TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
informaatika õppekava

Ain Uljas

Veebirakenduse loomine optimaalse eestikeelse klaviatuuripaigutuse leidmiseks

Magistritöö (30 EAP)

Juhendaja: Neeme Kahusk

Tartu 2014

Veebirakenduse loomine optimaalse eestikeelse klaviatuuripaigutuse leidmiseks

Lühikokkuvõte:

Antud töös vaatlesime lähemalt hea klaviatuuripaigutuse omadusi. Seletasime lahti ja rakendasime ühte põhjalikumat klaviatuuride analüüsimise ja tekitamise töövahendi Carpalx. Leidsime, et võimalusi oma seadeid määrata on väga palju, kuid ühtlasi võivad etteantud parameetrid osutuda väga subjektiivseks ning päriselule mitte vastata.

Koostasime inimeste trükkimisharjumuste uurimiseks veebitarkvara, mis palus vabatahtlikel trükkida etteantud tekste. Tekstid olid valitud nii, et kõigi tekstide peale oli iga tähepaari vaja sisestada vähemalt korra. Jäädvustasime kõiki klahvivajutusi ja nendega seotud ajahetki ja metaandmeid. Selle põhjal on edaspidi võimalik välja töötada uus ja parem klaviatuuripaigutus.

Võtmesõnad:

Klaviatuur, eesti keel, Dvorak

The creation of web application for the purpose of optimized Estonian keyboard layout

Abstract:.

In this paper we look closely some of the properties of a good keyboard layout. We reviewed a common keyboard layout creator and analyser Carpalx. We found that the number of options is great but it also allows the possibility of subjectivity.

We built a web site that asked volunteers to type in certain texts that contained all the symbol paris possible. All the keystrokes and timings were stored in a database. This allows to create a truly optimized keyboard layout based on real data.

Keywords:

Keyboard, Estonian language, Dvorak

Sisukord

1 Sissejuhatus

Praegu laialt kasutust leidev nn „QWERTY" klaviatuuripaigutus on ebaefektiivne. Ajalooliselt trükimasinate jaoks loodud sümbolite asetus sõrmistikul oli/on niisugune, mis pidi takistama sagedasti kasutatavate tähtede löögipeade omavahelist kinnikiilumist, mis avaldast tugevat negatiivset mõju trükkimise kiirusele[1] ja mugavusele. Tänapäeva klaviatuuridel ei ole niisugused piirangud mõistagi enam vajalikud. Praeguseks on nii mõnegi keele jaoks töötatud välja paremaid lahendusi, näiteks ameerika/inglise Dvorak[2] ja rootsi Sworak[1], milledel leidub täiesti arvestatav kasutajaskond, need on operatsioonisüsteemiga vaikimisi kaasas.

Dvorak klaviatuuripaigutuse mõtlesid 1936 aastal välja August Dvorak ja Dr. William Dealy. Aastate jooksul on nemad ning ANSI[2] seda täiendanud. Võrreldes QWERTY asetusega nõuab Dvorak trükkimisel vähem sõrmede liigutamist, vähendab valede klahvide tabamist ning suurendab trükkimiskiirust. Traditsioonilisel paigutusel oli nende hinnangul mõningad puuduseid. Paljud inglise keeles levinud tähekombinatsioonid nõudsid ebamugavaid sõrmeliigutusi või hüpet üle kodureas või tuli trükkida ühe käega. Ebaproportsionaalselt suur osa trükkimisest – 47% – toimub ülemisel real, 18% alumisel ja kõigest 34% keskmisel real[3]. Olgu öeldud, et eesti klaviatuuril on need näitajad vastavalt 37, 15 ja 34. Dvorak uuris tähtede ja tähekombinatsioonide sagedusi ning käte füsioloogiat. Tulemustest lähtudes koostas ta uue sõrmistiku, mis leevendaks QWERTY asetuse probleeme.

Eesti keele jaoks on on olemas Dvoraki kohandus[3], kus õ, ä, ö, ü, š ja ž on paigutatud vastavalt o, a, o, u, s ja z AltGr märgi alla. See ei ole mugavaks kirjutamiseks piisavalt hea lahendus, kuna esiteks on sealt neid sümboleid väga ebamugav vajutada ning teiseks on eesti keele tähtede ja tähepaaride/kolmikute sagedusjaotus hoopis teine. Töö eesmärgiks on uurida kas ja kuidas oleks usaldusväärsetele andmetel tuginedes võimalik luua eesti keelele optimiseeritud klaviatuuripaigutus, mis muuhulgas arvestaks järgmisi faktoreid:

- tähtede sagedus
- tähepaaride sagedus
- erinevate sõrmede tugevus klahvide vajutamisel

1 http://en.wikipedia.org/wiki/Svorak
2 http://et.wikipedia.org/wiki/ANSI
3 http://patorjk.com/keyboard-layout-analyzer/

- klaviatuuri erinevate veergude ja ridade vahel liikumise mugavus
- käte alteratsioon trükkimisel
- lisasümbolite (AltGr, š) ja levinumate klahvikombinatsioonide sisestamise mugavus

Pikaajalisem eesmärk on leida niisugune paigutus, mis minimeeriks sõrmede liikumise pingutust arvestades eelpooltoodud kitsendusi. Oma töös kasutan programmeerimiskeeli Python[4] ja PHP[5], samuti vahendeid statistikast ja andmekaevest, sealhulgas andmete visualiseerimist seaduspärasuste avastamiseks. Peamise osa tööst moodustab puhtakujuline programmeerimine, mille käigus töötlen suurte tekstikorpuste toorandmed ümber tavaliseks tekstiks ning tekitan tähepaaride sagedustabelid. Püüan selgust tuua küsimuses, kas tähepaaride vajutamiseks kuluvate keskmiste aegade ning klahvide asetusest tulenevatel mudeli vahel on võimalik tekitada usaldusväärset seost. Kirjeldan, kuidas toimib veebileht, mida kasutasin trükkimistulemuste kogumiseks ning kuidas ning mis alustel andmeid töötlesin.

4 http://www.python.org

5 http://www.php.net

2 Hea klaviatuuripaigutuse põhimõtted

Selles peatükis kirjeldan lähemalt peamise eesmärgi saavutamiseks vajalikke teadmisi ning metoodikat. Suurem osa siin esitatud ideid on saadud Carpalx autorilt [4]. Toon ära mõningad asjaolud, mis tugevalt mõjutavad ühe või teise klaviatuuripaigutuse nö headust.

Käte alteratsioon on trükkimisel oluline, kuna sama käe sõrmedel tippimine on üldiselt oluliselt aeglasem kui kahe käe koostöös toimuv tähtede ladumine. Samal ajal kui üks käsi klahvivajutust teostab, on teise käe sõrmed juba positsiooni sisse võtnud ja saab koheselt vajutada. Seepärast on oluline, et kummagi käe alla jäävad sümbolid oleksid valitud nii, et käte vahetamine oleks maksimaalne. Ideaalis annaks tähepaaride sagedustabelist moodustatud graaf meile täieliku kahealuselise graafi, mille kummagi aluse tippude arv oleks täpselt sama kui palju on ühe või teise käe all klaviatuuri sõrmiseid. Praktikas tähendab see aga seda, kaks sümbolit, mille omavahelise paari esinemissagedus st nende esinemiste arv testkorpuse tekstis on väga suur, tuleks paigutada erineva käe alla.

Sama sõrmega trükkimist peaks igal juhul vältima, kuna see on kõige aeglasem viis üldse tähti trükkida võrreldes näiteks erineva käe all või sama käe kuid erineva sõrme all oleva klahvi trükkimisel. Sümbolite paigutamisel tähendab see seda, et sama sõrme alla jäävate sümbolite omavahelisi tähepaare peab olema võimalikult vähe või üldse mitte.

Sõrmede rullumine on vastand käte alteratsioonile. Sellisel juhul toimub trükkimine sama käe sõrmedel, kusjuures eelistatum on rullumine väljastpoolt sisse („iu“ on parem kui „io“)

Väikese sõrme kasutust tuleb piirata, kuna tegu on käe kõige nõrgema ning seega ka kõige aeglasema sõrmega.

Klaviatuuri keskmise **kodurea kasutamist** tuleks suurendada ning alumise rea kasutust vähendada. Kodurida on juba mugavalt sõrmede all ning sellel olevate klahvide vajutamine on kiire ning vähest vaeva nõudev. Seevastu alumisele reale tasub paigutada vähem kasutatavaid sümboleid, kuna sinna pääseb raskemini kui ülemisele reale.

Sõrmede liikumise koguulatus on oluline heuristika mõõtmaks ja võrdlemaks erinevate paigutuste nö headust. See näitab, kui palju peavad sõrmed kokku erinevate klahvide vahel liikuma arvestades klahvi kaugust kodureast ning sümbolite järjestust. Suuremate tekstide korral võib see väärtus anda kokku mitukümmend kilomeetrit. Siiski ei anna see kogu pilti, kuna erinevate sõrmede ja käte koormust saab suures osas varieerida.

Ridadel piki ja külgsuunas liikumine on üks niisugustest printsiipidest, mille olulisuse üle on võimalik vaielda. Nii näiteks on sellises vähemlevinud paigutuses nagu Workman[5] selle autori eelistusest lähtuvalt esikohale seatud külgsuunalise liikumise vähendamine ehk teisisõnu, klahvide „g“ ja „h“ (QWERTY mõttes) kohal paiknevate sümbolite vajutusele eelistatakse hoopis klahve „w“ ja „e“, samas kui selline alternatiivpaigutuste hulgas suhteliselt populaarne klaviatuur nagu „Colemak“ paneb tihedamini kasutatavad tähed ritta ühtlaselt kogu põhireas. Antud töös teen ma eelduseks, et kiirem tippimine on ühtlasi ka kõige mugavam tippimine.

Kasutaja seisukohalt on väga oluline, et **levinumad klahvikombinatsioonid** (Ctrl+X,C,V,A,Z) oleksid ühe käega saavutatavad. Vastasel korral oleks võimaliku kasutajaskonna leidmine ülimalt keeruline.

Mõni alternatiivne inglisepärane klaviatuuripaigutus peab uue paigutuse loomisel esmatähtsaks hoopis **ümberõppimise lihtsust** ehk teisisõnu, tõsta klassikalise QWERTY klaviatuuril ümber võimalikult vähe sümboleid nii et sellest saadav kasu oleks maksimaalne. Sellist lähenemist kasutab näiteks Norman[6]. Antud töös ümberõppimise lihtsust ei käsitleta, kuna see oleks liiga piirav nõue.

Lisaks eelpool mainitud punktidele mõjutavad paigutuse subjektiivset „headust“ veel sellised välised tegurid nagu kasutusvaldkond ja otstarve. Näitena on hea tuua tüüpilist arvutitarkvara arendajata, rahvakeeli programmeerijat, kellel on tihti vaja kasutada programmeerimiskeelele spetsiifilisi erisümboleid, mida tavatekstides on oluliselt harvem ja seetõttu paigutataks suhteliselt halvasti kättesaadavatesse positsioonidesse. Teisisõnu, selleks et koostatav klaviatuuripaigutuse kasutamine oleks mingi spetsiifilist laadi trükkimist hõlmava ülesande täitmise juures üldse mõeldav (meie näitel koodi kirjutamine), tuleb koostada just selle tegevuse jaoks ettenähtud paigutus. Lihtne viis kuidas antud juhul probleemist üle saada, on vahetada omavahel numbrirea alumised ja ülemised sümbolid. Üldistel juhtudel võiks kõne alla tulla erialaspetsiifilise tekstikorpuse alusena võtmine.

3 Toorandmete teisendamine ja tulemuste visualiseerimine

Kirjeldan algandmete formaati ja nende teisendamisel ette tulnud komplikatsioone. Teksti analüüsil saadud sümbolipaaride sagedustabeli tegelikus sisust saab parema pildi, kui seda analüüsida, kasutades graafilisi töövahendeid.

Kõige olulisem ressurss antud töö tegemisel oli Tartu Ülikooli arvutilingvistika uurimisrühma koostatud tekstikorpus[7] millede tekste kasutasin tähtede ja tähepaaride leidmiseks. Sobiva korpuse valimisel oli kriteeriumiks, et diakriitilisi tähti nagu näiteks „š“ ja „ž“ sisaldavad sõnad oleksid korrektselt välja kirjutatud. Varasematel aastatel (1990 – 2000) oli ajalehtedel ja ajakirjadel, eriti võrguversioonis, kombeks teha asendusi kus nt sõna „garaaž“ oli kirjutatud „garaazh“ ja „tšekk“ kui „tshekk“. See ei ole aga soovitav kuna sagedustabeleid kalkuleeriv programm annaks sel juhul valesid tulemusi. Korrektseteks osutusid ajalehe Eesti Päevaleht aastakäikude (2003 – 2007) ning teadus ja ilukirjanduse korpused, milledest otsustasin kasutada ajalehe tekste.

Korpused olid antud TEI-märgendikeele failidena, igaüks eraldi ajalehe number. Sisuliselt oli tegu XML-failidega, kus põhilised märgendid <div0> - <div2> vastasid teatavale alamüksusele – ajalehe number, rubriik, artikkel – ja märgendid <p> ja <s> vastavalt lõik ning lause. Siin on üks lühike väljavõte sellest:

```
<p> <s> Veebruari alguses esilinastub Kanadas Eesti p&auml;ritolu
produtsendi Tina Pehme m&auml;ngufilm " Partition " , mis r&auml;&auml;gib
sikhi mehe ja noore musliminaise vahel t&auml;rganud keelatud armastusest .
</s> <s> " " Partition " on eepiline armastusfilm 1947. aasta Indiast ,
umbes nagu " Inglise patsient " , " iseloomustas filmi v&auml;liseestlane
Tina Pehme . </s> </p>
```

Suureks väljakutseks oli etteantud TEI-märgendikeelsete tekstide transleerimine tavaliseks tekstiks. Esiteks tuli tekstides ära asendada ASCII[6] tabelist väljapoole jäävate sümbolite olemid (*entity)*. Näiteks sümbol „š“ esines toorandmetes kui „š“ ja sõna „šašlõkk“ kirjutati välja kui „šašlõkk“ (õ - õ). See osutus tülikamaks kui esmapilgul tundus. Nimelt esines tekstides mõnikord eriti haruldasi sümboleid, mida korpuse kasutamise juhendites kirjas ei olnud ning tuli iseseisvalt hankida suuremad ja põhjalikumad teisendamise tabelid. Minu koostatud skriptis on loetletud üle kahe tuhande olemi. Praeguseks on üles pandud ka mõnede korpuste UTF-8 kodeeringus failid, kus teisendusi ei ole vaja teha,

6 http://en.wikipedia.org/wiki/ASCII

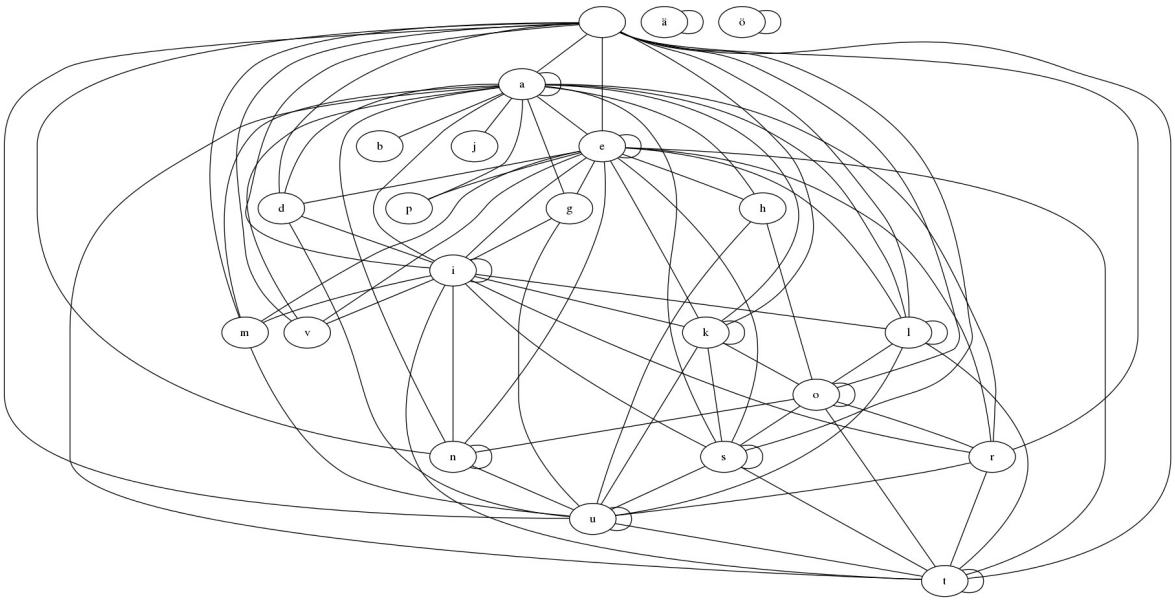
kuna tekst ise juba sisaldab õigeid sümboleid.

Toorandmetes esines mõningaid puuduseid. Märgendi <s> vahele peaks alati jääma üks lause, kuid praktikas see alati nii ei olnud. Esines olukordi, kus esimese lause kirjavahemärgid, eriti jutumärgid, olid liikunud järgmise lause märgendite vahele. Suure probleemi põhjustas ka asjaolu, et kirjavahemärgid olid alati mõlemalt poolt tühikutega eraldatud, seetõttu oli just üksteise sees paiknevate jutumärkide äratundmine ning sobiv paigutamine väga keeruline ülesanne. Kasutasin selleks regulaaravaldist, mis määrava osa jutumärkide asetusest suutis tuvastada õigesti. Kuigi XML-i töötlemiseks on regulaaravaldised ebasoovitavad, siis antud juhul piisas sellest täiesti – tuvastatud jutumärkide paigutamine ebaõnnestus vähem kui 1% juhtudest, mille tingis pigem ebatavaline lausekonstruktsioon ja väär lausestamine. Selline täpsus oli piisav. Mõnes korpuses oli alustav ja lõpetav jutumärk erineva sümboliga (olemiga) ja seetõttu oli lihtne aru saada, kummal pool asuv tühik on üleliigne – nii oleks võinud olla kõikjal. Teiste kirjavahemärkide paigutus (punkt, koma, sulud jt) oli üheselt määratavad.

Tulemuse maksimaalse usaldusväärsuse huvides tekitasin võimaluse protsessitavatest failidest konkreetseid tekstiblokke välja jätta. Pidasin artikli autori nime ning sissejuhatavat signatuuri ebavajalikuks, kuna tegu on korduva tekstiga. Samuti filtreerisin välja esinevad veebiaadressid (http://..), kuna need üldjuhul kopeeritakse, mitte ei trükita käsitsi. Nii saadud teksti salvestasin tekstifaili (UTF-8 kodeeringus). Pealkirjad, lõigud ning artiklid said omavahel eraldatud reavahetusega, artiklil kaks järjestikust. Seda faili kasutas sisendina järgmine skript, mis tekitas failid tähepaaride sagedustabeliga. Igas reas on tähe/sümboli paari esimene, veerus teine ning nende ristumiskohas selle tähepaari esinemise kordade arv. Võtsin arvesse nii väike- kui ka suurtähti, levinumaid kirjavahemärke, numbreid ja klaviatuuril kättesaadavaid erisümboleid, samuti tühiku ja reavahetuse. Erisümbolite osakaal üldarvestuses kujunes oodatust palju väiksemaks, suurem osa lahtreid tuli täita nulliga, vaid mõni üksik sümbolipaar sattus saja lähedale samas kui levinumatel tähepaaridel oli vastav näitaja mitusada tuhat ja rohkem.

Omandatud sagedustabelist ülevaate saamiseks püüdisin seda lihtsate vahenditega visualiseerida. Kasutasin *graphviz* paketti, mis nagu nimigi ütleb võimaldab joonistada graafe. Andes ette tipud ja servad ning nende kaalud paigutab tarkvara need nii, et suurema kaaluga servade pikkus on väiksem, teiste sõnadega, üksteisega tugevamalt seotud tipud on üksteisele lähemal. Kuna olenevalt sümbolitest võib tähepaaride arv kõikuda väga suures ulatuses ja joonis võib seetõttu tulla ebamõistlikult suur, kasutasin kaalu omistamisel logaritmimist. Lisaks olin sunnitud kasutama teatud fikseeritud piirmäära, millest allapoole jäävaid servi ei

joonistatud kuna vastasel korral tulnuks pilt liiga kirju. Kahjuks ei ole ka sel juhul tarkvaraline tippude paigutus eriti arusaadav. Võimalik et ilusama pildi saavutamiseks tuleb ette anda täpsemad seaded. Piirmäära varieerimisel saab mõningase hägusa ettekujutuse, mis tähepaarid on kõige levinumad ja mida võiks seetõttu kaaluda paigutada erineva käe alla, kuid midagi täpsemat selle põhjal öelda ei olnud võimalik. Ühte sellist näeb Joonisel 1. Katsetasin erinevaid joonestusmootoreid (‘neato’ , ‘dot’, ‘twopi’ ja’circo’), kuid üksi neis ei andnud soovitud tulemust.



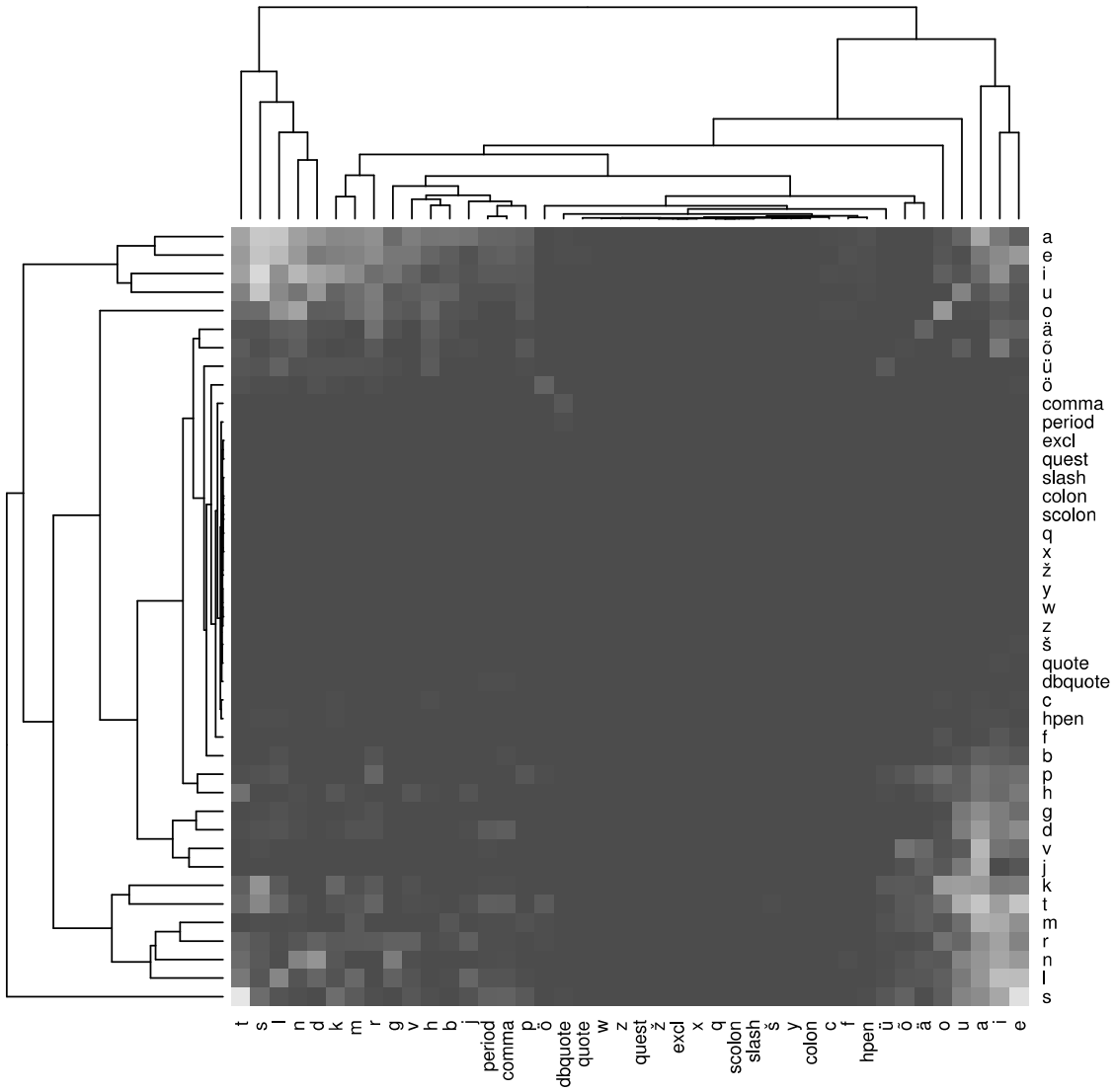
Joonistus 1: Graphviz tarkvara väljund tähepaaride sagedustabelile

Veel üks meetod, mida proovisin kasutada visualiseerimiseks, on serialiseerimine. Sisuliselt on tegemist admetabeli ridade ja veergude niisuguse ümberpaigutamisega, mis ideeliselt toob nähtavale omavahelised seaduspärasused ja seosed kui lähedase väärtusega lahtrid on paigutunud üksteise kõrvale. Suurepäraseks abivahendiks oli selle juures vabavaralise statistikapaketi R[7] moodul *seriation*[8]. Mõistuspärase pildi suutsid mulle anda järgmised algoritmid/meetodid: ARSA, Chen, MDS, HC, GW, OLO ja PCA. HC – hierarhilise klasterdamise – meetodi näidet võib näha Joonisel 2. Heledam ruut viitab sellele, et antud tähepaar esines tekstis võrreldes teistega oluliselt rohkem.

Antud jooniselt on nüüd võimalik juba üht-teist välja lugeda. Selgub, et täishäälikud a, e, i, u, o oleks arukas paigutada ühe ning kaashäälikud s, l, n, d, k, v, p, n, r, b ja m teise käe alla. Seda esiteks seetõttu et ühe ja sama grupi siseselt on tähepaaride arv suhteliselt väike samas kui erinevast grupist sümbolite paaride arv on märkimisväärselt suurem. Väikene probleem on

7 http://www.r-project.org/

tähega „t“, millel on suur arv paare nii ühest kui ka teisest grupist. Selle ning ka ülejäänud sümbolite asukohta tuleks kaaluda mõlema käe all.



Joonistus 2: hmap() väljund

4 Carpalx mudel

Selles peatükis kirjeldan ma lähemalt Carpalx'i autori poolt välja mõeldud mudelit paigutusel trükkimise jõupingutuse (v kaalu v hinde) leidmiseks. Seda mudelit püüdsin ka mina oma töös algselt kasutada, kuid siis leidsin, et konfiguratsiooniparameetrite väär seadistamine võib anda reaalsusele mittevastavaid tulemusi. Kahjuks ei ole 100% õiget seadistust olemas, kuna väga raske on numbrites hinnata, kui palju parem või halvem üks klaviatuuriridа teisest on või siis näiteks kui mitu korda on nimetissõrme kasutamine parem (kiirem? mugavam?) kui väikese sõrme.

Ühe tähepaari kaal e_i koosneb kolmest komponendist: baas (*b,* kaugus põhireast), trahv (*p,* ebasoodus asukoht) ja rada (*s*). Kõik kolm komponenti korrutatakse läbi osakordajatega (k_b, k_p *ja* k_s) ning liidetakse:

$$e_i = k_b b_i + k_p p_i + k_s s_i \tag{1}$$

Kogupingutus võrdub kõigi tähepaaride pingutuse summa jagatuna nende arvuga:

$$E = \frac{1}{N} \sum_i e_i \tag{2}$$

Baas- ning trahvkomponendid saadakse järgmiste valemitega

$$b_i = k_1 b_{i1} \left(1 + k_2 b_{i2}\right) \tag{3}$$

$$p_i = k_1 p_{i1} \left(1 + k_2 p_{i2}\right) \tag{4}$$

kus k_1 *ja* k_2 on esimese ja teise sümboli interaktsioonikordajad (alati mittenegatiivsed). b_{ij} märgib kaugust vastava sõrme puhkeasendist põhireal millimeetrites. Juhul kui $k_2 = 0$ siis tuleneb tähepaari kaal üksnes esimese sümboli kaalust – esimene ja teine sümbol ei mõjuta üksteist üldse. Niipea kui $k_2 > 0$ annab teise sümboli kaal proportsioonis esimese sümboli kaalule hinnet kõrgemaks. Niisugune valem simuleerib Carpalx'i autori nähemuses sõrmede väsimist, näiteks on ühendi „zx“ koos trükkimine vaevalisem kui nende tähtede eraldi sisse löömine. Trahv arvutatakse järgmise valemiga:

$$p_{ij} = w_0 + w_{käsi} P_{käsi_j} + w_{rida} P_{rida_j} + w_{sõrm} P_{sõrm_j} + \left[w_{positsioon} P_{positsioon_j} \right] \tag{5}$$

kus *w* on kordaja erinevate osatrahvide suuruse hindamiseks. Trahvi mõte on modelleerida suuremat pingutust vajavate vajutuste kaalu juhul kui sümbol asub raskemini ligipääsetaval real või tuleb sooritada nõrgema sõrmega. Lisaks on võimalik trahvida „vale käe“ kasutamise

eest, kuna parem käsi peaks teoreetiliselt olema tugevam ja seetõttu pisut eelistatum, ja omistada vaikimisi trahvi. Üks võimalik konfiguratsioon on toodud Tabelis 1. Võib juhtuda, et kui trahvipaari arvutamisel on esimeseks sõrmiseks klahv, mille individuaalne trahvikomponendi väärtus on null, siis on valemist tulenevalt kogu paari trahviks null. Kui selline olukord ei ole soovitav, siis võimalik seadistada vaikimisi trahv $w_0 > 0$. Täiendasin trahvikomponendi arvutamise valemit ebasoovitava positsiooniga sümbolite (AltGr) arvesse võtmiseks.

Käsi	$P_{käsi}$	***Rida***	P_{rida}	***Sõrm***	$P_{sõrm}$
Vasak	0.1	Numbririda	1,5	Nimetissõrm	0
Parem	0	Ülemine rida	0,5	Keskmine sõrm	0
		Keskmine rida	0	Sõrmuse sõrm	0,5
		Alumine rida	1	Väike sõrm	1
				Pöial	0

Tabel 1: Trahvikomponendi parameetrid

Kõige mitmetahulisem ja vaieldavam komponent on liikumisraja pingutuse arvutamine. Üldine mõte, millega kõik enamasti nõustuvad, on see, et erinevad tähepaaride/kolmikute käigud on erineva raskusega. Võtame näitena „fjo“ ja „jfy“. Esimene kasutab nõrka neljandat sõrme ja teine nõuab sirutust kaugel asetseva klahvi järgi. Olenevalt kordajatest võib nende kolmikute kaal tulla võrdne. Siiski, „fjo“ on lihtsam trükkida kui „jfy“. Seetõttu ongi mõistlik mudelisse lisada teatav funktsioon, mis aitaks hinnata eri trükkimiskäikude mugavust. Sellise funktsiooni koostamise on Carpalxi autor juba ära teinud, mille juures ta ka tunnistab, et tegu on väga subjektiivse tulemusega. Kasutasin mõnevõrra lihtsustatud versiooni, kuna käsitlen üksnes tähepaare, mitte kolmikuid. Valem on niisugune:

$$s_i = \sum_{j=käsi,rida,sõrm} f_j p_j \qquad (6)$$

kus p_j on väärtus, mis iseloomustab vastavalt käe, klaviatuurirea ja sõrme liikuvuse mustrit antud tähepaaris/kolmikus. Näiteks juhul kui trükkima peab vaid ühe käega, on $p_{käsi}$ väärtuseks kaks, mõlema käe kasutamisel üks. Kasutatava rea ja sõrme kontekstis saab tähekolmikus kokku lugeda seitse erinevat laadi liikumist, tähepaaris viis. f_i viitab tavapäraselt iga teatud komponendi eelseadistatud kaalule.

Carpalx mudelit iseloomustab mitmesuguste parameetrite ja kordajate rohkus. Mudeli

väljamõtleja ise kasutas trahvi ($w_{käsi}$, w_{rida} ja $w_{sõrm}$) ning komponentide osakordajatena (k_1, k_2) arve, mis andsid QWERTY klaviatuuril kogupingutuse suhteks 1:1:1. Interaktsioonikordaja valiti nii et $k_1,k_2=k_3=0$: $k_1,k_2,k_3=0$: k_1,k_2,k_3 oli 60:30:10, kus k_1 väärtuseks oli seatud 1. Teisisõnu, teise ja esimese sümboli koostoime andis 30%, ning kolmanda ja eelmiste sümbolite koostoime 10% protsenti selle tähekolmiku pingutusest. Kui need parameetrid paika said, valiti osakordajad nii, et igaühe kogupingutus eraldi andis väärtuse 1 ning kõik koos väärtuse kolm.

Paigutuse tekitamiseks kasutas ta praeguseks ajaks juba küllaltki tuntud Simuleeritud anniilimise (*Simulated annealing*[8]) meetodit, kus ühes sammus sooritati üks või mitu klahvide vahetust. Vahetuste vähim ning suurim kordade arv oli konfigureeritav, tegelik vahetuste hulk selgus (pseudo)juhuslikult sellest vahemikust.

Sooviks oli antud mudeli rakendamine eestikeelse teksti ning eestipärase klaviatuuriga. Kahjuks ei olnud see võimalik kuna AltGr klahvi all paiknevate sümbolitega ei olnud mudel arvestanud ning programmeerimiskeeles Perl[9], milles Carparlx on kirjutatud, ei ole käesoleva töö autor nii vilunud, et vastavad muudatused ise sisse suutnuks viia. Lahenduseks oli omaenese programmi kirjutamine.

Abivahendina kasutasin Pythoni pakette numpy[10] ja scipy,[11] kusjuures viimane on esimese laiendus ja sisaldab esimest. Lähenesin ülesandele nii, et kui omavahel korrutada tähepaaride sagedustabel ja klahvipaaride hinnete (permuteeritud) tabel ja see summeerida, siis tulemuseks ongi selle paigutuse koguhinne. See lähenemine osutus magnituudides tõhusamaks kui Carpalx. Suuri lootusi veelgi programmi kiiruse tõstmisel sai pandud asjaolule, et klahvide vahetamisel saab uue paigutuse hinde välja arvutada vaid vana hinnet ning paigutatavaid klahve teades. Siis ei pea sooritama korrutustehet kahe suure tabeli vahel. Teooria oli hea, kuid praktikas võttis indekseerimineära olulise osa jõudlusest. Näitlikustamise huvides toon siinkohal ära kõnealuste funktsioonide koodi.

8 http://en.wikipedia.org/wiki/Simulated_annealing

9 http://www.perl.org/

10 http://www.numpy.org/

11 http://www.scipy.org/

```
def swap_effort32(keyIndex1, keyIndex2, oldEffort):
        """Swaps two key positions on the keyboard and returns the new effort"""

        scrambledKeysMatrix = keysMatrix[permutationList, :][:, permutationList]
        rowdiff1 = scrambledKeysMatrix[keyIndex1, : ] - scrambledKeysMatrix[keyIndex2, : ]
        rowdiff1[keyIndex1]      =      scrambledKeysMatrix[keyIndex1,      keyIndex1]      -
scrambledKeysMatrix[keyIndex2, keyIndex2]
        rowdiff1[keyIndex2]      =      scrambledKeysMatrix[keyIndex1,      keyIndex2]      -
scrambledKeysMatrix[keyIndex2, keyIndex1]
        rowdiff2 = (-1) * rowdiff1
        rowdiff2[[keyIndex1, keyIndex2]] = rowdiff2[[keyIndex2, keyIndex1]]
        coldiff   =   sp.subtract(scrambledKeysMatrix[:,   keyIndex1],   scrambledKeysMatrix[:,
keyIndex2])
        coldiff[keyIndex1], coldiff[keyIndex2] = 0, 0
        return oldEffort - sp.sum(
                  rowdiff1 * statsMatrix[keyIndex1, :]
                + rowdiff2 * statsMatrix[keyIndex2, :]
                + coldiff * statsMatrix[:, keyIndex1]
                - coldiff * statsMatrix[:, keyIndex2])

def swap_effort1(keyIndex1, keyIndex2, oldEffort):
        """Swaps two key positions on the keyboard in place and returns the new effort"""

        keysMatrix[[keyIndex1, keyIndex2], :] = keysMatrix[[keyIndex2, keyIndex1], :]
        keysMatrix[:, [keyIndex1, keyIndex2]] = keysMatrix[:, [keyIndex2, keyIndex1]]
        return sp.sum(keysMatrix * statsMatrix)
```

Arvutuslikult kiirem kuid teostuslikult lihtsam oli siiski maatriksite korrutamisega (elemendi kaupa) leppida. Proovisin ka seda varianti, kus kõikvõimalike vajaminevate korrutiste tulemused on ühes suures, n^2 x n^2 maatriksis, kuid sellest saadav kiirusvõit ei olnud nii suur kui soovitud.

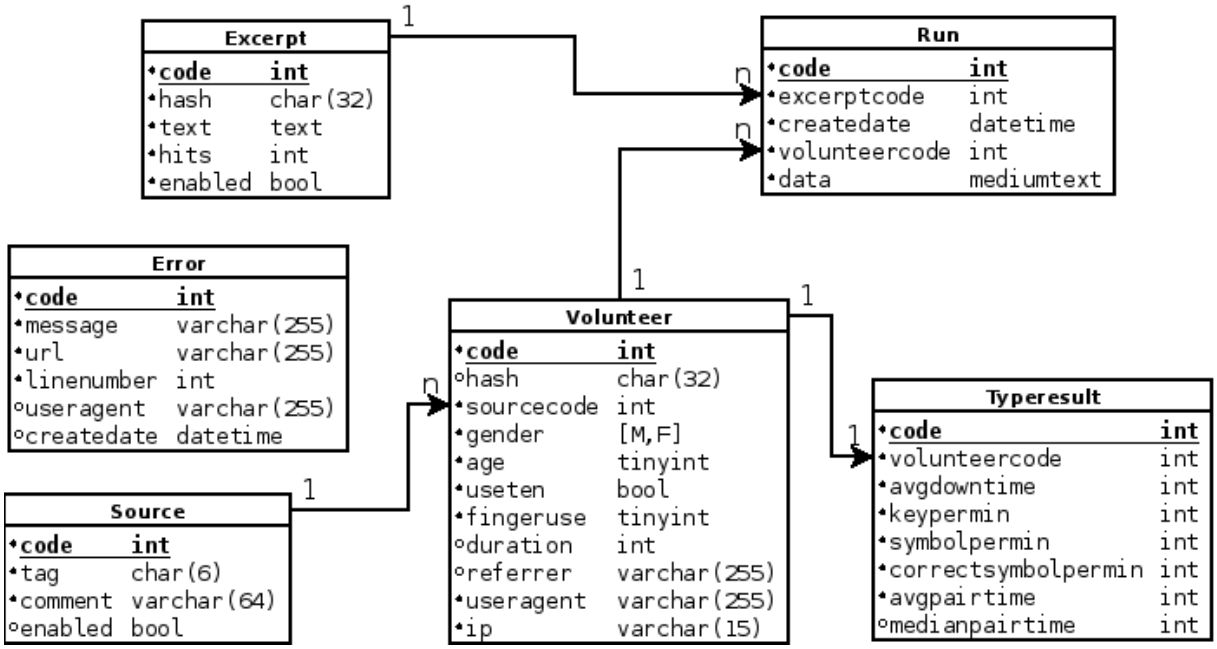
Katsetades erinevaid konfiguratsiooni parameetreid, kordajaid ja suhte asetusi tekkis minus kahtlus, nende „õiged“ ja „päris“ väärtused on tegelikult teadmata ja mistahes seadistused parameetrites on tegelikult puhas arvamismäng. Samuti ei tundunud olevat õige asi see, et kui näiteks sümbolipaari esimene täht juhtus olema „j“ (väga soodus asukoht), siis nii baas kui trahv komponendi pingutus mistahes teise sümboliga oli null ning tähepaari kaal sõltub täielikult liikumisraja osakaalust. See viitab sellele, et ka valem ise ei pruugi see õige olla. Leidsin, et vaja oleks reaalseid, mõõdetavaid ja tegelikkusele võimalikult lähedasi andmeid, et vältida valedele sisendandmetele tuginevate arvutuste tegemist, mis lõppkokkuvõttes ei pruugi päriselu üldsegi mitte kajastada.

5 Trükkimisandmete kogumine

Kirjutasin valmis interaktiivse veebilehe, mis peale üldstatistiliste küsimuste esitamist palub trükkida ühe fikseeritud sisu ja pikkusega kontroll/soojenduslause ning seejärel ühe või mitu hoolikalt valitud lõiku Eesti Päevalehe 2007. aasta kirjatükkidest. Jäädvustan klahvivajutustel toimuvaid ajahetki (klahv all, sümboli ilmumine ekraanile ja klahv üles) ning tulemused salvestan andmebaasi. Saadud andmetest püüan kokku panna võimalikult tõetruud mudelit klahvipaaride paiknemise ja nende vajutamiseks kuluva aja vahel. Järgnevalt kirjeldan kogu protsessi samm-sammult.

Struktuur

Serveripoolse loogika jaoks kasutasin PHP raamistikku Yii[12], mis muudab sisendi valideerimise ning andmebaasiga suhtluse väga mugavaks ja tõhusaks. Alloleva andmebaasimootorina kasutasin MySql'i. Tabelite andmevälju ning omavahelisi seoseid näeb Joonisel 3. Etteantavad täispikkuses tekstilõigud pikkusega 100 – 900 sümbolit paiknevad



Joonistus 3: Andmebaasi struktuur

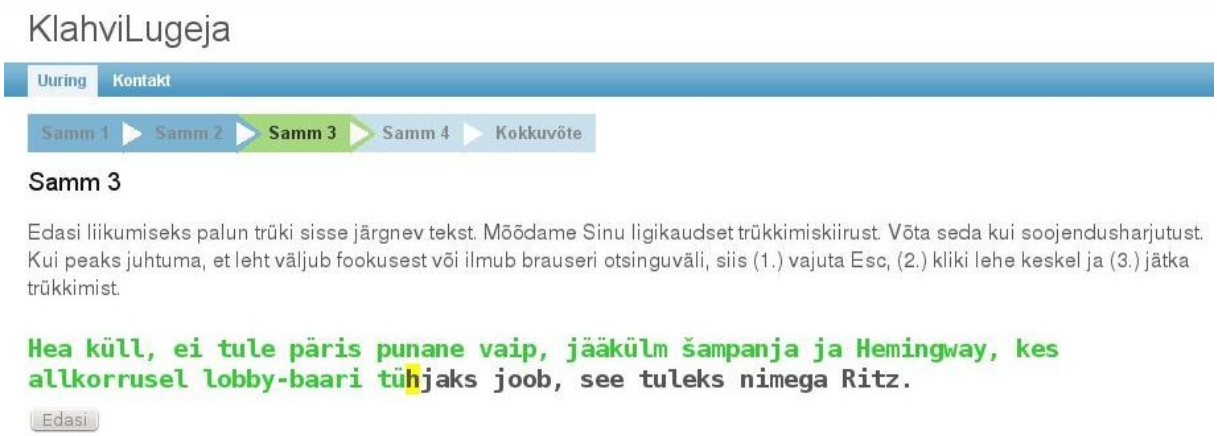
väljal *Excerpt.text.* Tabelis *Source* paiknevad süsteemi sisenemiseks vajalikud tunnuskoodid, mis antakse ette URL'i paramteetris *source.* Ilma selleta kuvatakse vastav sisestuskast, kuhu tuleb jätkamiseks kirjutada õige kood. Lehte ei kuvata juhul kui sama veebilehitsejaga on ettenähtud aja jooksul varem mõnda õiget koodi kasutatud. Jagasin vastavaid linke koos

12 http://www.yiiframework.com

selleks ettenähtud koodiga nii suuremates meililistides kui ka Facebook'is ning nii oli lihtne tuvastada, mis kanalit pidi keegi minu leheni jõudis. Tabelisse *Error* salvestatakse tegevuse käigus tekkivad võivad *JavaScripti* veateated koos kellaaja ning asukohaga koodis. Selle eesmärgiks oli võimaliku massiliselt ilmneva puuduse ilmnemisel koheselt midagi ette võtta, sest et veaolukorra registreerimisel saadetakse teavitussõnum ka eelseadistatud e-posti aadressile.

Ülesehitus

Esileheküljel avaneb kõigepealt sissejuhatav ja tutvustav tekst. Seejärel tuleb ette anda sugu ning vanus ja vastata küsimusele, kas inimene valdab kümne sõrmega trükkimist e pimekirja. Kui vastata „Ei“, ilmub veel üks küsimus, kus palutakse täpsustada, kui mitut sõrme (1 – 9) tegelikult kasutatakse. Need andmed salvestatakse hiljem *Volunteer* tabelisse. Kõikide nimetatud väljade väärtustega toimub range serveripoolne valideerimine. Edasi liikudes avaneb kolmas samm (Illustratsioon 1), kus palutakse trükkida üks lause, enne edasi liikuda ei



Illustratsioon 1: Kolmandal sammul tuleb trükkida üks konkreetne lause

saa. See lause on alati kõigile täpselt sama ning tema eesmärgiks on 1) valmistada inimene eelseisvaks suuremaks trükkimiseks ette ning 2) mõõta ära selleks kulunud aeg, et teada saada inimese umbkaudne klaviatuuril kirjutamise võimekus. Vastav number läheb kirja *Volunteer.duration* väljale. Valet klahvi vajutades ei juhtu midagi, st kursor sel juhul edasi ei liigu. Järgmisel sammul kuvatakse juba veidi pikem tekst ning palutakse see ilma peatusteta sisse trükkida, soovi korral saab tegevust mõne teise etteantud tekstiga korrata. Tekstid antakse ette juhuslikult, kuid eelisjärjekorras on need, mille *Excerpt.hits* väärtus on ülejäänutest väiksem. Järgnev koodilõik suudab seda loogikat võibolla paremini edasi anda:

```
$excerpt = ExcerptRecord::model()->find(array(
      'select'=> 'code, hash, text, rand() + hits as rand',
      'condition' => 'enabled = 1',
      'limit'=> 1,
      'order'=> 'rand',
));
```

Kõik tekstid (lõigud ajalehe artiklitest), mida ette anda võidakse, on hoolikalt välja valitud nii et kõigi tekstide peale, mida ise on võimalikult väikene arv, on iga tähepaari esinenud vähemalt korra. Selle saavutamine osutus ootamatult keeruliseks (hulga katte probleem)[13]. Valik tuli teha u kahesaja tuhande lõigu hulgast ning arvutamise käigus tekkis probleem arvuti RAM töömälu vähesusega. Proovisin erinevaid algoritme. Lõpuks osutus edukaks lähenemine, kus sisendfail loeti mällu osade kaupa ja leiti iga osa kate eraldi. Tulemused võeti ühte kokku ning rakendati sama algoritmi uuesti. Sisendfailiks oli seejuures mitte tavaline tekst lõikudega vaid bittide jadad, kus ühe biti väärtus vastas teatud sümbolipaari olemasolule või mitteolemasolule selles lõigus, mida antud bitivektor esindas. See formaat võimaldas mälu kokku hoida ning protsessida korraga suuremaid tükke. Tükkide suurusele seadis piirid ka võimalike sümbolite hulk, mille kõikvõimalikke paare arvesse võtma pidi. Vastavalt muutus ka tööks kuluv aeg. Valitud tekstid ei sisalda sümboleid, mida pole klaviatuuril.

Kui inimene ei soovi rohkem trükkida ja liigub edasi kõige viimasele lehele(Illustratsioon 2),

Illustratsioon 2: Viimane tulemuste leht

kuvatakse talle keskmised trükkimiskiirused, viis kõige lühemat ja kauemat aega all hoitud sümbolit ning samuti viis kõige kiiremini ja aeglasemini trükitud sümbolipaari. Viimasele

13 http://en.wikipedia.org/wiki/Set_cover_problem

lehele liikumise hetkel salvestatakse andmed ühe transaktsiooniga baasi. Juhul kui selle käigus tekib viga, pannakse teele vastavasisuline sõnum eelseadistatud e-posti aadressile. Salvestamise juures on veel see nüanss, et kui kasutaja märkis, et ta oskab kümnesõrmesüsteemi, siis nende trükitud lõikude *Excerpt.hits* väärtust suurendatakse ühe võrra, muidu mitte. Statistilised väärtused arvutab inimese veebibrauser ise kohapeal, mis saadetakse eraldi päringuga serverisse järele. Mingit otsest eesmärki nende salvestamine ei täida, lihtsalt nende pealt on hea üldstatistilisi päringuid teha ilma et peaks klahvivajutuste järjendeid töötlema hakkama (vt järgmine lõik).

Iga klahvivajutusega, mis tekitab ekraanile sümboli (sinna alla käivad muuseas ka tabulaatori ning tagasivõtu klahv), talletuvad järgmised andmed:

- vajutamise aeg
- lahtilaskmise aeg
- klahvi kood
- sümboli ekraanile ilmumise aeg
- trükitud sümboli kood
- trükitud sümbol
- tõeväärtused, kas samal ajal hoitakse all ka Shift, Alt või Ctrl klahvi
- tõeväärtus, kas vajutatud sümbol oli korrektne
- tõeväärtus, kas vajutati vasakpoolset Shift klahvi või mitte (usaldusväärne ainult Internet Explorer veebibrauseriga, teistega alati „väär“)
- samade andmeväljadega sümbolite loend, mida hoiti all hetkel kui antud sümbol ekraanile ilmus

Juhul kui vajutatakse mõnda meta-klahvi (Alt, Ctrl, Win/Cmd, Shift jt), *keypress* sündmus JavaScript'is ei rakendu ja talletuvad vaid eespool toodud nimekirja kolm esimest andmevälja. Andmebaasi (*Run.data*) salvestatakse kõik need järjestikused vajutused objektide listina JSON (– *JavaScript Object Notation*) formaadis.

Seda töötles edasi juba eraldiseiseisev Python'i skript, mis tekitas uue faili, mille ridade ja veergude ristumiskohta sai pandud keskmine antud sümbolipaari kirjutamiseks kuluv aeg,

teatud juhtudel aga mitte. Selleks et mingi tähepaari ajavõtt jõuaks faili, pidi vähemalt neli inimest seda kirjutanud olema nii et keskmine ja mediaan erinesid kuni 10% võrra või siis seda oli kirjutatud kümme või rohkem inimest. Muul juhul jäi lahter tühjaks. Selleks et nende nelja või kümne hulka üldse jõuda, ei tohtinud vastav number erineda mediaanist rohkem kui kaks korda. Niisugune mediaaniga piiritlemine filtreeris tõhusalt välja erindid. On võimalik, et seda tähepaari trükkis sama isik mitu korda. Arvesse lähevad vaid need ajad, mis jäävad samamoodi mediaanist kaks korda ühele või teisele poole. Nendest väärtustest võetakse mediaan ja korrutatakse läbi koefitsiendiga. Koefitsient saadakse nii, et globaalne keskmine tähepaari mediaan üle kõigi trükkinud (kümnesõrmeliste) inimeste jagatakse läbi selle inimese kõigi klahvivajutuste vaheliste aegade mediaaniga. Koefitsiendiga korrutamine tagab selle, et erinevate trükkimiskiirusega inimeste klahvivajutuseks kuluvad ajad on omavahel võrreldavad. Mediaani olen arvutuses kasutanud keskmise asemel kuna see statisik on erindite suhtes vähem tundlikum, eriti oluline on see väga väikeste hulkade puhul.

Veebilehel oli võimalus ka saata autorile tagasisidet. Seda võimalust kasutasid kümmekond inimest.

Tehtud vead.

Tegin ka mõningaid vigu. Esimesel korral, kui antud süsteem publikumi ette tõin, ei lubanud süsteem eespool enam parandusi teha, kui inimene oli juba mõnda õiget sümbolit vajutanud. Mõte oli selles, et kuna tahtsin koguda vaid klahvide vajutamise aegu, siis ei olnud põhimõtteliselt vahet, kas vajutatav klahv on just see mis ta olema peab või mitte ning saan selle arvesse ikkagi võtta. Asjaolud, mida ma aga arvesse ei võtnud ja mis selgusid alles siis kui ma esmakordselt kogutud andmeid töötlema hakkasin, on järgmised: esiteks, juhul kui trükkija tegi vea kohaga, kus on koos mõni harvaesinev sümbolipaar, siis läheb selle ajavõtu võimalus minu jaoks kaotsi, kuna parandada ei saa. Teiseks, olukorras kus inimene on mingil põhjusel valesti trükkinud, ei saa saadavaid mõõtmisi päris tõepähe võtta, kuna tulemus sündis tõenäoliselt ootamatul ning mitteettenähtud viisil. Teisisõnu, tegemist on erindiga, mis statistiliselt ei sobitu võimalikku mudelisse. Ma ei tahtund, et trükkimisel tehtavate vigade parandamisega rikutaks ära üldine tempo ning vool. Võimalik, et saavutasin risti vastupidise tulemuse, kuna hoolimata selgitavast tekstis oli parandamise võimatus inimestele üllatav ning mitteharjumuspärane. Peale muudatuste sisseviimist palusin mõnel pool ülesannet korrata („teine laine“).

Olgugi et keskmiselt trükkis iga inimene umbes kolm lõiku, arvesse läksid vaid need isikud, kes kümnesõrme trükkimist valdasid ning neid tuli minu veebilehele mitu korda vähem kui ma lootsin. Tegin selle vea, et algselt oli arvesse võetavate sümbolite hulgas ka numbrid, mis andis lõikude koguarvuks 588. Kuna mulle olulisi trükkimisi oli nn esimeses laines 1100, tähendab see seda, et kõigest 76 lõiku oli trükitud kolm, ülejäänud kaks korda. Eesmärgiks oli 3 – 5 korda kõiki tekste. Peale numbritest ja mõnedest sümbolitest loobumist langes tekstide koguarv 276 peale. Oleks ma kasutanud neid juba varem, ületanuks kõigi katvus kolme suurel enamusel isegi neljakordselt, koos nn teise laine 596 trükkimisega juba kuuekordselt.

Tulemused

Töötlemata kujul võib tulemusi näha Lisas 1. Nagu näha, siis ei ole arvud alati päris need, mis nad mudel järgi olla võiksid. Esineb palju vasturääkivusi ja ebakõlasid. Osutus, et tähepaaride vahelised mõõdetud ajad sõltuvad tugevasti selle tähepaari kui ka tähtede endi sagedusest. Logaritmiliselt tulemusi skaleerides ei ole saadud arvud siiski päris need mis võiks. Valemi ülesehitamine, mis tekitaks ridade, sõrmede ja klahvi kauste ja mõõdetud tulemuste vahel kindla seose nõuab omaette uurimist, kuid kindlasti on see võimalik. Salvestatud andmeid on võimalik edaspidi kasutada teistes uurimustes, mille eesmärgid ja kriteeriumid võivad olla oluliselt teised, kuid eelkõige loob see tugeva aluse reaalsetele andmetele tugineva optimaalse eesti keele klaviatuuripaigutuse väljatöötamiseks.

6 Kokkuvõte

Antud töös vaatlesime lähemalt hea klaviatuuripaigutuse omadusi. Seletasime lahti ühte põhjalikumat klaviatuuride analüüsimise ja tekitamise töövahendi Carpalx. Programmeerisime oluliselt kiirema ja tõhusama lahenduse. Leidsime, et võimalusi oma seadeid määrata on väga palju, kuid ühtlasi võivad etteantud parameetrid osutuda väga subjektiivseks ning päriselule mitte vastata.

Koostasime inimeste trükkimisharjumuste uurimiseks veebitarkvara, mis palus vabatahtlikel trükkida etteantud tekste. Tekstid olid valitud nii, et kõikgi tekstide peale oli iga tähepaari vaja sisestada vähemalt korra. Jäädvustasime kõiki klahvivajutusi ja nendega seotud ajahetki ja metaandmeid. Selle põhjal on edaspidi võimalik välja töötada uus ja parem klaviatuuripaigutus.

Viited

[1] Wikipedia QWERTY klaviatuur, http://en.wikipedia.org/wiki/QWERTY viimati külastatud: 05.2014

[2] Wikipedia Dvorak Simplified Keyboard, http://en.wikipedia.org/wiki/Dvorak_Simplified_Keyboard viimati külastatud: 05.2014

[3] Ilves, T. Täpiline ANSI Dvorak, http://dvorak.juhe.ee viimati külastatud: 27.11.2013

[4] Krzywinski, M. Carpalx keyboard layout optimizer, http://mkweb.bcgsc.ca/carpalx/ viimati külastatud: 20.11.2013

[5] Bucao, O.J. Workman Keyboard Layout, http://www.workmanlayout.com viimati külastatud: 05.2014

[6] Norman D. Norman Keyboard Layout, https://normanlayout.info viimati külastatud: 05.2014

[7] Research Group of Computational Linguistics, University of Tartu Tekstikorpuste arhiiv, http://www.cl.ut.ee/korpused/ viimati külastatud: 18.11.2013

[8] Hahsler, M., Buchta, C., Hornik, K. R Infrastructure for seriation, http://cran.r-project.org/web/packages/seriation/ viimati külastatud: 20.11.2013

Lisad

I. Tähepaaride vajutamiseks kulunud ajad

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	š	ž	z	t	u	v	w	õ	ä	ö	ü	x	y	,	.	-	'	sp
a	160	182	266	171	178	230	175	143	132	139	145	131	139	139	148	170	1053	163	142	736	900	397	174	160	200	343	373			252	339	236	203	224	247	695	122
b	126	195			163				141	173		165	272		160			219	200					146				191	335	298			221					147
c	150		160		170			159	161		141		202		135			287				671	244	145							246						224
d	111	396		167	168	242	249	447	122	168	172	150	169	177	147	347		225	220				267	128	242		380					207	179	197	692		126
e	142	222	286	205	156	249	195	158	128	192	155	133	143	135	174	173		137	177	709	996	390	168	194	213	230				398	391	227	175	208	250	749	126
f	138				170	155	235		122			142	182		125			228					260	142									274				142
g	116				122		176	159	114			156	146	190	146	337		212	221					120						140			198	271			130
h	114				112			130	139	193	183	229	233	216	163			235					125	160	179		211	205		206				274			206
i	143	188	230	132	142	238	153	208	141	224	187	190	174	157	180	217		140	119			524	147	172	169	677	343			314			259	266	349	769	137
j	107				119				160				220	223	168				236				190	168			198	197									
k	111	506			113	364		343	176		140	186	198	227	135	379		190	119				149	170	224		201	177	273	199		259	239	304	252		160
l	114	237		136	110	220	177	235	153	179	185	143	152	183	185	232		369	155				134	175	196		225	193	252	263		223	240	250	354		140
m	111	214			113	360			121			267	138	206	161	204			148					194			205	199	233	251		247	309	243	290		166
n	111	334	237	126	109	249	132	260	132	303	186	233		142	179	230		196	157			548	145	162	203		224	189		213		223	284	261	351		117
o	175	188	238	130	158	182	159	174	156	186	192	185	177	160	144	195		139	136	553		304	152	159	177	380					320	298	268	261			185
p	123				119			263	181		228	217	227		180	148		151	168				199	203			228	204	270	275			285	436	423		218
q	483																							236	723												
r	135	291	285	215	114	262	228	242	127	128	164	204	146	169	140	206		153	206			845	201	132	230		218	145	135	149		184	204	215	353		168
s	147		297		125	605	442	227	125	160	160	182	165	175	145	232		255	163			388	149	133	258		149	197	298	156		176	175	200	242	728	134
š					304				283		244				259								402	387													
ž	457				321				327						561													739		550							
z	226	322		320	278				206		269		395		160							166												548		414
t	117		355		115			145	124	160	159	160	164	192	132	305		198	189	518			153	119	288	425	143	135	153	170		177	203	206	292	823	130
u	143	174		128	142	182	160	171	135	194	208	188	216	193	329	221		135	121			356	139	140	177					261	341	312	289	265	692		144
v	115			286	160				125	224		176		228	145				223				278	145			129	149				189	223	285			173
w	180				217				147					142	132				370							182								151			273
õ		270		202	170			238	200	228	280	236	252	232		253		158	183				175	211	294		150										
ä		191		146	113		168	199	199		251	203	216	215	271	226		132	158				183		204			151									
ö		167		149	172		154		264		280	240	182			231		181	128				121		155				150				281				208
ü	192	298		133			157	231			223	214	213	209		273		189	161				168							149							
x	258								129														213														222
y	242	274		383	160							232		257	204				284				250	305									341		291	824	202
,																																					125
.		452	258		198								302	341	308			336																150	736		177
-	222			295		278	291	340	308		271	302	319	294	280	392		338	248				332		291												160
'									303					873					223																		
sp	181	315	277	360	175	322	321	243	201	166	178	193	174	187	152	207		258	187	611			202	237	206	366	296	247	366	277					198	590	

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina Ain Uljas (sünnikuupäev: 08.04.1985)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   Eesti keele optimaalse klaviatuuripaigutuse otsingul,
   mille juhendaja on Neeme Kahusk ,
   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, **26.05.2014**