmend) vangistust.",

kus on tõenäoliselt sõna *"päeva"* unustatud kirjutada.

Kuna kohtulahenditel on üldiselt kindel ülesehitus, on tihti uue lahendi moodustamiseks mõeldav eelmistest lahenditest teatavate osade kopeerimine uude dokumenti. Sellega kaasneb aga oht ühes dokumendis tehtud kirjaviga teistesse lahenditesse edasi kanda. Kuna tegemist ei ole enam üksiku kirjaveaga vaid veaga mis esineb potentsiaalselt suures hulgas dokumentides, on vajalik kas kirjaviga nendes lahendites ära parandada või vigase sõnaga grammatika koostamisel arvestada. Näiteks esines ühe kohtuniku poolt koostatud kohtulahendites kirjaviga

"Süüdi tunnistada X KarS § 424 järgi ja mõista karistuseks neli (4) kuud vangitust.",

ehk sõna *"vangistus"* asemel oli *"vangitus"*.

Probleeme valmistas ka kohtulahendite teksti lauseteks jagamine. Selle tõttu jäid mõned faktid tuvastamata, kuna teksti lausestamise tulemusena jagunes üks fakt kahte või enamasse tulemusena saadud lausesse. Näiteks jagati üks lause valesti kaheks osaks, millest esimene oli

"Süüdi tunnistada ja mõista X-le väärteoasjades nr. 2780,14,003331 ja nr."

ja teine

"2780,14,003271 karistuseks liiklusseaduse § 201 lg 2 ja § 224 lg 2 järgi, kohaldades KarS § 63 lg 1, arest kakskümmend (20) päeva."

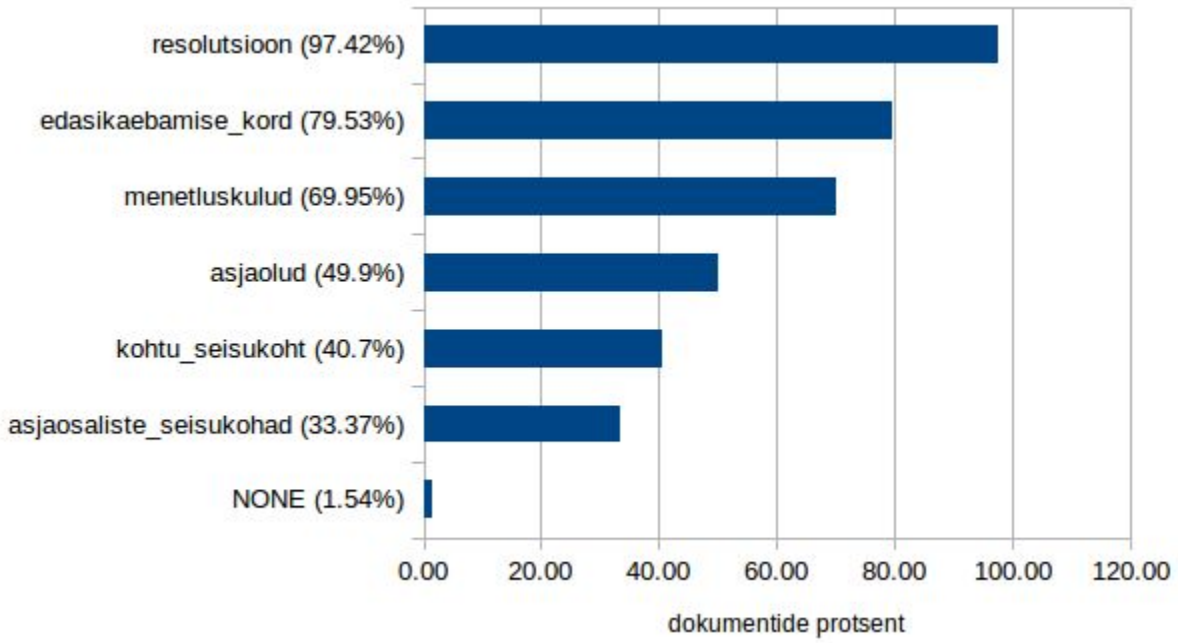
Kuna eelnev lause jagati ekslikult mitmeks osaks, jäi karistuse määramise fakt tuvastamata.

5. Analüüs

Kohtuotsuste töötlemise tulemusena tekkis andmebaas, mille tunnuste põhjal päringuid tehes oli võimalik leida palju eelnevalt teadmata informatsiooni.

5.1. Segmenteerimise tulemused

Pärast kohtuotsuste teksti segmenteerimist loeti kokku, mitmes dokumendis iga segment esines. Selleks koostati päring, mis otsis segmenteerimise tulemusena saadud kohtuotsuse tekstist vastava segmendi pealkirja. Järgmisel joonisel [joonis 9] on välja toodud, mitmes protsendis andmebaasis olevast 177 453-st kohtuotsusest iga segment esines:



Joonis 9. Segmenteerimise tulemused.

Nagu näha, oli resolutsiooniosa välja toodud peaaegu kõikides dokumentides ehk 97.42%-s, ning seda ei leitud vaid 2.58%-st otsustest. Kuna üldiselt kohtuotsusega siiski langetatakse mingi otsus, siis tõenäoliselt ei eraldatud nendes dokumentides resolutsiooniosa piisavalt selgelt, et segmenteerimisel seda tuvastatud oleks.

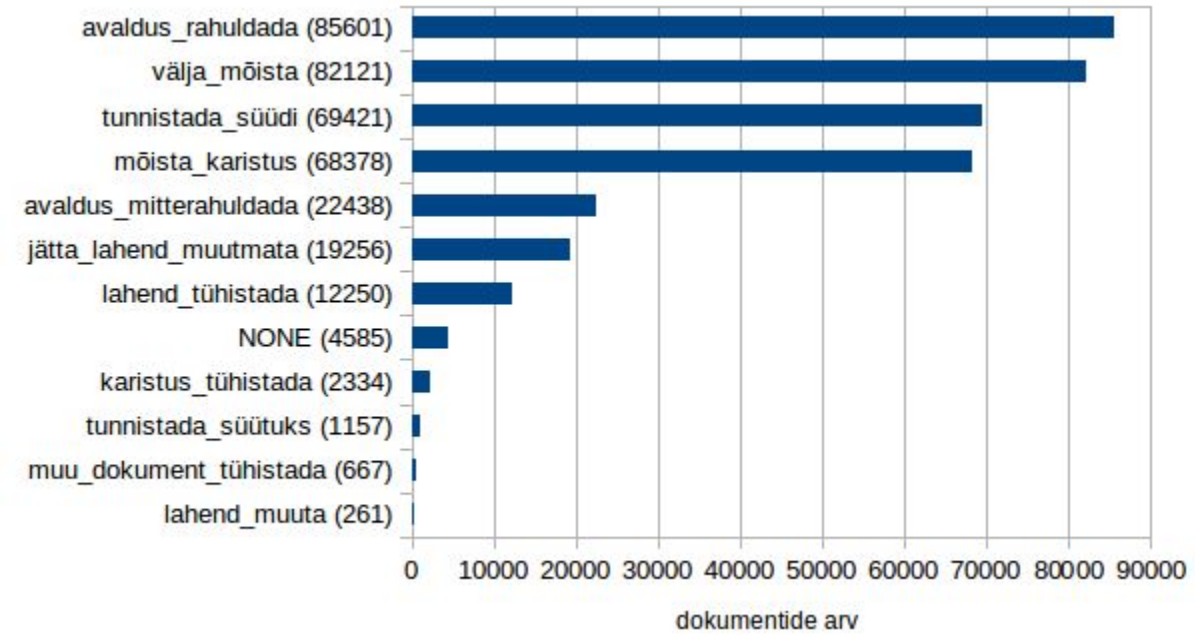
Dokumente, milles edasikaebamise korda väljendav tekstilõik tuvastati, oli 79.53%. Otsuste hulgas, millest vastavat segmenti ei tuvastatud, esines nii dokumente, milles edasikaebamise korda üldse ei mainitud kui ka neid, milles seda mainiti vaid ühe lausega ning ülejäänud tekstist selgelt ei eraldatud.

Sarnaselt tuvastati segmenti, milles määrati menetluskulude jaotumine, 69.95%-st kohtuotsustest, ning segmenti, milles kirjeldati kohtumenetluse asjaolusid, 49.9%-st kohtuotsustest. Kohtu seisukohta väljendav tekstilõik tuvastati 40.7%-st dokumendist, ning asjaosaliste seisukohta väljendav tekstilõik 33.37%-st otsusest.

Mitte ühtegi segmenti ei leitud 1.54%-st kohtuotsustest, mis võib tähendada, et nende otsuste ülesehitus ei olnud enam poolstruktureeritud vaid täielikult struktureerimata ning otsitavaid sektsioonipealkirju ei esinenud.

5.2. Faktituvastuse tulemused

Sarnaselt segmenteerimise tulemuste kokkulugemisele oli ka iga tuvastatud fakti puhul võimalik välja uurida, mitmes kohtuotsuses see esines. Selleks oli vaja koostada päring, mis leidis kirjed, kus vastavat fakti tuvastava grammatika nimi esines lausetegrupi pealkirjana. Järgmisel joonisel [joonis 10] on näha mitmest kohtuotsusest, mida oli kokku 177 453, iga fakt tuvastati:



Joonis 10. Faktituvastuse tulemused.

Nagu jooniselt näha, tuvastati kõige enam avalduste rahuldamist, kus avalduseks võis olla hagi, apellatsioon, kaebus või muu. Seda tuvastati kokku 85 601-st kohtuotsusest, mis moodustas 48.24% kogu andmehulgast, ehk peaaegu pooltes kohtulahendites rahuldati esitatud avaldus. Avaldused jäeti rahuldamata 22 438-s kohtuotsuses, mis moodustas 12.64%. Seega avalduste rahuldamist esines tunduvalt rohkem kui rahuldamata jätmist.

Kohtualuselt kannatanu kasuks raha välja mõistmist tuvastati 82 121-s kohtulahendis. Kohtualuse süüdi tunnistamist väljendavaid lauseid tuvastati 69 421-s dokumendis, karistust mõistvaid lauseid aga 68 378-s kohtuotsuses. Ekslikult võib eeldada, et kui kohtualune tunnistatakse süüdi, siis määratakse talle ka asjakohane karistus, kuid tuleb välja, et see ei ole tõsi. Esineb kohtuotsuseid, milles kohtualune tunnistatakse süüdi, kuid otseselt karistust ei määrata, vaid näiteks määratakse kontrollnõuded, mida süüalune peab täitma teatud ajaperioodi vältel. Samuti esineb ka kohtuotsuseid, milles ei mainita, et kohtualune on süüdi, kuid määratakse talle karistus. Süütuks mõistmist leidus vaid 1 157-s dokumendis ning karistuse tühistamist ainult 2 334-s dokumendis.

Eelnevalt koostatud lahendeid otsustati muuta 261-l korral ning tühistada 12 250-l korral. Muutmata jäeti need 19 256-l korral, millest võib järeldada, et kohus pigem jätab eelnevalt tehtud

otsuste muutmise nõudmised täitmata. Korraldusi, käskkirju ning ettekirjutisi tühistati 667-l korral.

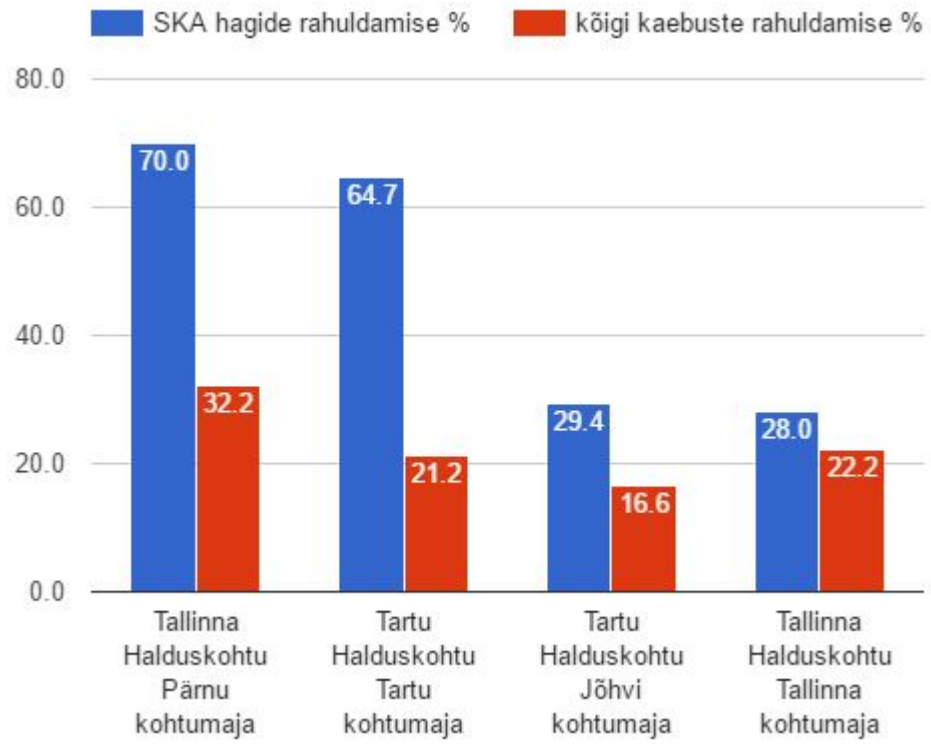
Mitte ühtegi tuvastatavatest faktidest ei leitud 4 585-st kohtuotsusest, milles esines abielulahutusi, pankrotikuulutamisi, tunnistamisi, kas kohtualune on väidetaval ametikohal töötanud, mille puhul on tegemist töövaidlustega, otstarbekuse kaalutlustel menetluse lõpetamisi, vanema õiguste äravõtmisi ja muud. Neid fakte käesolevas töös ei uuritud.

5.3. Näidisanalüüsid

Halduskohtutes kriminaal- ega väärteomenetlustega ei tegeleta, seega seal kohtualuseid süüdi ega süütuks ei mõistetud. Esines aga avalduste rahuldamisi ja mitterahuldamisi.

5.3.1. Avalduste rahuldamise määr maa- ja ringkonnakohtutes

Järgmisel joonisel [joonis 11] on iga halduskohtu kohta välja toodud, mitu protsenti kõigist esitatud avaldustest rahuldati ja mitu protsenti sotsiaalkindlustusameti (SKA) vastu esitatud avaldustest rahuldati.

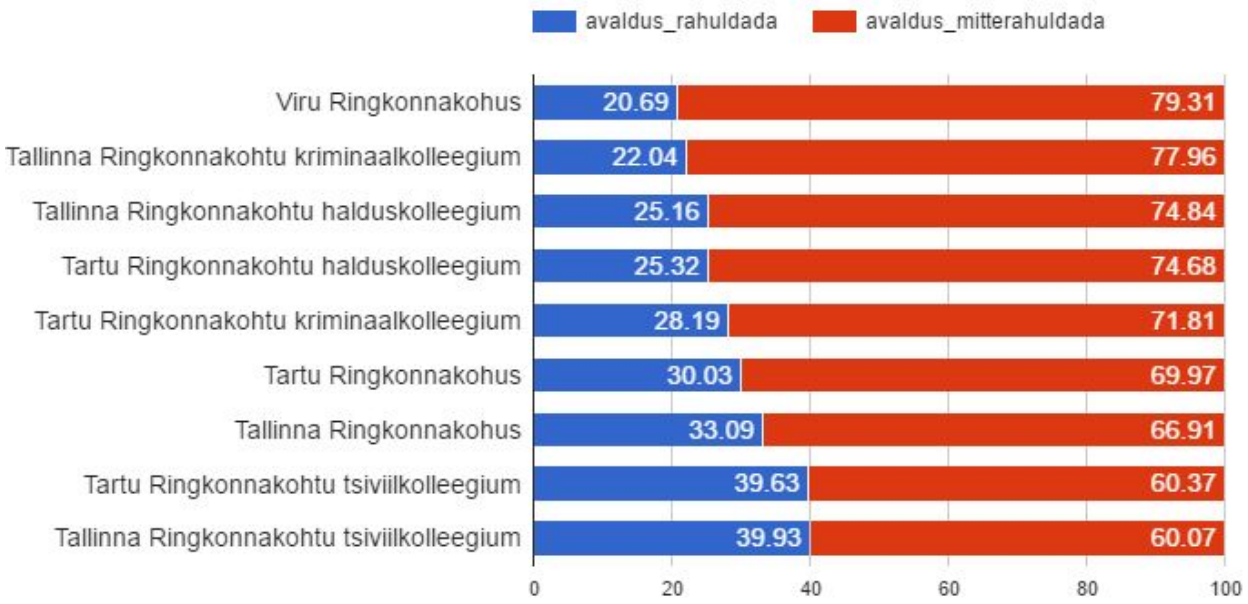


Joonis 11. Sotsiaalkindlustusameti hagide rahuldamine ja kõigi kaebuste rahuldamine halduskohtutes.

Nagu jooniselt on näha, on igas halduskohtus suurem osakaal rahuldamata jäetud avaldustel kui rahuldatud avaldustel. Kõige suurem protsent, milleks oli 32.2%, esitatud avaldustest rahuldati Tallinna Halduskohtu Pärnu kohtumajas ning kõige väiksem protsent, mis oli 16.6%, Tartu Halduskohtu Jõhvi kohtumajas. Tartu Halduskohtu Tartu kohtumajas ja Tallinna Halduskohtu Tallinna kohtumajas, kuhu saabus suurim osa avaldustest, rahuldati neist peaaegu sama protsent, vastavalt 21.2% ja 22.2%.

Sotsiaalkindlustusameti vastu esitatud avalduste rahuldamise jaotus oli tunduvalt ebaühtlasem. Tallinna Halduskohtu Pärnu kohtumajas rahuldati neist koguni 70% ning Tartu Halduskohtu Tartu kohtumajas 64.7%, mis tähendab et suurem osa SKA vastu esitatud avaldustest rahuldati. Vastupidiselt sellele, rahuldati Tartu Halduskohtu Jõhvi kohtumajas vaid 29.4% ning Tallinna Halduskohtu Tallinna kohtumajas 28% SKA vastu esitatud avaldustest, millest viimases olid SKA vastaste avalduste ja kõigi avalduste rahuldamise protsendid teineteisele kõige lähemal.

Samuti on võimalik vaadelda avalduste rahuldamist ja rahuldamata jätmist ringkonnakohtutes, mis on välja toodud järgmisel joonisel [joonis 12]. Nendesse kohtutesse ei jõudnud piisavalt SKA vastu esitatud avaldusi, et selle põhjal mingeid järeldusi teha.

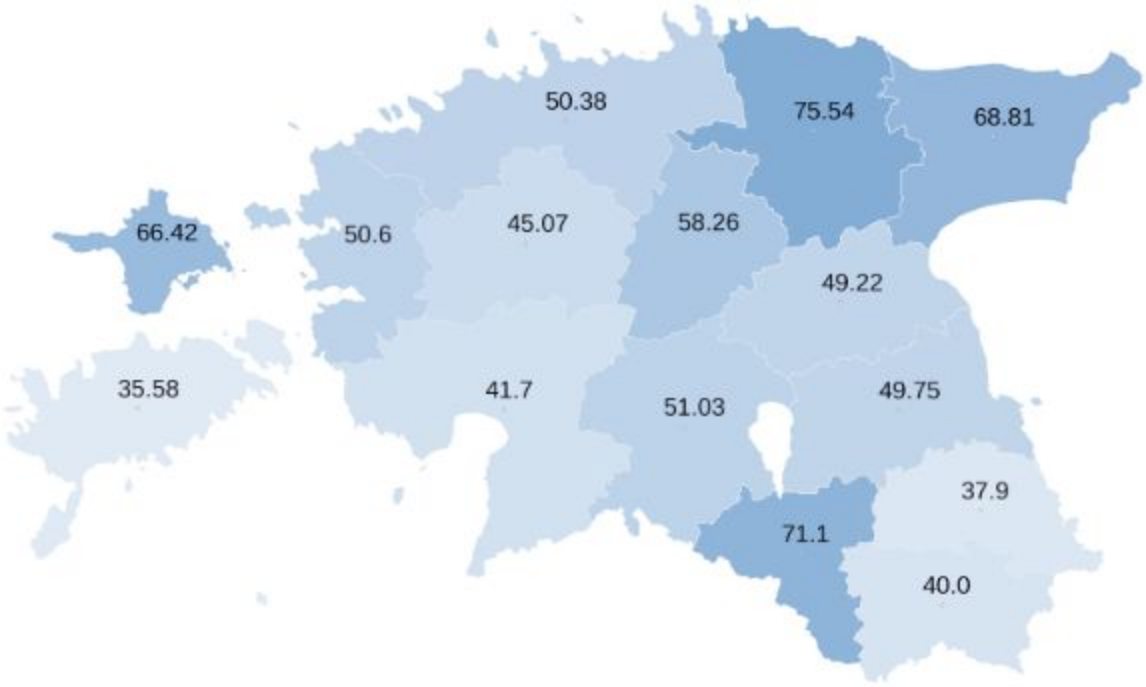


Joonis 12. Ringkonnakohtutes avalduste rahuldamine ja rahuldamata jätmine.

Nagu jooniselt on näha, on ringkonnakohtute avalduste rahuldamise protsendid sarnaselt halduskohtutele väiksemad kui avalduste rahuldamata jätmise protsendid. Kõige rohkem avaldusi on rahuldatud tsiviilkolleegiumites, kus nii Tartu Ringkonnakohtu ja ka Tallinna Ringkonnakohtu tsiviilkolleegiumis on rahuldatud umbes 40% esitatud avaldustest. Vähem avaldusi on rahuldatud haldus- ja kriminaalkolleegiumites, kuid kõige väiksem protsent esitatud avaldustest on rahuldatud Viru Ringkonnakohtus.

5.3.2. Süüdimõistmised maakohtutes

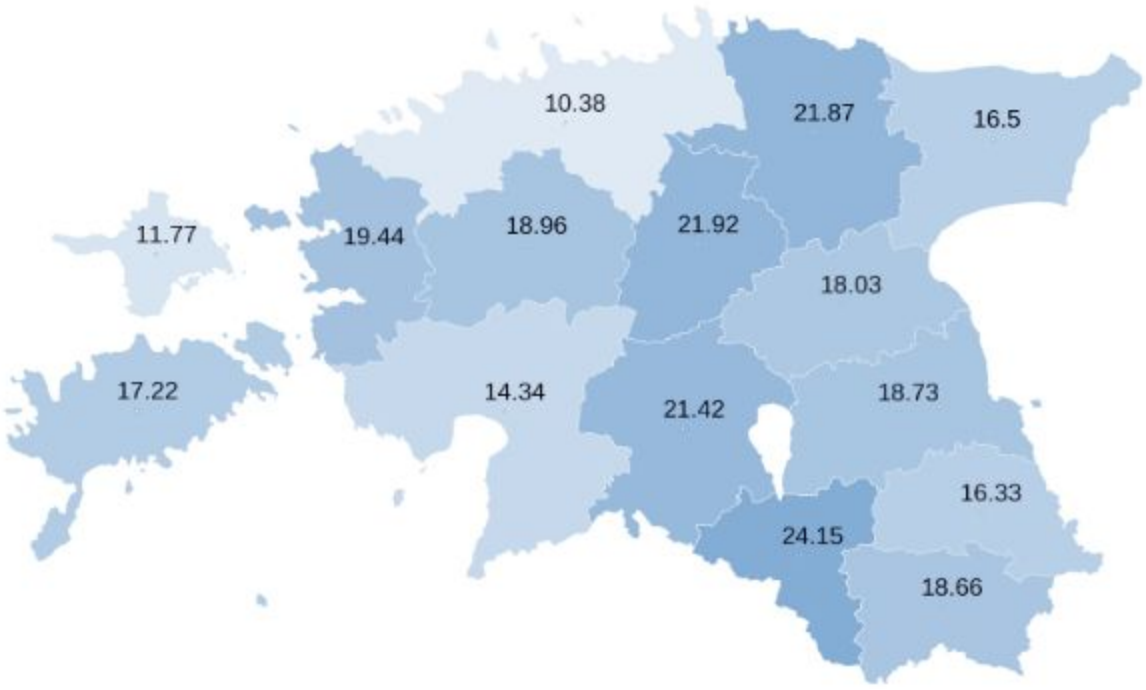
Iga maakonna jaoks on määratud kindlad maakohtud, mis selle maakonna kohtuasjadega tegelevad. Seetõttu on võimalik vaadelda kohtumenetluste tulemusi maakondade lõikes. Järgmisel joonisel [joonis 13] on välja toodud, mitu süüdimõistmist oli 1000 elaniku kohta vastavas maakonnas aastatel 2006-2015. Iga maakonna rahvaarv on võetud 2015. aasta seisuga Eesti statistikaameti veebilehel saadaval olevast andmebaasist [21].



Joonis 13. Süüdimõistmisi 1000 elaniku kohta aastatel 2006-2015.

Kuni 40 süüdimõistmist 1000 elaniku kohta oli vaid Saare, Põlva ja Võru maakonnas. 41 kuni 50 süüdimõistmist 1000 elaniku kohta oli Pärnu, Rapla, Jõgeva ja Tartu maakonnas, 50 kuni 60 süüdimõistmist Lääne, Harju, Viljandi ja Järva maakonnas ning üle 60 süüdimõistmise 1000 elaniku kohta esines Hiiu, Ida-Viru, Valga ja Lääne-Viru maakonnas.

Süüdimõistmisi on võimalik uurida täpse karistusseadustiku paragrahvi lõikes. Järgmisel joonisel [joonis 14] on näha mitu inimest 1000-st mõisteti süüdi joobes juhtimise eest, mis vastab karistusseadustiku paragrahvile 424.



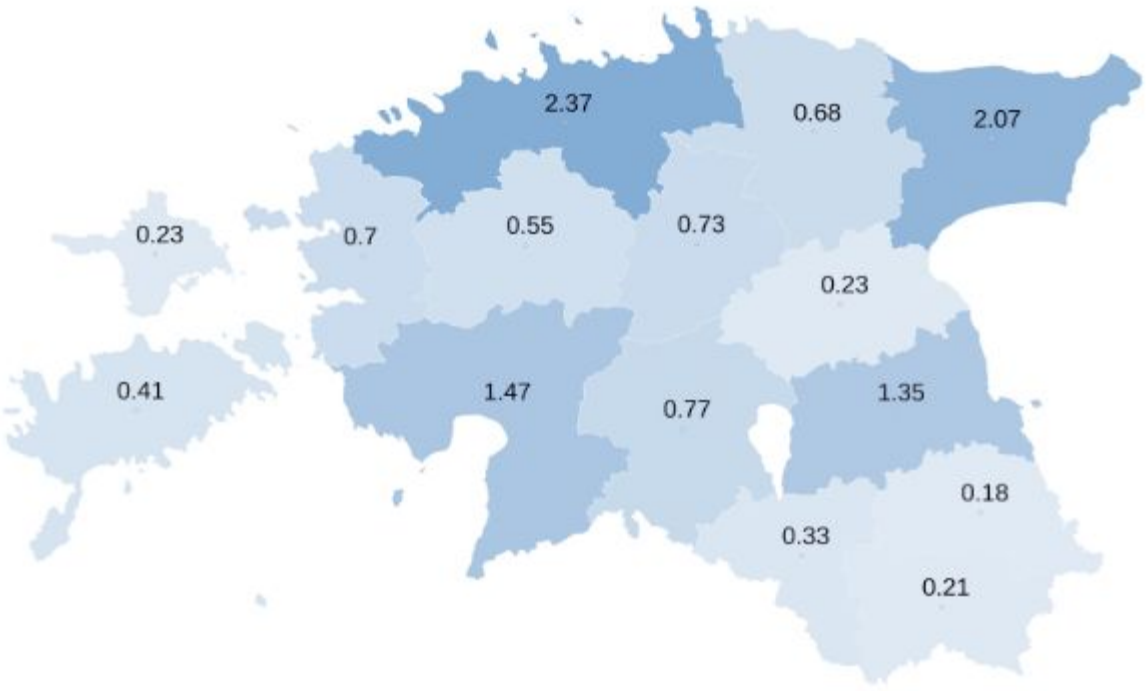
Joonis 14. Süüdimõistmisi KarS §424 (joobes juhtimise) põhjal 1000 elaniku kohta aastatel 2006-2015.

Nagu joonisel on välja toodud, on Harju maakonnas 1000 elaniku kohta joobes juhtimise eest süüdimõistmisi kõige vähem. Samuti on see näitaja madal ka Hiiu maakonnas. Eelmiselt jooniselt oli aga näha, et Harju maakonnas oli süüdimõistmisi 1000 inimese kohta rohkem kui paljudes teistes maakondades ning Hiiu maakond oli süüdimõistmiste rohkuse poolest neljas Eesti maakond. Seega joobes juhtide osakaal võrreldes teiste kurjategijatega on Hiiu ja Harju maakondades väiksem kui mujal.

Maakonnad, milles joobes juhtimise eest süüdimõistmisi 1000 elaniku kohta oli 14-18, olid Pärnu, Põlva, Ida-Viru ja Saare maakond. Eelmiselt jooniselt on näha, et Saare maakond oli väikseima kuritegevusega maakond, kuid tuleb välja et umbes pooled süüdimõistetutest olid joobes autojuhid. Samuti oli kuritegevus võrdlemisi madal ka Põlva maakonnas ning jällegi on näha, et suur osa sellest oli seoses joobes juhtimisega. Erinevalt Põlva ja Saare maakonnast, oli joobes juhtide osakaal Pärnu ja Ida-Viru maakonnas tunduvalt väiksem.

Ülejäänud maakondades oli joobes juhtimiste arv märgatavalt kõrgem, alates umbes 18 inimesest kuni umbes 24 inimeseni 1000 elaniku kohta. Kõige ohtlikumate teedega maakonnaks on Valga maakond, kus on üle kahe korra rohkem joobes juhte kui Harju ja Hiiu maakonnas.

Sarnaselt on võimalik ka uurida narkootiliste ainetega seotud kuritegusid, mis vastavad karistusseadustiku paragrahvidele 183-190. Nende alla kuuluvad näiteks narkootiliste ainete ebaseaduslik käitlemine, alaealistele andmine, kasvatamine ja levitamine. Järgmisel joonisel [joonis 15] on välja toodud, mitu inimest 1000-st elanikust on süüdi mõistetud seoses narkootiliste ainetega.



Joonis 15. Narkosüütegusid 1000 elaniku kohta aastatel 2006-2015.

Narkootiliste ainete süütegusid 1000 inimese kohta on kõige rohkem Harju, Ida-Viru, Pärnu ja Tartu maakondades, mis on juhtumisi ka neli Eesti rahvarohkeimat maakonda. Nende maakondade seas on narkokuritegude kaal kahanevas järjekorras vastavalt maakonna rahvaarvule. Erandiks on Tartu maakond, mis on rahvaarvu poolest teine, kuid narkootiliste ainete süütegude poolest viimane.

Väiksema rahvaarvuga maakondades on narkokuritegude arv 1000 elaniku kohta madalam, mis võib tulla näiteks halvemast kättesaadavusest või vähemast kontrollimisest.

6. Edaspidine töö

Edaspidises töös on vaja läbi viia informatsiooni eraldamise tulemuste valideerimine. Senimaani on tulemusi kontrollitud vaid visuaalselt TEXTA tööriista abil. Näiteks süüdimõistmist väljendavaid lauseid tuvastava grammatika täiendamiseks tehti TEXTA's päring, mis tagastaks kõik dokumendid, millest grammatika hetkeseisuga süüdimõistmist ei tuvastanud. Tulemusena saadud kirjetest uuriti, mis põhjusel seda ei tuvastatud, ning vajadusel täiendati grammatikat.

Lõpetamata ülesandeks jäi karistusmäärade eraldamine. Selle eesmärgiks on karistuse mõistmist väljendavatest lausetest leida täpsed karistused ja nende suurus. Selleks võib olla näiteks:

- vangistus
- arest
- rahatrahv
- juhtimisõiguse äravõtmine
- ühiskondlikult kasuliku töö määramine.

Tulemusena on võimalik näiteks mistahes kuriteo põhjal võrrelda, milliseid karistusi erinevad kohtud määrasid või mis oli karistuse keskmine suurus.

Samuti on plaanis kannatanu kasuks raha välja mõistvatest lausetest täpsete summade eraldamine ning nende analüüsimine. Näiteks oleks võimalik iga karistusseadustiku paragrahvi kohta leida keskmine summa, mis kohtualuselt välja mõisteti. Samuti võib huvi pakkuda välja mõistetud raha kogusumma näiteks maakondade või kohtute lõikes.

7. Kokkuvõte

Tekstikaeve ning selle alamülesanne informatsiooni eraldamine on tänapäeval väga relevantsed teemad. Käesolevas töös rakendati mustripõhist informatsiooni eraldamist Eesti kohtute lahenditele kasutades eesti loomuliku keele töötluse teegi EstNLTK grammatikate koostamise moodulit. Kuna vastav moodul oli eelnevalt vähe testitud, leiti töö käigus selle positiivseid kui ka negatiivseid külgi, mis oli üheks töö eesmärgiks. Peamiseks eesmärgiks oli moodustada andmebaas, mille toel oleks võimalik arendada rakendus, mis võimaldaks juristidel efektiivselt otsida enda jaoks huvipakkuvaid kohtulahendeid ning läbi viia praktilisi analüüse.

Selle saavutamiseks võeti kõigepealt Riigi Teataja veebilehelt kättesaadavad kohtulahendid ning konverteeriti need pdf-formaadist txt-formaati. Tulemusena saadud dokumendid puhastati regulaaravaldiste abil ning seejärel eemaldati mittevaliidsed lahendid. Viimaseks eeltöötluse sammuks oli teksti lemmatiseerimine ehk sõnade algvormi viimine.

Järgmiseks sammuks oli informatsiooni eraldamise rakendamine kohtuotsustele. Kõigepealt jagati dokumentide tekstid segmentideks vastavalt nende sisule. See oli võimalik, kuna tihti eelnes mingi kindla sisuga lõigule vastav sektsioonipealkiri.

Pärast teksti segmenteerimist tuvastati iga kohtuotsuse teksti resolutsiooniosast huvipakkuvaid fakte, milleks olid kohtualuse süüdi või süütuks mõistmine, esitatud avalduste rahuldamine ja rahuldamata jätmine, eelnevalt tehtud otsuste muutmine, muutmata jätmine ja tühistamine, karistuse määramine ja tühistamine ning kannatanu jaoks raha välja nõudmine.

Tulemusena tekkis andmebaas, milles iga kohtuotsuse jaoks olid välja toodud selle segmenteeritud tekst ning tuvastatud faktid. Tuvastatud faktide ja segmentide arvust anti ülevaade ning lisaks toodi välja ka mõned näiteanalüüsid, mida on võimalik vastava andmebaasi põhjal läbi viia.

8. Kasutatud materjalid

[1] Andrew McAfee, Erik Brynjolfsson. (2012, oktoober) Harvard Business Review. [Internetiallikas]. http://www.rosebt.com/uploads/8/1/8/1/8181762/big_data_the_management_revolution.pdf

[2] Andreas Hotho, Andreas Nürnberger, Gerhard Paaß. (2005, mai) LDV Forum - GLDV Journal for Computational Linguistics and Language Technology. [Internetiallikas]. http://www.kde.cs.uni-kassel.de/hotho/pub/2005/hotho05TextMining.pdf

[3] EstNLTK (külastatud 17.05.2016). [Internetiallikas]. http://estnltk.github.io/estnltk/1.3/index.html

[4] W. G. Lehnert, M. H. Ringle, "Strategies for Natural Language Processing", *Taylor & Francis*, lk 3-5, aprill 2014.

[5] Charu C. Aggarwal, ChengXiang Zhai, "Mining Text Data", *Springer US*, lk 4-8, 2012.

[6] T. Poibeau, H. Saggion, J. Piskorski, R. Yangarber, "Multi-source, Multilingual Information Extraction and Summarization", *Springer Berlin Heidelberg*, lk 26-27, 2013.

[7] K. Brown, "Encyclopedia of Language & Linguistics, Second Edition", *Elsevier*, lk 667-671, 2006.

[8] C. Cardie. (1997, november) American Association for Artificial Intelligence. [Internetiallikas]. http://www.aaai.org/ojs/index.php/aimagazine/article/view/1322

[9] K. Müürisepp, "Eesti keele arvutigrammatika: süntaks", *Ülikooli kirjastus*, 2000.

[10] L. Hirschman, A. Yeh, C. Blaschke, A. Valencia. (2005, mai) BioMed Central. [Internetiallikas]. http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-6-S1-S1

[11] C. Blaschke, M. A. Andrade, C. Ouzounis, A. Valencia. (1999, august) American Association for Artificial Intelligence. [Internetiallikas]. https://www.researchgate.net/publication/221317167_Automatic_Extraction_of_Biological_Information_from_Scientific_Text_Protein-Protein_Interactions

[12] F. Reiss, S. Raghavan, R. Krishnamurthy, H. Zhu, S. Vaithyanathan. (2008) IEEE Computer Society Washington. [Internetiallikas]. https://www.computer.org/csdl/proceedings/icde/2008/1836/00/04497502-abs.html

[13] T. Petmanson. "Mustripõhine faktituletus eestikeelsetest tekstidest" (2012). [Internetiallikas]. http://dspace.ut.ee/handle/10062/32988

[14] A. Tkatšenko. "Nimega üksuste tuvastamine eestikeelsetes tekstides" (2010). [Internetiallikas]. http://dspace.ut.ee/handle/10062/32972

[15] Riigi Teataja (külastatud 17.05.2016). [Internetiallikas]. https://www.riigiteataja.ee/index.html

[16] PostgreSQL (külastatud 17.05.2016). [Internetiallikas]. http://www.postgresql.org/

[17] Elastic (külastatud 17.05.2016). [Internetiallikas]. https://www.elastic.co/

[18] Poppler (külastatud 17.05.2016). [Internetiallikas]. https://poppler.freedesktop.org/

[19] Psycopg2 (külastatud 17.05.2016). [Internetiallikas]. http://initd.org/psycopg/

[20] Terminology EXtraction and Text Analytics (TEXTA) toolkit (külastatud 17.05.2016). [Internetiallikas]. http://www.folklore.ee/dh/en/dhe_2015/sirel/

[21] Eesti Statistika (külastatud 17.05.2016). [Internetiallikas]. http://www.stat.ee/ee

Lisad

I. Segmenteerimisel kasutatud sektsioonipealkirjade näited

Resolutsiooni sektsioonipealkirjad:

"RESOLUTSIOON", "RESOLUTIIVOSA", "KOHTUOTSUSE RESOLUTSIOON", "RESULUTSIOON", "/RESOLUTSIOON/", "OTSUSE RESOLUTSIOON", "KOHTUOTSUSE RESOLUTIIVOSA"

Menetluse asjaolude sektsioonipealkirjad:

"ASJAOLUD", "ASJAOLUD JA MENETLUSE KÄIK", "MENETLUSE KÄIK", "ASJA KÄIK", "VAIDLUSTATUD OTSUSE SISU", "ASJAOLUD JA MENETLUSKÄIK", "ASJAOLUD JA ASJA KÄIK"

Asjaosaliste seisukoha sektsioonipealkirjad:

"ASJAOSALISTE PÕHJENDUSED", "ASJAOSALISTE PÕHJENDUSED RIIGIKOHTUS", "KAEBAJA SEISUKOHT", "KAEBAJATE SEISUKOHT", "MENETLUSOSALISTE ARVAMUSED", "MENETLUSOSALISTE SEISUKOHAD", "MENETLUSOSALISTE PÕHJENDUSED RIIGIKOHTUS"

Kohtu seisukoha sektsioonipealkirjad:

"TSIVIILKOLLEEGIUMI SEISUKOHT", "HALDUSKOHTU KOLLEEGIUMI SEISUKOHT", "PÕHISEADUSLIKKUSE JÄRELEVALVE KOLLEEGIUMI SEISUKOHT", "ÜLDKOGU SEISUKOHT", "KRIMINAALKOLLEEGIUMI SEISUKOHT", "KOLLEEGIUMI SEISUKOHT", "KOHTUOTSUSE PÕHJENDUSED"

Menetluskulude jaotuse sektsioonipealkirjad:

"MENETLUSKULUDE JAOTUS", "MENETLUSKULUD", "KOHTUKULUDE JAOTUS", "MENETLUSKULUDE JAOTUS JA KINDLAKSMÄÄRAMINE", "KOHTUKULUD", "KOHTU PÕHJENDUSED JA MENETLUSKULUDE JAOTUS", "KOHTU PÕHJENDUSED JA KOHTUKULUDE JAOTUS"

Edasikaebamise korra sektsioonipealkirjad:

"EDASIKAEBAMISE KORD", "EDASIKAEBE KORD", "EDASIKAEBAMISKORD", "EDASIKAEBAMISE KORD.", "EDASIKAEBAMISE KORD JA TÄHTAEG"

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Katrin Valdson**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   Mustripõhine informatsiooni eraldamine Eesti kohtulahenditest,
   mille juhendajad on Raul Sirel ja Aleksandr Tkatšenko,
   1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **19.05.2016**