Haaga-Helia
ammattikorkeakoulu Oy

Tietokanta journalistisena tekona: eduskunnan keskustelupöytäkirjat 1907–1999 hakutoiminnolla varustetussa tietokannassa

Juuso Koponen

Master-opinnäytetyö

Uudistuvan journalismin koulutusohjelma

2022

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä
30.4.2022

Tekijä(t)
Juuso Koponen

Suuntautuminen
Journalismi (YAMK)

Opinnäytetyön nimi

Tietokanta journalistisena tekona: eduskunnan keskustelupöytäkirjat 1907–1999 hakutoiminnolla varustetussa tietokannassa

Sivumäärä + liitesivumäärä
74 + 6

Tämän kehittämistyön tarkoituksena on tutkia tietokantaa journalistisena ilmaisumuotona ja datajournalistin työvälineenä.

Toteutin työssäni hakutoiminnolla varustetun tietokannan eduskunnan täysistunnoissa vuosina 1907–1999 käydyistä keskusteluista. Työ koostui skannattujen dokumenttien muuttamisesta koneluettavaan muotoon tekstintunnistuksella, aineiston analysoinnista luonnollisen kielen käsittelyn ja muiden menetelmien avulla, kansanedustajien biografiatietojen yhdistämisestä heidän puheenvuoroihinsa, näiden tietojen tallettamisesta tietokantaan ja selainpohjaisen hakukäyttöliittymän rakentamisesta.

Työni tarkoituksena oli tukia tietokantaa sekä journalismin työvälineenä että journalistisena ilmaisumuotona. Tarkoituksenani oli vastata seuraaviin kysymyksiin:

- Millaisia taitoja ja välineitä skannatuista asiakirjoista koostuvan aineiston siirtäminen tietokantamuotoon vaatii?
- Miten journalistit voisivat hyödyntää luonnollisen kielen käsittelyä ja tietokantoja omassa työssään?
- Kuinka hyvin tietokanta soveltuu journalistisena sisältönä julkaistavaksi?

Tavoitteenani oli oppia käyttämään luonnollisen kielen käsittelyn menetelmiä journalistisessa kontekstissa ja tuottamaan käyttökelpoinen tietokanta, sekä kirjata työn vaiheet ja vaikeudet niin, että kehittämistyötä voidaan käyttää apuna myöhemmissä samankaltaisissa projekteissa, jotta kehittämistyöstäni olisi mahdollisimman paljon apua muille, jotka ovat kiinnostuneita vastaavan tai samankaltaisen tietokannan toteuttamisesta.

Vaikka tietokannan rakentaminen osoittautui teknisesti vaativaksi, kehittämistyön lopputuloksena syntyi toimiva tietokanta käyttöliittymineen, joiden avulla oli mahdollista kerätä tietoa datajournalistisia artikkeleita varten.

Haaga-Helia
ammattikorkeakoulu Oy

Tiivistelmä
30.4.2022

Asiasanat
Datajournalismi, tietokantajournalismi, journalismi, tietokanta, luonnollisen kielen käsittely, eduskunta

Sisällys

1 Johdanto

Kehittämistyönäni olen toteuttanut tietokannan, joka sisältää kaikki eduskunnassa käydyt, keskustelupöytäkirjoihin merkityt keskustelut vuosilta 1907–1999 puhujan mukaan eriteltynä ja suomen kielen jäsentimen avulla perusmuotoistettuna ja analysoituna, sekä verkkopohjaisen käyttöliittymän, jonka avulla kannassa olevia tietoja voi hakea. Käyttöliittymä sijaitsee verkko-osoitteessa http://demo.koponen-hilden.fi/eduskunta-keskustelut/

Olen lisäksi hahmotellut tietokannan tietoja hyödyntäen kaksi erityyppistä datajournalistista juttukokonaisuutta. Niiden laatimisen tarkoituksena on ollut ennen kaikkea testata käytännössä kannan ja verkkokäyttöliittymän soveltuvuutta journalistisen tiedonhankinnan välineeksi.

Työni tarkoituksena on tukia tietokantaa sekä journalismin työvälineenä että journalistisena ilmaisumuotona. Pyrin työssäni vastaamaan seuraaviin kysymyksiin:

- Millaisia taitoja ja välineitä skannatuista asiakirjoista koostuvan aineiston siirtäminen tietokantamuotoon vaatii?
- Miten journalistit voivat hyödyntää luonnollisen kielen käsittelyä ja tietokantoja omassa työssään?
- Kuinka hyvin tietokanta soveltuu journalistisena sisältönä julkaistavaksi?

1.1 Työhön liittyvät sidosryhmät

Työllä ei ole toimeksiantajaa.

Toimin Haaga-Heliassa datajournalismin opettajana. Opetan lisäksi datan visualisointia ja analysointia useissa muissa suomalaisissa korkeakouluissa. Uskon, että kehittämistyössäni tehdyt havainnot hyödyttävät datajournalismin opetusta ja sen kehittämistä.

Vaikka projektissa tuotettu koodi ei ole riittävän laadukasta sellaisenaan avoimena lähdekoodina julkaistavaksi, olen pyrkinyt huolellisesti dokumentoimaan eri työvaiheet, niissä kohdatut ongelmat ja tehdyt havainnot myöhemmin mahdollisesti vastaavan projektin toteuttamista suunnittelevien hyödyksi. Kehittämistyötäni voi siis pitää pilottiprojektina jollekin myöhemmin toteutettavalle hankkeelle, jonka tekijät pystyvät hyötymään ylös kirjaamistani havainnoista ja tiedoista niin, että heidän ei tarvitse välttämättä toistaa kaikkia samoja työvaiheita eikä tehdä kaikkia samoja virheitä. Tämän vuoksi näen, että työ hyödyttää ainakin jossain määrin koko suomalaista datajournalistista kenttää.

1.2 Työn rakenne

Tätä johdantolukua seuraavassa toisessa pääluvussa kuvailen kehittämistyölle asetetut tavoitteet, aiheen rajauksen sekä käyttämäni kehittämismenetelmän.

Kolmannessa pääluvussa esittelen työn tietoperustan, joka jakautuu kolmeen alakokonaisuuteen: tietotekniikka, luonnollisen kielen käsittely sekä datajournalismi.

Työn ylivoimaisesti laajin osa on neljäs pääluku, jossa kuvailen tarkoin käyttämäni tekniset ja muut menetelmät. Olen pyrkinyt dokumentoimaan käytetyt välineet ja tehdyt toimenpiteet niin tarkoin, että tämä luku voisi toimia apuna ja ohjenuorana lukijalle, joka haluaa toteuttaa itse vastaavan tietokannan.

Viidennessä pääluvussa esittelen toteuttamani kaksi datajournalistista juttuhahmotelmaa, sekä mitä niiden tekeminen paljasti tietokannan ja siihen rakennetun selainkäyttöliittymän käyttökelpoisuudesta. Juttuihin liittyvät grafiikat löytyvät kehittämistyön liitteistä 3 ja 4.

Kuudes ja viimeinen pääluku vetää yhteen kehittämistyön tulokset ja vastaa työn alussa esitettyihin tutkimuskysymyksiin. Lisäksi luku sisältää tiiviin kuvauksen siitä, mitä tekisin työn tuloksena oppimani valossa toisin, jos ryhtyisin toteuttamaan vastaavaa hanketta nyt.

2 Tavoitteet

Tässä luvussa kuvaan valitsemani kehittämismenetelmän, tutkimuskysymykset sekä tavoitteeni tutkimuksen kohteena olevan ongelman ratkaisevalle tuotokselle, joka on hakutoiminnolla varustettu tietokanta eduskunnan täysistunnon keskusteluista.

2.1 Aiheen valinta ja rajaus

Pyrin työlläni vastaamaan seuraaviin tutkimuskysymyksiin:

- Millaisia taitoja ja välineitä skannatuista asiakirjoista koostuvan aineiston siirtäminen tietokantamuotoon vaatii?
- Miten journalistit voivat hyödyntää luonnollisen kielen käsittelyä ja tietokantoja omassa työssään?
- Kuinka hyvin tietokanta soveltuu journalistisena sisältönä julkaistavaksi?

Kehittämistyön konkreettisena tuotoksena on eduskunnan täysistuntojen keskustelupöytäkirjat sisältävä, hakutoiminnolla varustettu (*searchable*) tietokanta ja sen selainpohjaisen käyttöliittymä. Perustelen seuraavaksi, miksi katson juuri tämän konkreettisen tuotoksen soveltuvan hyvin konstruktiivisen tutkimukseni kohteena oleviin tutkimuskysymyksiin vastaamiseen.

Eduskunta on Suomen merkittävin yhteiskunnallinen vallankäyttäjä, jolla on yli satavuotinen historia. Eduskunnan täysistunnossa käydyt keskustelut ovat lähtökohtaisesti aina uutisarvoisia ja tiedotusvälineet käsittelevätkin niitä säännöllisesti ja kattavasti. Parlamentissa käytyjen keskustelujen pohjalta voidaan tarkastella sekä yhteiskunnallisia ilmiöitä, että kieltä itseään (Hyvönen et al. 2021). Kuitenkaan eduskunnan omilla sivuilla oleva hakutoiminto ei kovin hyvin sovellu journalistiseen tiedonhankintaan etenkään eduskunnan vanhemmista keskusteluista (käsittelen tätä lähemmin luvussa 4), joten oman, hakutoiminnolla varustetun tietokannan kehittäminen on mielestäni journalistisesti perusteltua. Lisäksi projekti vastaa hyvin itselleni asettamiini oppimistavoitteisiin, jotka esittelen seuraavassa alaluvussa.s

2.2 Tavoitteet kehittämistyölle

Kehittämistyötä aloittaessani asetin sille seuraavat tavoitteet:

- Tutkia, millainen prosessi eduskunnan keskustelupöytäkirjojen analysointi ja siirtäminen hakutoiminnolla varustettuun tietokantaan olisi
- Dokumentoida työn vaiheet ja vaikeudet niin, että kehittämistyötä voidaan käyttää myöhemmin vastaavan projektin apuna ja välttää mahdolliset prosessissa itse tekemäni virheet

- Saada aikaiseksi toimiva verkkopalvelu, jonka avulla voi tehdä hakuja eduskunnan keskustelupöytäkirjoista ja ladata hakutulokset massa-aineistona
- Oppia hyödyntämään luonnollisen kielen käsittelyn menetelmiä ison aineiston käsittelyyn
- Oppia pystyttämään ja hallinnoimaan tietokantaa ja tekemään tietokantakyselyitä SQL-kielen avulla
- Oppia, miten rakennetaan pilvipalveluna toimivaa tietokantaa hyödyntävä verkkopalvelu

Kehittämistyöni lopputuotteena olisi ihannetapauksessa siis toimiva verkkopalvelu, mutta se on vain yksi työlle asetetuista tavoitteista, enkä pidä työtä kokonaisuudessaan epäonnistuneena, vaikka verkkopalvelun toteuttaminen ei toivotulla tavalla onnistuisi.

2.3 Kehittämismenetelmä

Ylempään ammattikorkeakoulututkintoon sisältyvän työelämän kehittämistyön on tarkoitus osoittaa tekijänsä valmius itsenäiseen ja vaativaan asiantuntijatyöhön. Työssä korostuu näyttöön perustuvan tiedon ja tutkimustiedon soveltaminen ja valittujen menetelmien perusteltu ja itsenäinen käyttö työelämän ongelmien erittelyyn ja ratkaisemiseen. (Tuomi 2021.) Työ ei ole jokapäiväisen kehittämistyön kuvaus eikä projektiraportti, mutta ei myöskään tieteellinen tutkielma. Lopputuotteena on jonkinlainen käytännön parannus tai uudenlainen ratkaisu tai toimintamalli, mutta sen jalkauttaminen käytäntöön voi myös jäädä opinnäytetyön jälkeiseen vaiheeseen. (Humak 2021.)

Tässä kehittämistyössä lähestymistavaksi on valittu tutkimuksellinen kehittämistoiminta, jossa yhdistyy konkreettisen kehittämistoiminnan ja tieteellisen tutkimuksen piirteitä. Sen lähtökohtana ovat käytännön ongelmat ja kysymykset, mutta tuotettu tieto palvelee yhtä työpaikkaa laajemman akateemisen ja ammatillisen yhteisön intressejä. (Tuomi 2021.) Kehittämistutkimukselle on ominaista syklinen lähestymistapa ja iterointi eli samojen työvaiheiden toistaminen, kunnes haluttu tulos on saavutettu (Tilastokeskus 2020). Pyrkimyksenä on löytää aiemmin tunnistettuun ongelmaan ratkaisu tai parantaa olemassa olevaa menetelmää tai prosessia, ei luoda varsinaista uutta tieteellistä teoriaa (Kananen 2015, s. 33–34). Pyrkimyksenä ei kuitenkaan ole vain yhden konkreettisen ongelman ratkaisu vaan myös tiedon tuottaminen laajemman keskustelun tarpeisiin (Tuomi 2021).

Tutkimuksellisessa kehittämistyössä voidaan soveltaa mitä tahansa läpinäkyvää ja selkeästi dokumentoitua menetelmää tai niiden yhdistelmää (Humak 2021). Tavallisia menetelmiä ovat toimintatutkimus, kehittävä työntutkimus, käytäntötutkimus, työelämän tutkimusavusteinen kehittäminen, konstruktiivinen tutkimus, tapaustutkimus, palvelumuotoilu, kokeilukulttuuri, taisteleva tutkimus, ennakointi ja innovaatioiden tuottaminen (Tuomi 2021; Humak 2021). Menetelmiä voi kehittää myös itse, sillä

menetelmä ei kehittämistyössä ole itsetarkoitus, vaan sen tärkein tehtävä on taata, että lukija pystyy seuraamaan miten lopputuotokset ja tutkimustulokset ovat syntyneet (Humak 2021).

Sovellan työssäni suurelta osin konstruktiivisen tutkimuksen menetelmiä. Konstruktiivisessa tutkimuksessa rakennetaan tutkimuksen kohteena olevan ongelman ratkaiseva tuotos (Humak 2021), joka omassa kehittämistyössäni on hakutoiminnolla varustettu tietokanta eduskunnan keskustelupöytäkirjojen sisällöistä.

2.3.1 Kehittämistutkimuksen vaiheet

Kanasen (2015, s. 40–42) mukaan kehittämistutkimus jakautuu kahteen vaiheeseen: tutkimus- ja muutossykliin. Ne jakautuvat edelleen alakohtiin seuraavasti:

Tutkimussykli

1. Nykytilan kartoitus
2. Ongelman havaitseminen
3. Ongelman määrittely
4. Vaihtoehtojen etsiminen
5. Vaihtoehtojen arviointi ja ratkaisun valitseminen

Muutossykli

1. Kokeilu/toteutus
2. Arviointi
3. Seuranta

Olen jäsentänyt tämän kehittämistyön luvut 4 ja 5 Kanasen esittämään malliin osin nojautuen. On kuitenkin syytä huomioida, että työmäärällisesti selkeästi isoin osuus omassa työssäni on ollut muutossyklin ensimmäinen vaihe, ”kokeilu/toteutus”, mikä näkyy myös siinä, että sille on varattu isompi osuus sivumäärästä kuin kaikille muille vaiheille yhteensä. Muutossyklin viimeinen vaihe (seuranta) jää kehittämistyön valmistumisen jälkeiseen aikaan.

Oma työni jakaantui seuraaviin vaiheisiin:

- Nykytilan kartoitus ja ongelman tunnistaminen
- Pöytäkirjojen ja kansanedustajamatrikkelin tietojen kerääminen ja käsittely tietokantaa varten
- Käsitellyn tiedon siirtäminen tietokantaan
- Verkkopohjaisen käyttöliittymän rakentaminen
- Datajournalististen juttujen laatiminen tietokannan tietojen pohjalta

Olen seuraavassa hahmotellut, miten yllä kuvatut työvaiheet vastaavat Kanasen mallin vaiheita. Numerointi viittaa kehittämistyöni alalukujen numerointiin.

Tutkimussykli

4.1.1 Nykytilan kartoitus, ongelman havaitseminen ja määrittely (tutkimussykli 1.–3.)

4.1.2 Vaihtoehtojen etsiminen ja arviointi (tutkimussykli 4.–5.)

Muutossykli

4.2 Aineiston kerääminen ja muuttaminen koneluettavaan muotoon (muutossykli 1.)

4.3 Aineiston analysointi ja järjestäminen (muutossykli 1.)

4.4 Aineiston tallettaminen tietokantaan (muutossykli 1.)

4.5 Verkkopohjaisen käyttöliittymän rakentaminen (muutossykli 1.)

5.1 Datajournalististen juttujen laatiminen tietokannan tietojen pohjalta (muutossykli 2.)

6 Yhteenveto ja pohdintaa (muutossykli 2.)

Muutossyklin vaihe 3 jää kehittämistyön jälkeiseen aikaan.

3 Tietoperusta

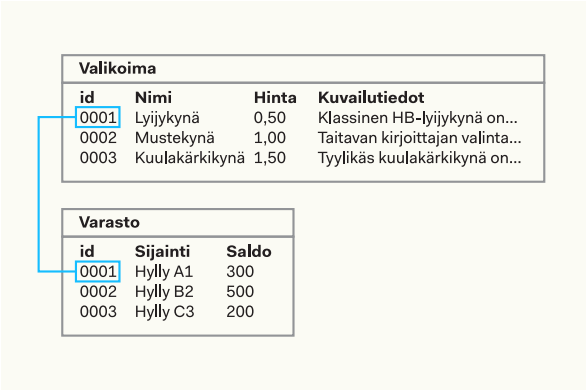
Esittelen tässä luvussa kehittämistyöni aikaisempaan tutkimus- ja kehittämistyöhön nojaavan tietoperustan. Olen strukturoinut luvun siten, että esittelen työni kannalta keskeisten käsitteiden merkityksen sekä samassa yhteydessä niihin liittyvää aiempaa tutkimus- ja muuta tietoa.

3.1 Tietoteknisiä käsitteitä

Esittelen seuraavassa eräitä kehittämistyöni kannalta keskeisiä tietotekniikan käsitteitä.

Tietokanta on TEPA-termipankin (Sanastokeskus TSK s.a.) mukaan ”digitaalisessa muodossa oleva rakenteinen tietojen kokoelma, jota yksi tai useampi tietojärjestelmä käyttää ja päivittää”. Laaksonen (2020) muotoilee saman asian siten, että tietokanta on ”tietokoneella oleva kokoelma tietoa, johon voidaan suorittaa hakuja ja jonka sisältöä voidaan muuttaa”. Tietokanta voidaan organisoida monin eri tavoin, mutta nykyään yleisin on **relaatiomalliin** perustuva tietokanta, jossa tiedot esitetään **tauluihin** tallennettuina **riveinä** ja **sarakkeina**. Saman tyyppistä tietoa sisältävät rivit tallennetaan samaan tauluun ja eri tauluihin tallennetut rivit voivat viitata toisiinsa.

Esimerkiksi verkkokaupan tuotetietokanta voitaisiin organisoida vaikkapa siten, että jokainen myynnissä oleva tuote on rivi *Valikoima*-taulussa, jonka sarakkeita ovat esimerkiksi tuotteen nimi, hinta ja kuvailutiedot. Tavallisesti yhtenä sarakkeena olisi myös **id-numero**: yksilöivä tunniste, jonka avulla rivi voidaan yhdistää muihin tauluihin – kuten esimerkiksi *Varasto*-tauluun, jonka sarakkeina ovat kunkin tuotteen sijainti varastossa sekä varastosaldo. Yleinen tietokannan suunnitteluperiaate on, että kukin tieto tallennetaan kantaan vain kerran (Laaksonen 2020) – esimerkiksi tuotteen nimi esiintyy vain *Valikoima*-taulussa, eikä sitä toisteta enää uudestaan *Varasto*-taulussa.



Kuva 1. Esimerkki yksinkertaisesta tietokantarakenteesta.

Tietokannan taulut muistuttavat loogiselta rakenteeltaan taulukkolaskentaohjelman (esim. Excel) työkirjoja ja saman tiedon voisi useimmissa tapauksissa helposti tallentaa tietokannan sijaan tiedostoiksi. Ongelmaksi tässä vaihtoehdossa kuitenkin muodostuu, että tiedostojen sisäiset ja niiden väliset viittaukset

eivät toimi yhtä mutkattomasti kuin viittaukset tietokantataulujen välillä, vaan tietyn kohteen löytäminen tiedoston muotoon tallennetusta taulukosta vaatii usein koko tiedoston läpikäymistä rivi riviltä kunnes haluttu tieto löytyy. Niinpä tietomäärän kasvaessa suureksi relaatiotietokanta tarjoaa huomattavasti tiedostomuotoista tallennusta helpomman ja nopeamman pääsyn aineistoon.

Relaatiotietokantoja käsitellään tavallisesti **SQL-ohjelmointikielellä** (Laaksonen 2020).

Esittelen esimerkkejä tietokantojen käytöstä journalismissa alaluvussa 3.3.2.

Rakenteisella aineistolla tarkoitetaan sähköisessä muodossa tallennettua aineistoa, jossa tietoelementit ja niiden väliset suhteet on ilmaistu sovitulla tavalla. (Sanastokeskus TSK s.a.) Tällöin aineisto on **koneluettava**, eli se pystytään helposti lukemaan ja tulkitsemaan tietokoneella ohjelmallisesti – erotuksena **ihmisluettavasta** aineistosta, joka taas soveltuu helposti ihmisen tulkittavaksi.

Rajapinta on ”standardin mukainen käytäntö tai yhtymäkohta, joka mahdollistaa tietojen siirron laitteiden, ohjelmien tai käyttäjän välillä” (Sanastokeskus TSK s.a.). Yleisimmin rajapinnasta puhuessa tarkoitetaan ohjelmointi- tai palvelurajapintaa eli APIa (*application programming interface*). Ohjelmointirajapinta määrittelee ne tavat, joilla ohjelmisto tarjoaa tietoa tai palveluita toisille sovelluksille tai tietojärjestelmille. Kyseessä voi olla datarajapinta, jonka kautta saa luettua dataa tai toiminnallinen rajapinta, joka tarjoaa erilaisia laskentapalveluita tai mahdollisuuden muuttaa järjestelmään tallennettua tietoa. Avoin rajapinta vuorostaan on sellainen ohjelmointirajapinta, jonka kaikki ominaisuudet ovat julkisia ja vapaasti ja maksutta kenen hyvänsä käytettävissä. (Poikola et al. 2014.)

Tiedonharavointi (engl. *[data] scraping*) tarkoittaa TEPA-termipankin (Sanastokeskus TSK s.a.) mukaan ”tietojen automaatti[sta] kokoami[sta] ihmisen luettavassa muodossa olevasta aineistosta.” Menetelmän avulla siis kerätään useaan eri kohteeseen tallennettua ihmisluettavaa tietoa yhteen paikkaan rakenteiseen eli koneluettavaan muotoon. Tavallisimmin termillä tarkoitetaan nimenomaisesti **verkkosivujen haravointia** (engl. *web scraping*), jossa koottavien tietojen lähteenä ovat verkkosivut. Verkkosivujen haravoinnissa haravointiohjelma (ammattislangissa *skreipperi*), joka on usein ohjelmoitu erikseen tiettyä haravointitehtävää varten, käy siis systemaattisesti läpi suuren joukon verkkosivuja.

Tavallisin käyttötapaus on, että haravointiohjelma tallentaa halutut tiedot yhteen paikkaan rakenteisessa muodossa myöhempää laskennallista käsittelyä varten, mutta haravoinnin verrattain yleinen käyttökohde ovat myös verkkosivujen päivityksiä reaaliaikaisesti seuraavat robotit kuten Pikkulintu, joka on STT:n uutistoimituksille tarjoama palvelu

verkkosivujen seurannan automatisointiin. Se hakee haluttujen sivujen tiedot säännöllisesti ja lähettää hälytyksen toimittajan sähköpostiin kun se havaitsee sivulla etukäteen määriteltyihin kriteereihin sopivan muutoksen. STT:n politiikan toimittajat käyttävät työkalua itse esimerkiksi eduskunnan valiokuntien toiminnan seuraamiseen. Pikkulintu lähettää hälytyksen toimittajalle kun valiokunnan esityslistalla esiintyy etukäteen määritelty, journalistisesti kiinnostava avainsana tai tietty oleva asiantuntija on kutsuttu valiokunnan kuultavaksi. (STT s.a.)

Tiedonharavointi on yksi käytetyimmistä datajournalismin menetelmistä ja sitä on käytetty mitä erilaisimpien juttutyyppien tiedonhankintamenetelmänä. The Economist (29.1.2022) on esimerkiksi selvittänyt Spotify-suoratoistopalvelusta haravoidun datan avulla englanninkielisen musiikin suosion kehitystä muunkieliseen musiikkiin.

Avoimella lähdekoodilla tarkoitetaan sellaista tapaa kehittää ja jaella tietokoneohjelmistoja, joissa ohjelmiston käyttäjä saa ohjelman ja sen lähdekoodin ilmaiseksi käyttöönsä ja saa halutessaan vapaasti kopioida, muunnella ja jakaa ohjelmaa eteenpäin. (COSS ry. s.a.) Avoimen lähdekoodin ohjelmia kehitetään etenkin yliopistomaailmassa sekä harrastajien ja aktivistien toimesta, mutta myös monet kaupalliset yritykset ovat katsoneet hyötyvänsä enemmän tavalla tai toisella siitä, että heidän alkujaan kehittämiensä ohjelmien lähdekoodi on avattu vapaasti käytettäväksi ja muokattavaksi. Journalisteille avoimen lähdekoodin ohjelmistot tarjoavat paitsi kustannustehokkuutta ja laajemman valikoiman työkaluja, myös läpinäkyvyyttä, sillä kuka hyvänsä voi tarkistaa ohjelman lähdekoodin sisällön.

Keinoälyn alakäsite **koneoppiminen** on yleisnimitys tietokoneohjelmille, jotka analysoivat automaattisesti suuria aineistojoukkoja ja etsivät niistä potentiaalisesti kiinnostavia piirteitä. Eräs yleinen koneoppivien mallien ryhmä on *syväoppivat neuroverkot*, joita käytetään mm. luonnollisen kielen käsittelyyn (ks. seuraava alaluku). (Kelleher & Tierney 2021, s. 25.) Koneoppimisen erityisominaisuus perinteisiin tietokoneohjelmiin verrattuna on se, että neuroverkot oppivat uusia taitoja, ilman, että niitä tarvitsee malliin erikseen ohjelmoida (Kukkamäki & Lehtomäki 2020). Pohjana on mallin kouluttaminen soveltuvalla koulutusaineistolla. Esimerkiksi koneoppimiseen perustuva roskapostisuodatin koulutetaan näyttämällä neuroverkolle suuri joukko sähköposteja, joista osa on nimiöity roskapostiksi ja osa ei. (Kelleher & Tierney 2021, s. 102–106.)

Suurin osa koneoppivista algoritmeista voidaan jakaa karkeasti kahteen luokkaan: **ohjattua** ja **ohjaamatonta oppimista** hyödyntäviksi (Kelleher & Tierney 2021, s. 100–102). Ohjattu oppiminen toimii siten, että mallille näytetään joukko oikeita ratkaisuja ja se päättelee niiden perusteella mikä funktio tuottaa halutunlaisia tuloksia. Esimerkiksi edellä

kuvattu neuroverkkoon perustuva roskapostisuodatin soveltaa ohjattua oppimista: malli muodostaa aineiston perusteella itsenäisesti käsityksen siitä, mitkä tekijät ovat tyypillisiä roskapostille ja pystyy sen jälkeen luokittelemaan sille annetut uudet, koulutusaineistoon kuulumattomat sähköpostit kohtuullisella tarkkuudella roskapostiksi tai ei-roskapostiksi. Mallin kouluttajien ei siis tarvitse pystyä kertomaan ohjelmalle, *miksi* jokin viesti on roskapostia ja jokin toinen ei – riittää, että mallille annetaan riittävän laaja koulutusaineisto ja kerrotaan *mitkä* viestit ovat roskapostia ja mitkä ei, ja se oppii itsenäisesti päättelemään, mitkä tekijät tekevät viestistä roskapostia.

Ohjaamatonta oppimista soveltava malli vuorostaan ei pyri löytämään mitään tiettyjä ”oikeita” vastauksia vaan ainoastaan hahmottamaan aineistossa esiintyviä piirteitä ja esimerkiksi tunnistamaan toisiinsa liittyvien tietojen ryppäitä eli klustereita (Kelleher & Tierney 2021, s. 102–106). Ohjaamattomaan oppimiseen perustuvat esimerkiksi monet pääkomponenttianalyysin ja muiden tilastollisten monimuuttujamenetelmien pohjana olevat laskentamenetelmät.

Keinoälyä käytetään yhä enenevässä määrin journalismin työkaluna. Knight-säätiö löysi laajassa kartoituksessaan 130 vuosina 2012–2020 eri medioiden kehittämää koneoppimista hyödyntävää hanketta tai työkalua. Suomesta listalle oli päässyt Ylen Voitto-botti. Tavallisimmin menetelmää hyödynnetään perinteistä toimitustyötä tukevissa työkaluissa, mutta projektit, joissa koneoppimisella on itsenäinen rooli tutkivan journalismin välineenä ovat yleistymässä. (Keefe et al. 12.5.2021.) Ukrainalainen Texty.org.ua-verkkomedia esimerkiksi paljasti laajan laittomien meripihkakaivosten verkoston maan Valko-Venäjän vastaisen rajan läheisyydessä analysoimalla neuroverkon avulla 450 000 alueelta otettua satelliittikuvaa. Analyysiä täydennettiin ja korjattiin lukijoilta saadun palautteen perusteella. (Fouquenet 7.7.2021.)

3.2 Luonnollisen kielen käsittely

Luonnollisella kielellä tarkoitetaan kieltä, jolla ihmiset kommunikoivat keskenään vapaamuotoisesti puheen tai tekstin muodossa (Kukkamäki & Lehtomäki 2020). Ihmisten käyttämät kielet ovat rakenteiltaan monimutkaisempia ja sanastoltaan laajempia kuin esimerkiksi tietokoneohjelmien rakentamiseen käytetyt keinotekoiset kielet, joten niitä ei voi ainakaan kovin helposti tulkita automaattisesti. Tietokantaan tai muuhun muotoon tallennettuna luonnollista kieltä voi pitää *rakenteettomana* datan muotona.

Luonnollisen kielen käsittely, josta käytetään usein englanninkielistä lyhennettä NLP (*natural language processing*) on yleiskäsite niille menetelmille, joilla ihmisen tuottamaa kielellistä ilmaisua pyritään käsittelemään ja tulkitsemaan tietokoneen avulla (Kukkamäki

& Lehtomäki 2020). Luonnollisen kielen käsittely pohjaa yleensä koneoppimisen menetelmien hyödyntämiseen.

Luonnollisen kielen käsittelyn menetelmiä on hyödynnetty datajournalismissa kansainvälisesti melko paljon. Tavanomaisia käyttökohteita ovat isojen dokumenttimassojen kuten esim. ns. Panaman papereiden läpikäynti automaattisesti uutisarvoisten dokumenttien löytämiseksi, ja esimerkiksi sosiaalisessa mediassa, parlamentissa tai jossain muussa foorumissa käydyn keskustelun jäsentely teemoihin.

Esittelen seuraavassa eräitä keskeisiä luonnollisen kielen käsittelyn käsitteitä, joihin viittaan useasti myöhemmin tässä dokumentissa.

Korpus on Tieteen termipankin (4.1.2022) määritelmän mukaan ” suhteellisen laaja, järjestelmällisesti kerätty kokoelma näytteitä luonnollisesta kirjoitetusta tai puhutusta kielestä”. Tunnettuja suomalaisia korpuksia ovat esim. 1960-luvulla koottu ns. Oulun korpus sekä 1990-luvun lopulla koostettu Suomen kielen tekstipankki (Suominen 2010). Luonnollisen kielen analyysissä korpuksia käytetään mm. koneoppimismallien kouluttamiseen. Tässä kehitystyössä muodostettua tietokantaa eduskunnan täysistunnon keskusteluista voi pitää korpuksena, joskin sen tekninen laatu on heikompi kuin tutkimuskäyttöön kerättyjen korpusten. Eduskunnan keskusteluista on koostettu myös tutkimuskorpuksia, joita käsitellään tarkemmin alaluvussa 4.1.2.

Jäsennin (*parser*) on ”tietokoneohjelma, joka lisää kielelliseen ilmaukseen kieliopillista tai rakennetta koskevaa tietoa” (Sanastokeskus TSK s.a.). Kehittämistyöni kannalta kiinnostavia ovat **morfologiset jäsentimet**, joiden tehtäviä ovat mm. sanaluokittelu eli tekstissä esiintyvien sanojen sanaluokkien merkitseminen sekä perusmuotoistaminen eli normalisointi, jota käsitellään lähemmin alempana. (Kukkamäki & Lehtomäki 2020.)

Saneistus eli tokenisointi tarkoittaa tekstin pilkkomista analyysia varten perusyksiköihin eli saneisiin, joita ovat sanat, sanaliitot ja esimerkiksi puhelinnumerot, verkko-osoitteet ja muut merkitykselliset numero- ja merkkisarjat. Suomen tapaisessa kielessä, jossa suositaan sanaliittojen sijaan yhdyssanoja saneistus onnistuu kohtuullisen hyvin yksinkertaisesti erottelemalla lauseiden sanat toisistaan välilyöntien ja rivivaihtojen avulla, mutta esimerkiksi englannissa monen sanaparin kuten vaikkapa ”ice cream” merkitys on eri kuin sen muodostavien sanojen erikseen. Tällöin saneiden erittely pelkästään välilyöntien ja -merkkien perusteella ei tuota tyydyttävää tulosta, vaan tarkoitukseen on suotavaa käyttää jonkinlaista kaavaa tai mallia. (Kukkamäki & Lehtomäki 2020.)

Normalisoinnilla tarkoitetaan menetelmiä, joilla tekstissä esiintyvät eri sanojen taivutusmuodot muutetaan niin, että saman sanan eri taivutusmuodot tunnistetaan

samaksi sanaksi. Kaksi yleisimmin käytettyä menetelmää ovat stemmaus ja lemmaus eli perusmuotoistaminen. (Kukkamäki & Lehtomäki 2020.) **Stemmauksessa** sanoista poistetaan niiden lopusta sanojen taivutusmuotoon viittaavat kirjaimet, siten että esimerkiksi sanasta ”kirjoitettu” poistetaan partisiippimuotoa ilmaisevat kirjaimet *-ettu*, jolloin jäljelle jäävä sanan runko-osa on ”kirjoit”. Vaikka tämä muoto ei sellaisenaan vastaa mitään sanakirjasta löytyvää sanaa, eräissä lauserakenteeltaan yksinkertaisissa kielissä kuten englannissa ja ruotsissa stemmaus tuottaa usein käyttökelpoisia tuloksia, joissa runko-osien merkitys on sellaisenaan helposti ymmärrettävissä tai ne ovat joissain tapauksissa jopa algoritmisesti muutettavissa sanan perusmuodoksi. Stemmausalgoritmin tekninen toteutus on suhteellisen helppoa ja yksinkertaisiin sääntöihin perustuvina algoritmit ovat myös varsin nopeita. Niinpä stemmaus onkin maailmanlaajuisesti kaikkein yleisimmin käytetty sanojen normalisointitekniikka. (Kukkamäki & Lehtomäki 2020.) **Lemmaus** eroaa stemmauksesta siten, että siinä sanan perusmuoto eli lemma etsitään sanakirjan avulla morfologista analyysiä eli tekstin muoto-opillisten piirteiden tulkintaa hyödyntäen. Suomen tapaisen kielen normalisointi stemmaamalla tuottaa epätyydyttäviä tuloksia, joten suomenkielisen tekstin normalisointiin käytetään yleensä mieluummin morfologiseen analyysiin ja sanakirjaan perustuvaa **perusmuotoistajaa** eli **lemmatisoijaa**. (Kukkamäki & Lehtomäki 2020.) Toisaalta suomi on morfologisesti rikas kieli, mikä tuottaa erityisiä haasteita tekstin koneelliselle analyysille. Morfologiseen analyysiin käytettävät ohjelmistot voivat tuottaa toisistaan poikkeavia tuloksia ja tehdä virheitä, joten käyttäjän on suhtauduttava kriittisesti käyttämänsä ohjelman tuotoksiin ja tarkistettava ne ainakin pistokoeluontoisesti käsin. (Suominen 2010.)

Hukkasana (*stop word*) on TEPA-termipankin (Sanastokeskus TSK s.a.) mukaan ”hyödyttömäksi katsottu sananmuoto, joka suljetaan käsittelyn ulkopuolelle”. Hukkasanoilla tarkoitetaan hyvin yleisiä sanoja, joilla ei ole itsenäistä merkitystä käsittelyssä olevan aineiston kontekstissa. Tavallisesti hukkasanoina pidetään esimerkiksi partikkeleita ja olla-verbin taivutusmuotoja (Jaakkola 1997), mutta jotkin sellaiset sanat, joita ei normaalisti pidetä hukkasanoina voivat tietyssä kontekstissa olla sellaisia ja päin vastoin. Esimerkiksi eduskunnan keskustelupöytäkirjojen kontekstissa vaikkapa sanaa ”puhemies” voisi pitää hukkasanana, sillä sanapari ”herra puhemies”, ”rouva puhemies” tai ”arvoisa puhemies” toistuu useimmissa puheenvuorossa – monesti jopa useita kertoja.

Koska tämä kehittämistyö on ensimmäinen laajalla aineistolla tekemäni luonnollisen kielen käsittelyn alaan lukeutuva työ, päätin rajata siinä käytettävät menetelmät pääosin edellä kuvattuihin. Esittelen seuraavassa kuitenkin vielä lyhyesti muutamia yleisiä NLP-menetelmiä, joiden soveltaminen eduskunnan täysistuntojen keskusteluista muodostettuun korpukseen voisi myös olla sisällöllisesti perusteltua.

Aihemallinnus on yleisnimitys joukolle tilastollisia menetelmiä, joiden avulla pyritään tunnistamaan merkityksellisiä ryhmiä jäsentelemättömästä tekstistä. Eräs yleisimmistä aihemalleista, LDA (*latent Dirichlet allocation*) on ohjaamattomaan oppimiseen perustuva koneoppimismalli. Siinä käyttäjä antaa algoritmille erillisistä teksteistä koostuvan aineiston (korpuksen) ja määrittelee, montako ryhmää eli aihetta sen pohjalta algoritmi luo. Algoritmi luokittelee sen jälkeen tekstit ja antaa kullekin todennäköisyyden kuulumisesta tiettyyn aiheeseen. Sopivan aihemäärän valitseminen on kriittinen tehtävä LDA-mallia käytettäessä, sillä liian pieni aihemäärä tuottaa laajoja, epäspesifisiä aiheita ja liian suuri määrä taas tuottaa ryhmiä, jotka kuvaavat sisällöllisesti toissijaisia, kielen ominaisuuksiin ja käyttötapoihin perustuvia yhteyksiä. Esimerkiksi eduskunnan täysistunnon pöytäkirjoja mallinnettaessa sanapari ”ministeri” ja ”kysyä” voisi tällaisessa tapauksessa muodostua omaksi aiheekseen, sillä nämä kaksi sanaa esiintyvät korpuksessa erittäin usein yhdessä. (Makkonen & Loukasmäki 2019.)

Sentimentti- eli tunneanalyysi pyrkii tunnistamaan tekstissä esiintyviä tunteita. Sentimenttianalyysi voidaan toteuttaa myös koneoppimista kevyemmillä menetelmillä hyödyntäen ns. *tunnesanalistoja*. Ne ovat yksinkertaisimmillaan listoja sanoista, jotka yhdistetään tiettyihin tunteisiin kuten pelko, suru tai ilo. (Helsingin yliopisto 3.3.2021.)

Luonnollisen kielen ymmärtäminen (NLU, *natural language understanding*) on kattotermi teknologioille, joita hyödynnetään luonnollista kieltä ymmärtävien käyttöliittymien kuten virtuaaliavustajien ja keskustelubottien rakentamiseen. Tyypillisesti kyseessä on neuroverkko, joka on koulutettu suurella määrällä ihmisen koneen ymmärtämään muotoon tulkkaamia syötteitä. (Kukkamäki & Lehtomäki 2020.)

3.3 Datajournalismi

The Guardian -lehden entisen datapäätoimittajan Simon Rodgersin napakan määritelmän mukaan datajournalismi on ”uutisten löytämistä ja kertomista numeroiden avulla” (Stray 2010). Knightin (2015) mukaan datajournalismin lavein määritelmä on, että jutun lähteenä tai aiheena ovat numerot, tai että jutussa on mukana merkittävä data- tai visualisointikomponentti. Datajournalismia onkin pidetty toisaalta tutkivan journalismin, toisaalta visuaalisen journalismin alalajina (esim. Hewett 2013, s. 3–5). Yhtenä datajournalismin erityispiirteenä on pidetty myös sitä, että jutut sisältävät yleensä jonkinlaisen kuvauksen siitä, miten ne ovat syntyneet (Uskali & Kuutti 2016, s. 14).

Paul Bradshaw (2012) kirjoittaa: ”Mikä tekee datajournalismista erilaista kuin muu journalismi? Ehkäpä ne uudet mahdollisuudet, jotka avautuvat, kun perinteinen ’uutisvainu’ ja taito kirjoittaa mukaansatempaavia juttuja yhdistyy digitaalisessa muodossa

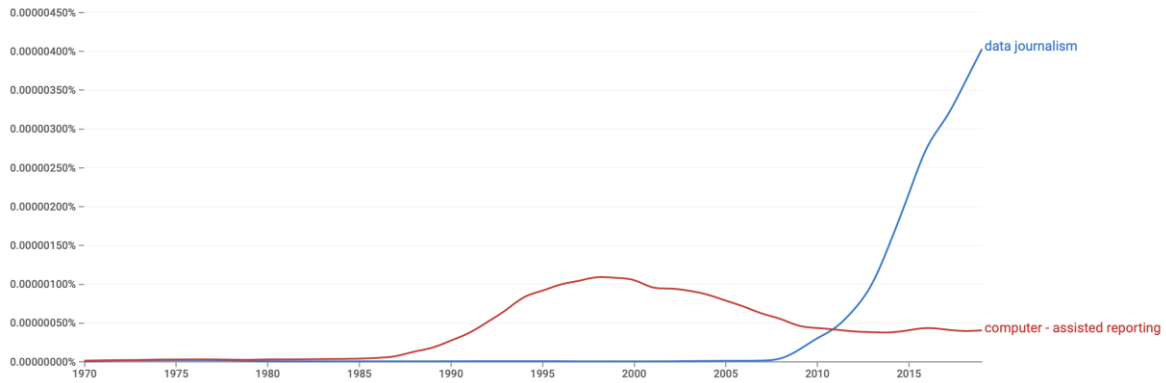
nykyisin saatavilla olevan tiedon valtavaan laajuuteen ja monipuolisuuteen. – – Data voi toimia datajournalismin lähdeaineistona tai tarinankerronnan työkaluna – tai molempina. Kuten muihinkin tietolähteisiin, siihen pitää suhtautua kriittisesti; ja kuten muidenkin työkalujen tapauksessa, on syytä olla tietoinen siitä, miten data muovaa ja rajoittaa sitä, millaisia juttuja sen avulla voidaan synnyttää."

Emily Bellin (2012) mukaan vielä 2000-luvun ensimmäisen vuosikymmenen lopulla isojen data-aineistojen journalistinen käyttö oli harvinaista lukuun ottamatta kourallista rahoitusalan ammattilaisia palvelevia talousmedioita kuten Bloombergia, Dow Jonesia ja Reutersin taloustoimitusta. Nykyisin datajournalisteja työskentelee kuitenkin mitä erilaisimmissa medioissa. Nimikkeellä työskentelevien ihmisten työnkuva saattaa vaihdella taulukkolaskentaohjelman hyödyntämisestä perinteisten uutisjuttujen kirjoittamisessa tilastollisen mallintamisen, graafiseen suunnittelun ja ohjelmistokehityksen menetelmien soveltamiseen. Jotkin mediat ovat ottaneet käyttöön spesifimpiä ammattinimikkeitä kuten koodaava toimittaja (*programmer journalist*, esim. Yle), tietokantatoimittaja (*database editor*, esim. The Los Angeles Times), grafiikkatoimittaja (*graphics editor*, esim. The New York Times) tai visuaalinen datajournalisti (*visual data journalist*, esim. The Economist), datakirjeenvaihtaja (*data correspondent*, esim. The Economist), taikka tilastotoimittaja (*statistics journalist*, esim. The Financial Times) kuvaamaan täsmällisemmin datajournalismin eri työtehtäviä. Tavallista kuitenkin on, että samalla tehtävänimikkeellä työskentelevien toimittajien tarkka työnkuva voi vaihdella saman toimituksen sisälläkin.

Keskeinen, etenkin yhdysvaltalaisessa kontekstissa paljon käytetty datajournalismia sivuava vanhempi termi on jo ainakin 1970-luvulta asti käytössä ollut **tietokoneavusteinen journalismi** (*computer-assisted reporting*, CAR). Esimerkiksi tärkein datajournalismin alan koulutusohjelma ja konferenssi on nimeltään NICAR, National Institute for Computer-Assisted Reporting. Joidenkin (esim. Houston 2019, s. 4) mukaan tietokoneavusteinen journalismi ja datajournalismi ovat synonyymejä. Toiset taas pitävät datajournalismia uuden teknologian mahdollistamana tietokoneavusteisen journalismin ja siitä aiemmin erillisten journalismin muotojen, esim. uutisgrafiikan (Knight 2015) uudenlaisena yhdistelmänä.

Knightin (2015) mukaan *data journalism* -termi on mainittu ensimmäisen kerran The Guardian -sanomalehdessä vuonna 2008. Se on syrjäyttänyt vanhemman termin nopeasti: englanninkielisessä kirjallisuudessa *data journalism* -termin maininnat ohittivat *computer-assisted reporting* -termin jo vuonna 2011 (Google s.a.). Tarve uudelle termille on ymmärrettävä, sillä kuten Knight huomauttaa, CAR-termi on kehitetty aikana, jolloin

tietokoneet olivat huoneen kokoisia jättiläisiä, joihin vain harvalla oli pääsy, kun taas nykyään jokainen toimittaja käyttää työssään tietokonetta.



Kuva 2. *Data journalism* ja *computer-assisted reporting* -termien maininnat englanninkielisessä kirjallisuudessa Google Books Ngram Viewerin (Google s.a.) mukaan.

3.3.1 Datajournalismin historiaa

Ensimmäisenä varsinaisena datajournalistisena juttuna on usein mainittu The Manchester Guardian -lehden (nykyisin The Guardian) ensimmäisessä näytenumerossa 5.5.1821 julkaistu, nimettömään tietovuotoon perustuva suurikokoinen taulukko, joka paljasti mm., että Manchesterissa ilmaiseksi koulua käyvien eli hyvin köyhien lasten määrä oli paljon suurempi, kuin julkisuuteen oli aiemmin kerrottu (Uskali & Kuutti 2016, s. 19). Taulukoita oli toki julkaistu sanomalehdissä aiemminkin – Suomessakin jo vuonna 1784 Tidningar Utgifne Af et Sällskap i Åbo -lehdessä (Mervola 1995, s. 70–76) –, mutta Guardianin artikkeli lienee ainakin vanhin täysin datavetoisella jutulla saatu uutisvoitto.

Vaikka datajournalismin keskeinen ilmaisumuoto, visualisoinnit ja infografiikat, olivat sanomalehdistössä varsin harvinaisia ennen rainaoffset-painomenetelmien[1] yleistymistä 1970-luvuilla, taulukoiden lisäksi myös yksittäisiä grafiikkoja, varsinkin karttoja, julkaistiin

[1] Ennen offset-painoa lehtien painamiseen käytetty kohopainomenetelmä vaati jokaisen julkaistun grafiikan valmistamista metalliseksi kuvalaataksi eli kliseeksi, mikä oli työläs ja kallis operaatio. (Kertaalleen valmistettuja kuvia kierrätettiinkin lehdissä uudestaan ja uudestaan, mistä yleisen käsityksen mukaan juontuu klisee-sanan nykyään tavallisempi, kuvainnollinen merkitys.) Offsetissa taitto-originaalit muutetaan painolevyiksi valokuvaamalla, mikä helpottaa kuvan käyttöä lehdessä ratkaisevasti. Aikakauslehdistö alkoikin siirtyä offset-painamiseen jo 1930-luvulla, mutta sanomalehtipaperin huokoisuus ja tarve suurten painosten tuottamiselle hyvin nopeasti olivat offset-menetelmälle liian suuri haaste ennen rainaoffsetin keksimistä 1950-luvulla. Senkin jälkeen olemassa oleviin paino- ja ladontakoneisiin tehdyt isot investoinnit piti ensin kuolettaa ennen kuin sanomalehdet olivat valmiita investoimaan uuteen teknologiaan.

lehdissä ajoittain jo 1700-luvulla. Vanhin sanomalehdessä julkaistu grafiikka lienee englantilaisen The Daily Courant -lehden vuonna 1702 julkaisema kartta Cádizin lahden linnoituksista (Koponen et al. 2019, s. 150). Varhaisin suomalaisessa sanomalehdessä kartta on vuonna 1855 Sanomia Turusta -nimisessä lehdessä julkaistu Sevastopolin linnoitusta esittävä kartta. Kuvituksellisempia infografiikkoja Suomessa oli julkaistu vuonna 1844 Maamiehen ystävä -lehdessä peräti kaksi kappaletta: navetan rakennepiirros sekä kuva *jäkytteliästä* eli paviaanista. (Mervola 1995, s. 70–76.) Varhaisin tunnettu sanomalehdessä julkaistu tilastografiikka on New York Daily Tribune -lehdessä 29.9.1849 julkaistu viivakuvio koleraan New Yorkissa kuolleiden määrästä (Klein 16.3.2016).

Laskennalliset menetelmät tulivat journalistiseen käyttöön ensimmäisenä Yhdysvalloissa 1900-luvun jälkimmäisellä puoliskolla. Cox (2000) pitää varhaisimpana esimerkkinä tietokoneavusteisesta journalismista CBS:n laatimaa ennustetta Yhdysvaltain vuoden 1952 presidentinvaalien tuloksesta. Ensimmäinen Pulitzer-palkittu tietokoneavusteinen artikkeli oli Detroit Free Pressin analyysi Detroitin vuoden 1967 mellakoiden mustista osallistujista. Jutun kirjoittanut toimittaja Philip Meyer kirjoitti myös ensimmäisen data-analyysin menetelmien käyttö journalismissa käsittelevän kirjan ”Precision journalism” vuonna 1973. Hänestä tuli Pohjois-Carolinan Chapel Hill -yliopiston journalismin professori vuonna 1981. (Cox 2000.)

Yhdysvalloissa varhainen tietokoneavusteinen journalismi käsitteli usein rasismiin ja yhteiskunnalliseen eriarvoisuuteen liittyviä teemoja. Cox (2000) mainitsee tästä useita esimerkkejä, joista edellä mainitun Detroit Free Pressin artikkelin ohella tunnetuin on toinen Pulitzer-voittaja, Atlanta Journal-Constitutionin 1989 julkaisema The color of money -juttusarja, joka paljasti atlantalaisten pankkien syrjivän mustia lainanhakijoita asuntolainoituksessaan.

Kuten edellisessä alaluvussa mainittiin, datajournalismi-termi tuli käyttöön The Guardian -lehdessä vuonna 2008 ja syrjäytti nopeasti aiemman tietokoneavusteinen journalismi (CAR) -käsitteen. Knightin (2015) mukaan vuoden 2010 isot Wikileaks-tietovuodot olivat käännekohta, joka nosti datajournalismin yhtäkkiä journalismin keskiöön.

Suomessa datajournalismilla on lyhyemmät perinteet kuin Yhdysvalloissa. Siinä missä Cox (2000) esittelee joukon eri kokoisia sanomalehtitoimituksia, joissa datan analysoiminen ja visualisoiminen tietokoneilla on ollut arkipäiväistä jo 1990-luvun lopulla, vielä vuonna 2010 Tampereen yliopiston NextMedia-tutkimusprojektia varten Suomesta ei löytynyt ensimmäistäkään datajournalistia haastateltavaksi (Uskali & Kuutti 2016).

Suomalaisen datajournalismin lähtölaukauksena voi pitää Helsingin Sanomien vuosina 2011–2012 järjestämiä HS Open -tapahtumia, jotka kokosivat hackathon-henkisesti toimittajia, graafikoita ja koodaajia saman pöydän ääreen ideoimaan datajournalistisia juttuja avoimen datan pohjalta (Mäkinen 2.3.2011; Mäkinen 28.6.2012). Osallistuin itsekin useimpiin näistä tapahtumista, jotka olivat keskeisessä roolissa suomalaisen datajournalismin varhaisten verkostojen synnyttämisessä.

Helsingin Sanomien sivuilla sana ”datajournalismi” esiintyi ensimmäistä kertaa vasta kun lehti tiedotti oman datajournalismiin keskittyvän ryhmänsä perustamisesta vuonna 2012 (HS 21.5.2012). Vaikka sen muuttumisesta datadeskiksi tiedotettiin vasta vuonna 2015 (Sanoma Media Finland 27.10.2015), toimituspäällikkö Esa Mäkisen mukaan tiimi on toiminut lehdessä itsenäisenä yksikkönä jo vuodesta 2011 alkaen (Mäkinen 16.11.2021). Yle aloitti säännöllisen datajournalististen sisältöjen julkaisemisen vuonna 2012 kun nykyisin Euroopan yleisradiounioni EBU:ssa työskentelevä datajournalisti Teemo Tebest rekrytoitiin Svenska Ylelle. Datajournalistisiin sisältöihin erikoistunut uutis- ja ajankohtaistoimitusten kanssa työskentelevä Plus-deski perustettiin Yleen keväällä 2013. Tämän lisäksi Svenska Ylellä on lisäksi oma, niin ikään vuonna 2013 perustettu Grävnavet-niminen datajournalistinen tiiminsä. (Toikkanen 2014.)

3.3.2 Tietokantajournalismi

Tietokantajournalismi-termille ei ole olemassa täysin vakiintunutta, yleisesti hyväksyttyä määritelmää. Jotkut (esim. Gawiser & Evans Witt 1994, s. 26) käyttävät sitä tietokoneavusteisen journalismin (CAR) tai datarjournalismin synonyyminä, mutta esimerkiksi Kuutin (2011) mukaan tietokantajournalismilla on spesifimpi merkitys. Se on hänen mukaansa ”journalistinen suuntaus tai toimintamalli, jossa tiedonhankinnan kohteena ovat erityisesti viranomaisten hallinnoimat tietokannat tai rekisterit kokonaisuudessaan tai niiden valikoidut yksittäiset sisällöt”. Kuutti erottelee tietokantajournalismin ”verkkojournalismista” eli internetin hyödyntämisestä tiedonhankinnassa, mutta toteaa, että yhdysvaltalaisessa keskustelussa tällaista erottelua ei tehdä vaan siellä yleiskäsite tietokoneavusteinen journalismi kattaa molemmat tiedonhankintakeinot. Hänen mukaansa tietokantajournalismia voi pitää osana ”tallennejournalismia”, jonka toinen osa-alue on ei-digitaalisessa muodossa olevia asiakirjoja hyödyntävä ”dokumenttijournalismi”.

Vaikka Kuutti määrittelee tietokantajournalismin lähinnä tiedonhankintakeinona, hän viittaa artikkelissaan useassa kohdin myös toimitusten omien tietokantojen suunnitteluun ja rakentamiseen ja toteaa, että ”[t]iedonhankinnan ohella tietokantajournalismilla on oma tärkeä merkityksensä myös journalistisessa julkaistutoiminnassa”, mainiten esimerkkinä

mm. journalistisen tiedonhankinnan tuloksena syntyneen raakadatan julkaisemisen sellaisenaan. (Kuutti 2011.) Käsitteen kansainvälisissä määritelmissä (esim. Loosen 2002) taas korostuu yleensä tietokannan merkitys nimenomaan julkaisukanavana tai lopputuotteena, mutta useimmat määritelmät sisältävät myös jonkinlaisen viittauksen tietokantoihin journalistisen tiedonhankinnan välineenä.

On luonnollista, että varhaisemmissa lähteissä tietokantajournalismin määritelmässä korostuu tietokannan rooli nimenomaan tiedonhankinnassa, sillä vielä 1990-luvun alkupuolella tietokannan julkaiseminen lukijoiden tutkittavaksi ei olisi ollut teknisesti mahdollista. Internetin myötä tämä kuitenkin muuttui. Ensimmäisiä dokumentoituja esimerkkejä tietokannasta nimenomaan journalistisena lopputuotteena on Charlotte Observer -sanomalehden vuonna 1998 verkossa julkaisema hakutoiminnolla varustettu tietokanta Charlotten alueen asuntojen hinnoista (Cox 2000).

Yhtenä tietokantajournalismin merkkipaaluna on pidetty Adrian Holovatyn vuonna 2006 kirjoittamaa esseetä ”A fundamental way newspaper sites need to change”. Vaikka Holovaty ei mainitse tekstissään kertaakaan sanaparia *database journalism*, hän antaa monia esimerkkejä journalismissa yleisesti käsitellyistä rakenteisen datan muodoista ja haastaa toimittajat kysymään, tuoko perinteinen tekstimuotoinen journalismi välttämättä esiin kaikkia niiden uutisarvoisia piirteitä, ja onko perinteinen, tekstimuotoisille jutuille tarkoitettu julkaisujärjestelmä paras tapa niiden tallentamiseen ja käsittelyyn julkaisua varten. Holovaty haastaa lukijansa kyseenalaistamaan perinteisen kirjoitetun jutun itsestäänselvyyden sanomalehden ainoana vakavasti otettavana juttutyyppinä ja nostaa esiin esimerkkejä sisällöistä, joissa journalistinen lopputuote on hakutoiminnolla varustettu tietokanta eikä teksti. Hän huomauttaa, että toimittajien pitäisi käyttää vähemmän energiaa sen murehtimiseen, onko tämä tai tuo ”journalismia” ja enemmän ”tärkeän, kohdennetun tiedon, joka hyödyttää ihmisten arkielämää ja auttaa heitä ymmärtämään maailmaa” synnyttämiseen.

Tässä kehittämistyössä lähestyn tietokantajournalismia sanan sellaisessa merkityksessä, joka korostaa tietokantaa journalistisena julkaisukanavana tai lopputuotteena erotuksena muusta datajournalismista.

4 Työn vaiheet ja käytetyt menetelmät

Tässä luvussa kuvailen kehittämistyöni vaiheet, käytetyt tekniset menetelmät ja sekä välineet, jotka voi karkeasti ottaen jakaa luonnollisen kielen käsittelyn (ks. alaluku 3.2), aineiston järjestämisen, tietokannan pystyttämisen sekä käyttöliittymän rakentamisen menetelmiin ja välineisiin.

Kuva 3. Kehittämistyöni vaiheet

4.1 Taustaa

Kerron seuraavaksi, miksi eduskunnan hakutoiminnolla varustetun tietokannan rakentaminen eduskunnan keskustelupöytäkirjoista on journalistisesti kiinnostavaa, mitä osin samaan tarpeeseen vastaavia aineistoja ja palveluja on entuudestaan saatavilla, sekä miksi pidän tarpeellisena rakentaa kokonaan uuden palvelun tarkoitusta varten.

4.1.1 Nykytilan kartoitus, ongelman havaitseminen ja määrittely

Kaikki täysistunnoissa käydyt keskustelut eduskunnan perustamisesta vuonna 1907 alkaen on julkaistu painettuina pöytäkirjoina. Pöytäkirjat ovat saatavissa myös eduskunnan sivuilta sähköisessä muodossa, mutta aineistot ovat osin hankalasti käytettäviä, sillä niiden tallennustapa ja muoto vaihtelee aineiston alkuperäisen tuottamisajankohdan mukaan, tekninen laatu on vaihteleva ja aineiston kuvailutiedot puuttuvat (Hyvönen et al. 2021).

Vuosien 1907–2000 aikana julkaistut pöytäkirjat ovat ladattavissa eduskunnan sivuilta vain skannattuna pdf-muodossa. Aineistot on skannattu Canon DR-G1100 -

dokumenttiskannerilla 300 dpi:n[2] tarkkuudella ja tekstintunnistettu Adobe Acrobat X -ohjelmalla. Skannauslaatu vaihtelee: painolaatu on heikoin 1920–30-lukujen pöytäkirjoissa, mikä heijastuu myös skannattujen aineistojen ja tekstintunnistuksen laatuun. (Apilo 2018.) Myös tätä vanhempien pöytäkirjojen tekninen laatu on heikompi kuin myöhempien aineistojen (Simola 2020). Vuodesta 2015 alkaen aineistot ovat saatavissa korkealaatuisessa digitaalisessa muodossa, mutta vuosien 2001–2014 aineistoja ei ole eduskunnan avoimen latauspalvelussa tällä hetkellä saatavilla lainkaan. (Eduskunta 2018a.) Myös näiden vuosien keskustelupöytäkirjat ja muut valtiopäiväasiakirjat ovat kyllä saatavissa eduskunnan verkkosivujen hakupalvelun (Eduskunta s.a.-b) kautta, mutta niiden kerääminen onnistuu vain käsityönä tai tiedonharavointia hyödyntäen.

Eduskunnan verkkosivuilla on Microsoftin SharePoint-ohjelmistolla toteutettu hakutoiminto, jonka avulla keskustelupöytäkirjoihin merkittyjen keskustelujen sisällöstä on mahdollista tehdä hakuja (Eduskunta s.a.-b). Johtuen tekstintunnistuksen vaihtelevasta laadusta myös hakutulokset etenkin sotia edeltävän aikakauden pöytäkirjoista ovat puutteellisia ja sekavia. Esimerkiksi haku sanalla ”kieltolaki” tuottaa vain 624 tulosta, vaikka rakentamani tietokannan perusteella sana esiintyy täysistuntojen pöytäkirjoissa todellisuudessa yli 1 600 kertaa. Vanhin hakutulos tällä hakusanalla on täysistunnosta 2.8.1917, vaikka termi on todellisuudessa mainittu asiakirjoissa n. 350 kertaa tätä aiemminkin. Tekstintunnistuksen laatu on myös heikko. Esimerkki 2.8.1917 täysistunnon pöytäkirjasta:

> *Ed. He 1e llli u s- S e:p p älä: \\lainitussa ve-*
>
> *romietinn~ssä, jonka ed~skunta. on hy\Tfi.~synyt, on asetuttu mmenomaan s1lle kannalle, että: puhe-ena- oleva väliaikainen kieltolaki astuisi ,jo varsin pian"voimaan. Nytkuitenkinjokainenedus~n nalll jäsen tietälllee, ehkäpä sentään ed. Sopanen~m, että s·en jälkeen, kun tuo päätörs Suomen korkeim- man valtiovrullan käyttämisestä, jonka hänkin mainitsi, on eduskunnassa tehty, on luultavaa - ja tl.edetään se jokseenkin varmasti - että väliaikaiselta hallitukselta eivät rpalaa senaatin sinne lä-*
>
> *hettämät ehdotukset eduskunnaUe annettaviksi esityksiksi.*

[2] DPI on lyhenne sanoista *dots per inch*, pistettä tuumalla. Se on painotekniikassa käytetty mittayksikkö kuvantarkkuudelle eli resoluutiolle.

Teknisen laadun puutteiden lisäksi haun käyttöä rajoittava tekijä on se, että hakutuloksien rajaus ja suodattaminen on hankalaa tai mahdotonta. Aikarajaustoiminto mahdollistaa rajauksen vain yhteen vuoteen kerrallaan ja rajaus esimerkiksi kansanedustajan sukupuolen, eduskuntaryhmän tai vaalipiirin perusteella ei onnistu lainkaan. Lisäksi haussa on suoranaisia bugeja: esimerkiksi usean villikorttikriteerin yhdistäminen ”tai”-operaattorilla hakurajauksissa tuottaa hakuehdoista riippumatta noin 1,6 miljoonaa tulosta – ilmeisesti siksi, että tuloksiin vaikuttaa päätyvän mukaan kaikki tekstit, joissa esiintyy jokin kirjaimilla ”or” alkava sana.

4.1.2 Vaihtoehtojen etsiminen ja arviointi

Eduskunnan omat sivut eivät ole ainut paikka, josta eduskunnan täysistuntojen keskustelut voi löytää sähköisessä muodossa. Hyvönen ynnä muut (2021) ovat kuvanneet kattavasti eduskunnan keskusteluista Suomessa tehtyä akateemista tutkimusta ja sen tuotteena syntyneitä kielitieteellisiä aineistoja.

Suurin osa tutkimuksesta keskittyy uudempiin, vuoden 2007 jälkeisiin keskusteluihin, jotka ovat jo valmiiksi saatavissa korkealaatuisina digitaalisina aineistoina eduskunnan sivuilta tai suomalaisten yliopistojen ja tutkimusorganisaatioiden yhteisestä Kielipankki-palvelusta (FIN-CLARIN 2015). Eduskunnan vuosien 2008–2016 täysistuntojen keskustelut on tallennettu Kielipankin Eduskuntakorpukseen, joka sisältää kattavat kuvailutiedot ja linkit alkuperäisiin täysistuntovideoihin tekstiin kohdistettuna (Hyvönen et al. 2021; FIN-CLARIN 2015). Muutamissa tutkimushankkeissa on luotu myös omia, kattavuudeltaan laajempia korkealaatuisia sähköisiä aineistoja, jotka ovat myös potentiaalisesti hyödynnettävissä journalistisessa käytössä.

Voices of Democracy -hanke on tuottanut tutkimuskorpuksen, joka sisältää kattavasti annotoidut täysistuntojen pöytäkirjat vuosilta 1980–2018. (Hyvönen et al. 2021). Korpuksen verkkosivu ei kuitenkaan tämän kehittämistyön kirjoitusaikaan ollut toiminnassa ainakaan tutkimusryhmän artikkelissa (Andrushchenko et al. 2021) mainitussa osoitteessa. Vuosien 1991–2015 pöytäkirjat on tallennettu kansainväliseen Harvard Parlspeech -korupukseen, mutta tämän korpuksen kattavuus on Hyvösen ynnä muiden (2021) mukaan puutteellinen.

Salla Simola on julkaisematonta työpaperiaan (2020) varten luonut tutkimusaineiston, joka sisältää kaikki eduskunnassa 1907–2018 käydyt keskustelut. Tätä aineistoa ei valitettavasti ole kuitenkaan toistaiseksi julkaistu. Koska Simolan laatima aineisto vastaa monin tavoin tässä kehittämistyössä koostamaani digitoitua aineistoa, palaan siihen vielä lähemmin tuonnempana, alaluvussa 4.3.2.

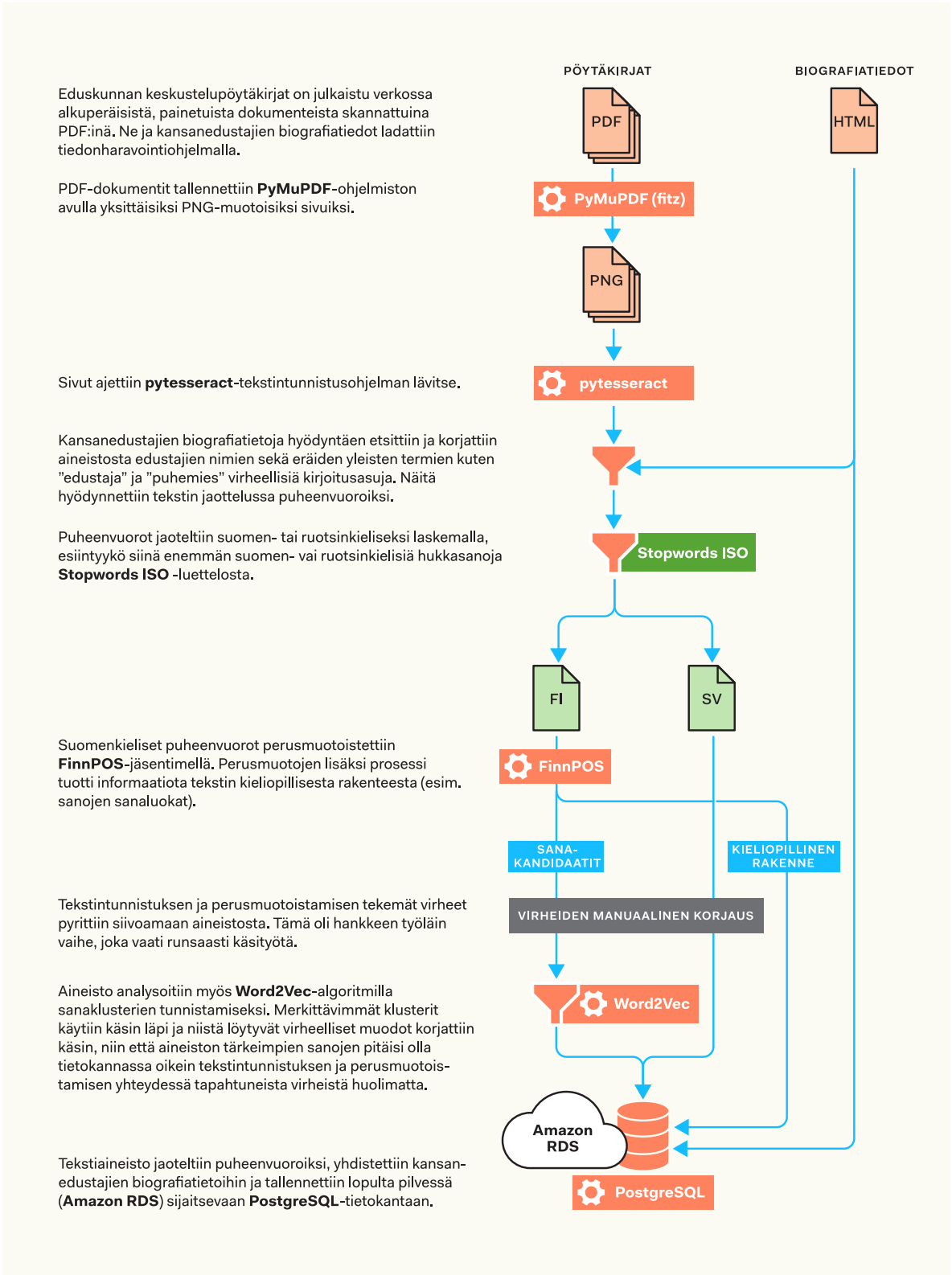
Tutkijoiden lisäksi myös vapaaehtoispohjalta toimivat aktiiviset kansalaiset ovat luoneet verkkopalveluita, joiden kautta eduskunnan keskustelut olisivat helpommin saavutettavissa.

Kansan muisti oli saman nimisen yhdistyksen vuonna 2010 ylläpitämä verkkopalvelu, joka seurasi eduskunnassa käytyjä keskusteluja, äänestystuloksia, tehtyjä esityksiä, aloitteita ja kysymyksiä, sekä mm. kansanedustajien sosiaalisen median profiileja (Kansan muisti ry. 2010; Forum Virium 7.4.2015). Helppokäyttöiseksi ja selkeäksi kuvatusta (esim. Rantanen 17.3.2015) palvelusta julkaistiin uusi versio vuonna 2015 ja sen tiedot olivat myös journalistien ja muiden kiinnostuneiden käytettävissä avoimen rajapinnan kautta (Forum Virium 7.4.2015). Sivuston perustamisen aikaan eduskunnalla ei vielä tarjonnut rajapintaa, jonka kautta keskustelut ja muut asiakirjat olisi pystynyt rakenteisessa muodossa lataamaan, joten palvelu oli toteutettu tiedonharavointi-menetelmällä. Vaikka verkkosivu on edelleen olemassa, se ei enää päivity, sillä eduskunta teki verkkosivu-uudistuksen pian uusitun version julkistamisen jälkeen, jolloin haravointiohjelma lakkasi toimimasta. (Yrjölä 4.1.2021.) Varhaisimmat sivulta löytyvät keskustelut ovat vuodelta 1999 ja viimeisin täysistunto, jonka tiedot sivulta edelleen löytyvät oli 14.3.2015 (Kansan muisti ry. 2010).

Govt.fi on palvelu, joka näyttää eduskunnan 2016 käyttöön otetun rajapinnan kautta saatavilla olevat tiedot hieman eduskunnan omaa sivustoa helppokäyttöisemmässä käyttöliittymässä. Keskustelupöytäkirjat eivät kuitenkaan ole sivuston kautta saatavilla olevien aineistojen joukossa. (Inha 2021)

Aalto-yliopiston tietotekniikan laitoksella toimiva Semanttisen laskennan tutkimusryhmä (SeCo), Helsingin yliopiston digitaalisten ihmistieteiden keskus (HELDIG) ja Turun yliopiston eduskuntatutkimuksen keskus kehittävät Parlamenttisampo-nimistä, semanttisen webin teknologioille perustuvaa palvelua, joka yhdistää eduskunnan tuottamiin aineistoihin kansanedustajien biografiatietoja ja sosiaalisen median tietoja samaan tapaan kuin edellä kuvattu Kansan muisti -palvelu. Se hyödyntää kuitenkin virallisia rajapintoja ja semanttisen webin teknologioita, joten sen toimivuus on pitkällä tähtäimellä turvatumpi kuin Kansan muistilla, joka kaatui eduskunnan verkkosivu-uudistukseen. Palvelu on määrä julkistaa vuoden 2022 aikana. (SeCo 2021)

4.2 Aineiston kerääminen ja muuttaminen koneluettavaan muotoon



Kuva 4. Yleisesitys kehittämistyöni aineiston keräämisen, analysoinnin ja tietokantaan tallettamisen työvaiheista ja niissä käytetyistä tekniikoista (ks. alaluvut 4.2.1–4.4.2).

Kuvailen seuraavassa, miten keräsin aineistona käyttämäni täysistuntojen keskustelupöytäkirjat ja kansanedustajien biografiatiedot sekä miten niitä käsittelin. Prosessi oli luonteeltaan vapaamuotoinen ja iteratiivinen eli samoja työvaiheita useasti toistava. Eri työvaiheita työstettiin osin rinnakkain ja usein myöhemmässä vaiheessa huomatut ongelmat johtivat prosessissa taaksepäin palaamiseen. Nämä työvaiheet toteutettiin kokonaisuudessaan Python-ohjelmointikielen avulla.

4.2.1 Aineiston kerääminen tiedonharavoinnin avulla

Kehittämistyön produktiivisen osan ensimmäisenä työvaihe oli pdf-muodossa julkaistujen eduskunnan keskustelupöytäkirjojen ja kansanedustajien biografiatietojen kerääminen. PDF-muodossa julkaistuja suomen- ja ruotsinkielisiä pöytäkirjoja on yhteensä 411 kappaletta ja eduskunnan sivuilta löytyy 2 605 kansanedustajan tiedot, joten tietojen kerääminen käsityönä yksi kerrallaan olisi ollut varsin työlästä. Niinpä päätyin kirjoittamaan haravointiohjelmat, jotka hakivat haluamani tiedot automaattisesti.

Kanadalainen journalisti Nael Shiab (2015) on pohtinut tiedonharavoinnin etiikkaa journalistisena tiedonhankintakeinona. Hän tekee eron haravoinnin ja hakkeroinnin välille: ”Toimittajien ei tulisi yrittää urkkia suojattua dataa. Jos tavallinen käyttäjä ei pääse tietoon käsiksi, ei journalistienkaan tulisi sitä yrittää saada käsiinsä.” Shiab kehottaa myös kunnioittamaan mahdollisia sivuston ylläpitäjän asettamia rajoituksia tietojen haravoinnille, jotka ilmoitetaan vakiintuneen käytännön mukaisesti verkko-osoitteen juurihakemistoon sijoitetussa robots.txt-nimisessä tiedostossa. (Shiab 2015.) (Eduskunnan sivuilla ei ole robots.txt-tiedostoa, joka asettaisi rajoitteita tiedonharavoinnille.)

Shiabin haastattelemat toimittajat ovat erimielisiä siitä, tulisiko journalistin jollain tapaa ilmaista olevansa journalisti tiedonharavoinnin yhteydessä (Shiab 2015). Suomalaisenkin journalismin eettisten ohjeiden mukaan ”journalistin on suositeltavaa ilmoittaa ammattinsa” tietoja hankkiessaan (JSN 2011) ja samalla linjalla on osa Shiabin haastateltavista. Esimerkiksi Ottawa Citizen -sanomalehden Glen McGregor kertoo lisäävänsä haravointiohjelmansa lähettämiin http-otsikkotietoihin yhteystietonsa ja kehotuksen ottaa yhteyttä, mikäli asiassa on ongelma. L’Actualité-sanomalehden Philippe Gohier taas sanoo mieluummin esiintyvänsä anonyyminä tietoja kerätessään. (Shiab 2015.) Koska itse toimin tätä kehittämistyötä varten tietoja kerätessäni opiskelijan enkä toimittajan roolissa, katsoin, että yhteystietojen lisääminen haravointiohjelman lähettämiin tietoihin ei tässä tapauksessa ollut tarpeen. Kerätessäni tietoja ammatillisiin tarkoituksiin noudatan kuitenkin McGregorin kuvailemaa käytäntöä.

Varsin yleinen eettinen käytäntö on myös käyttää esimerkiksi 2 sekunnin mittaista viivettä sivulatausten välissä palvelimen ylenmääräisen kuormittamisen välttämiseksi. Tällä on myös käytännöllinen merkitys, sillä monilla servereillä on käytössä palvelunestohyökkäysten torjuntaan tarkoitettu suojausohjelmisto, joka estää samasta verkko-osoitteesta tulevat pyynnöt, jos niitä lähetetään liian nopeaan tahtiin.

Edellä kuvattuja eettisiä periaatteita seuraten rakensin tiedonharavointiohjelmat eli ns. skreipperit pöytäkirjojen ja biografiatietojen keräämistä varten. Yksinkertaisimmillaan haravointiohjelma voisi toimia niin, että se hakee sivun html-muotoisen lähdekoodin esimerkiksi Python-ohjelmointikielen requests-moduulilla ja kerää halutut tiedot sitä analysoimalla. Eduskunnan sivut on kuitenkin toteutettu ns. verkkosovelluskehyksellä (*web framework*), jossa verkkosivu rakentuu vasta käyttäjän selaimessa eikä lähdekoodi sisällä kaikkea sivun teksti- tai muuta sisältöä. Niinpä kummankin haravointiohjelman toteuttamisessa hyödynnettiin Pythonin selenium-moduulia, joka on Selenium WebDriver -ohjelmistoon perustuva selainautomaatiokirjasto. Se mahdollistaa Chromen tai muiden verkkoselainten käyttämisen ohjelmallisesti ja selaimen muistiin tallentuvien tietojen lukemisen.

Pöytäkirjat keräävä haravointiohjelma on varsin suoraviivainen. Linkit kaikkiin dokumentteihin löytyvät samalta sivulta hakuvalintojen takaa, joten skreipperi käy yksinkertaisesti läpi digitoitujen valtiopäiväasiakirjojen luettelon (Eduskunta 2018b) vuosikymmen vuosikymmeneltä ja vuosi vuodelta, etsii kaikki ”Pöytäkirjat”-hakuvalinnalla löytyvät dokumentit ja lataa ne. Kansanedustajien biografiatiedot keräävä haravointiohjelma joutuu lisäksi huomioimaan, että kansanedustajahaku (Edustunta s.a.-a) näyttää vain 20 kansanedustajaa kerralla. Näin ollen skreipperi joutuu käymään läpi suuren joukon tulossivuja sekä vielä varsinaiset kansanedustajien tiedot sisältävät sivut. Kun joka sivulatauksen jälkeen pidetään 2 sekunnin tauko, kansanedustajien tietojen lataaminen kestää ohjelmalta pari tuntia.

4.2.2 Tekstintunnistus

Jotta alkujaan paperille painetut pöytäkirjat olisivat käytettävissä laskennallista analyysiä varten ja siirrettävissä tietokantaan ne pitää ensin muuttaa koneluettavaan merkkimuotoon. Skannatut pöytäkirjan sivut ovat *kuvia* tekstistä. Vaikka näkevän ihmisen kannalta eri tallennus- ja esitystavoilla ei ole tavallisimmissa käyttötilanteissa suurtakaan käytännön eroa, tietokoneen – tai ruudunlukijaa käyttävän näkövammaisen ihmisen – kannalta ei ole lainkaan yhdentekevää, esitetäänkö tieto kuvana vai merkkimuodossa eli yksittäisinä merkkeinä.

Skannattu teksti muutetaan merkkimuotoon käyttämällä optista tekstintunnistusta, josta käytetään usein englanninkielistä lyhennettä OCR (*optical character recognition*). Tämä on menetelmä, jossa tavallisesti koneoppimista hyödyntäen tunnistetaan kuvissa esiintyvät tekstit ja muutetaan ne koneluettaviksi merkkijonoiksi.

Eduskunnan sivuilta ladattavissa pdf-tiedostoissa on mukana Adobe Acrobat X -ohjelmalla tehty tekstintunnistus (Apilo 2018). Valitettavasti tekstintunnistuksen laatu on kuitenkin hyvin vaihteleva. Tässä esimerkki pääministeri Cajanderin puheesta eduskunnassa 30.11.1939 Neuvostoliiton aloitettua sotatoimet Suomea vastaan:

> *Kuluneina viikkoirua on hallitus ol~ut jatkuvasti kooketuksessa ,eduskunnan herra puhemiehen ja eduskuntaryhmien puheenjohtajien ja heidän kauttansa eduskuntaryhmien kanssa. Tilante,en kehittyessä nykyiselle asteelleen hallitus kuitenkin katsoo selkä asianmukaiseksi että velvollisuudekseen siinä .t:arlmituksessa ja järjestyksessä kuin valtiopäiväjärjestyksen 36 § edelil.yttää, antaa eduskunnalle kokonaisuudessaan tiedonannon Suomen suhteista Sosialistisien Neuvostotasavaltain Liittoon ja tulee herra ulkoasiainministeri antamaan asiasta yksityiskohtaisen seiostuksen. Siitä on näkyvä, ,että hallitus on kaiken aikaa tehnyt voitavansa päättävästi puolustaakseen Suomen etua ja oikeUJtta. Samalla on oltu valmiit sovittel,emaan, missä se 0111 ollut mahdollista.*

Vaikka laatu on osassa aineistoa selvästi tätä parempikin, pistokoeluontoisesti tutkittuna tekstintunnistuksen taso oli kuitenkin kokonaisuutena arvioni mukaan liian heikko ainakin vanhemmissa pöytäkirjoissa, jotta merkkijonoksi muutettu teksti olisi sellaisenaan käyttökelpoista. Niinpä päädyin toteuttamaan tekstintunnistuksen aineistoon itse. Useiden iteraatioiden jälkeen päädyin lopulta seuraavanlaiseen tekniseen ratkaisuun:

Ohjelma käy läpi kaikki pöytäkirjat ja tallentaa jokaisen sivun vuorollaan png-muotoiseksi kuvaksi Pythonin PyMuPDF-moduulin avulla. Kuva syötetään pytesseract-tekstintunnistusmoduulille, joka hyödyntää Hewlett Packardin ja Googlen kehittämää Tesseract OCR -kirjastoa (Tesseract OCR 2015) ja tekstintunnistuksen läpikäynyt aineisto tallennetaan lopulta pöytäkirjoittain ilmitekstitiedostoiksi (*plain text file*).

Vaikka alkuperäiset pdf:t on skannattu 300 dpi:n tarkkuudella, kuvien tallentaminen alkuperäistä korkeammalla tarkkuudella vaikutti parantavan kokeiluissa hieman tekstintunnistuksen laatua. Suurempi tarkkuus kuitenkin myös hidastaa tunnistusta: karkeasti ottaen kuvan tarkkuuden tuplaaminen nelinkertaistaa siihen kuluvan ajan. Päädyin lopulta käyttämään alkuperäistä 300 dpi tarkkuutta, sillä pieni laadun parantuminen ei ollut mielekkäässä suhteessa kasvaneeseen laskentatehovaatimukseen.

Kun pytesseractille vielä kerrotaan, että dokumentit ovat suomenkielisiä, tekstintunnistus tuottaa kohtuullisen hyviä tuloksia. Tässä edellä esimerkkinä käytetty pätkä pääministeri Cajanderin puheesta Tesseractin avulla merkkimuotoon muutettuna:

> *Kuluneina viikkoina on hallitus ollut jatkuvasti kosketuksessa eduskunnan herra puhemiehen ja eduskuntaryhmien puheen johtajien ja heidän kauttansa eduskuntaryhmien kanssa. Tilanteen kehittyessä nykyiselle asteelleen hallitus kuitenkin katsoo sekä asianmukaiseksi että velvollisuudekseen siinä tarkoituksessa ja järjestyksessä kuin valtiopäiväjärjestyksen 36 $ edellyttää, antaa eduskunnalle kokonaisuudessaan tiedonannon Suomen suhteista Sosialististen Neuvostotasavaltain Liittoon ja tulee herra ulkoasiainministeri antamaan asiasta yksityiskohtaisen. selostuksen. Siitä on näkyvä, että hallitus on kaiken aikaa tehnyt voitavansa päättävästi puolustaakseen Suomen etua ja oikeutta. Samalla on oltu valmiit sovittelemaan, missä se on ollut mahdollista.*

Vaikka tekstissä on edelleen pieniä virheitä, tekstintunnistuksen laatu on selvästi edellistä esimerkkiä parempi. Vaikka OCR-laatu edelleen vaihtelee ja on joissain dokumenteissa selvästi huonompi kuin tässä esimerkissä, pistokoeluontoisesti testattuna se on *suomenkielisissä* teksteissä kautta linjan parempi, kuin alkuperäisessä pdf:ssä.

Ongelmaksi tosin muodostuu, että pöytäkirjoissa on sekä suomen- että ruotsinkielisiä tekstejä. Tesseract ei suoriudu yhtä hyvin ruotsinkielisen tekstin muuttamisesta merkkimuotoon kun käytössä ovat suomen mukaiset kieliasetukset. Esimerkkinä RKP:n pätkä kansanedustaja Ole Wasz-Höckertin puheenvuorosta täysistunnossa 26.3.1985:

> *För mängen beslutsfattare, säväl pä det kommunala som rikspolitiska planet, kommer en noggrann genomläsning av den nu framlagda redogörelse att klargöra dagens situation inom hälsovärdspolitiken. Man kan ur redogörelsen utläsa hur utvecklingen av vät hälso- och sjukvärd inriktat sig pä och anpassat sig efter de sjukdomsgrupper som under olika skeden drabbat landets befolkning.*

Tässä sama sitaatti alkuperäisen pdf:n OCR-aineistosta:

> *För mången beslutsfattare, såväl på det kommunala som rikspolitiska planet, kommer en noggrann genomläsning av den nu framlagda re- dogörelse att klargöra dagens situation inom häl- sovårdspolitiken. Man kan ur redogörelsen utläsa hur utvecklingen av vår hälso- och sjukvård inrik- tat sig på och anpassat sig efter de sjukdomsgrup- per som under olika skeden drabbat landets be- folkning.*

Systemaattisina ongelmina ovat varsinkin Q- ja Å-kirjainten korvautuminen G- ja Ä-kirjaimilla. Myös monien kansanedustajien nimissä esiintyvä É-kirjain muuttuu usein esim. numeroksi 6. Paikoin Tesseract ei myöskään ymmärrä, että sivun teksti on kaksipalstaista ja lukee eri palstojen vierekkäiset rivit yhdeksi riviksi, joka sotkee sisällön pahoin.

Painolaadultaan huonommista aineistoista Tesseract suoriutuu väärillä kieliasetuksillakin silti paremmin kuin eduskunnan pdf:ien tekstintunnistukseen käytetty Adobe Acrobat X. Tässä pätkä kansanedustaja Vilhelm Rosenqvistin puheenvuorosta kieltolakiehdotuksen käsittelystä 18.2.1909 Tesseractilla käsiteltynä:

> *Det synes mig, att -någon egentlig diskussion om denna fräga är fullkomligt onidig. Det gäller nti hvad som är det naturliga, det Täitipiga och det passande; och meningarna i detta af- seemide tyckas ju vara olika. Da saken en gäng blifvit "hänskjuten till landtdagen, föreslär jag, att vi utan vidare mäite skrida till omröstning. För egen del förordar jag herr talmannens förslag.*

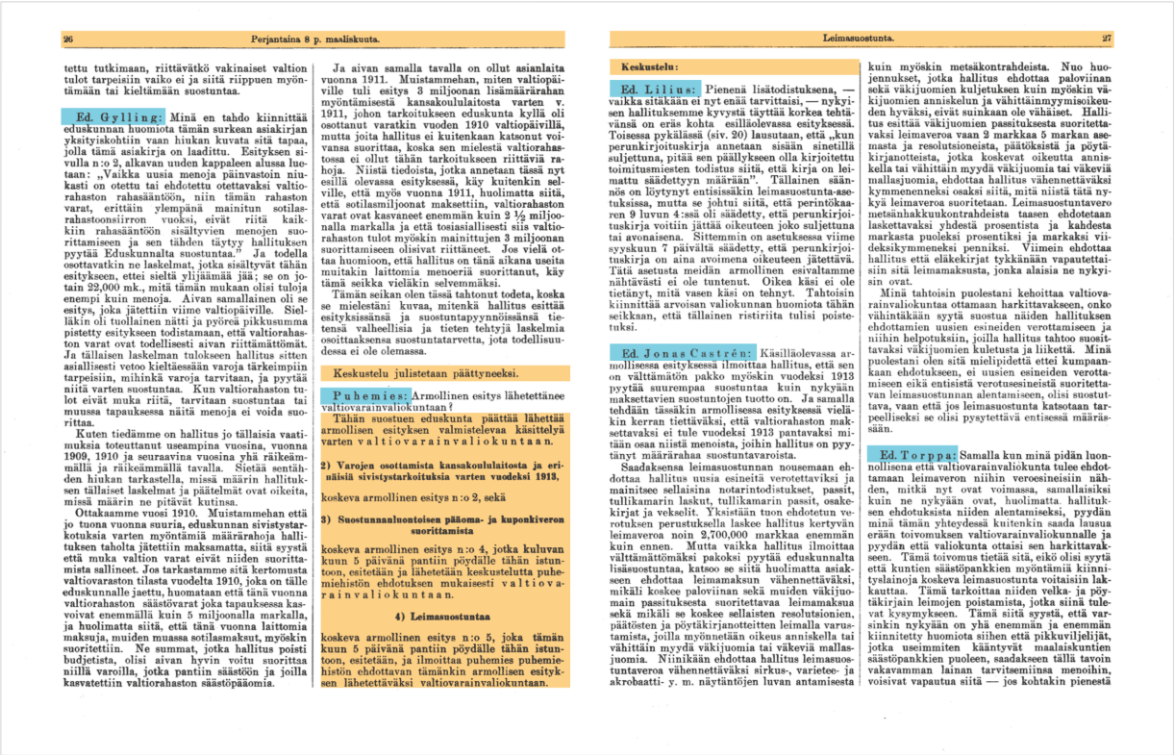
Sama PDF:ssä Acrobatin tekstintunnistuksen jäljiltä:

> *Detsynesmig,att ·någcm egenflig diskussion om denna fråga är fullkomligt .dfWdig. Det gäller nu hvad som är det naturliga, det låt11pliga och det passande; och meningarna i detta af- set!t'lde tyckas ju vara oHka. Då saken en gång blifvit hartskjuten tilllandtdagen, föreslår jag, att vi utan vidare n'Jfrtte skrida till omröstning. För eg·en del förordar jag herr talmannens förslag.*

Parhaan tuloksen saamiseksi aineisto olisi luultavasti aiheellista ajaa pytesseract-moduulin läpi kahdesti: kerran suomen ja kerran ruotsin kielen mukaisilla asetuksilla ja yhdistää tulokset. Tätä ei valitettavasti ole ollut mahdollistaa tämän kehittämistyön puitteissa, johtuen varsinkin tekstintunnistuksen vaatimasta pitkästä laskenta-ajasta. Tekstintunnistusprosessi oli teknisesti sinänsä suoraviivainen, mutta laskennallisesti varsin raskas. Jätettyäni pois ruotsinkieliset pöytäkirjat, jotka sisältävät vain asiakirja-aineistoa, ei varsinaista keskustelua (Puttonen et al. 2012), sain puristettua ympäri vuorokauden raksuttaneella late 2013 -mallin 27″ iMac-pöytäkoneella koko aineiston tekstintunnistuksen vaatiman ajan noin **kolmeen viikkoon**. (Aineiston ensimmäinen läpimeno vielä pari viikkoa pitempään, koska käytin aluksi 400 dpi:n resoluutiota png-kuvissa ja ruotsinkieliset aineistot olivat mukana.) On tosin syytä huomioida, että prosessointiaikaa olisi helposti lyhennettyä murto-osaan tästä hyödyntämällä useampaa prosessoriydintä samanaikaisesti tai vielä mieluummin suorittamalla laskennan pilvipalveluna rinnakkaislaskentana. Oman osaamiseni asettamien rajoitteiden vuoksi käytössä oli kuitenkin vain yksi iMacin moniytimisen prosessorin ytimistä, eli koneen maksimitehosta oli käytössä vain murto-osa.

4.3 Aineiston analysointi ja järjestäminen

Aineiston keruu- ja käsittelyvaiheen ylivoimaisesti vaikeimmaksi ja työläimmäksi osuudeksi osoittautui tekstintunnistuksen läpi käyneen materiaalin (ilmitekstitiedostot) analysointi ja järjestäminen siten, että kansanedustajien puheenvuorot eroteltiin muusta tekstistä, esimerkiksi äänestystuloksista ja kirjallisista kysymyksistä, ja pyrittiin tunnistamaan, kuka kansanedustaja kulloinkin on äänessä.



Kuva 5. Esimerkki keskustelupöytäkirjan aukeamasta. Puhujien nimet on korostettu sinisellä ja oranssin sävyllä muut sellaiset tekstit, jotka eivät ole keskustelua.

4.3.1 Keskustelun erottaminen muusta aineistosta

Yhtenä keskeisimpänä haasteena aineiston käsittelyssä on erotella, mitkä rivit pöytäkirjoissa ovat keskustelua ja mitkä jotakin muuta, kuten esimerkiksi äänestystuloksia tai kirjallisia kysymyksiä. Tätä hankaloittavat satunnaiset painovirheet, skannauksessa ja tekstintunnistuksessa tapahtuneet virheet sekä satunnaistekijät kuten esimerkiksi uuden kappaleen alkaminen täsmälleen sivun ylälaidasta, jolloin uutta puheenvuoroa ei edellä tyhjää riviä, jota voisi käyttää indikaattorina siitä, että puhuja vaihtuu. Lisäksi on huomioitava eduskunnan pöytäkirjanpitokäytännöissä tapahtuneet muutokset.

Salla Simola, joka on tutkimustaan varten koostanut koneluettavan aineiston eduskunnan keskustelupöytäkirjoista vuosilta 1907–2018, kuvaa julkaisemattomassa työpaperissaan (2020) käyttämänsä menetelmät keskusteluosuuksien ja puhujien tunnistamiseen eduskunnan skannatuista keskustelupöytäkirjoista. Simola on työssään ratkonut osin täsmälleen samoja ongelmia kuin itse olen kehittämistyössäni tehnyt, mutta hänen artikkelinsa ei ole valitettavasti ollut käytettävissäni kehittämistyön produktiivista osiota tehdessäni, sillä se on valmistunut vasta loppuvuodesta 2020, kun itse painin näiden ongelmien kanssa edeltävänä kesänä. Simolan valitsemat menetelmät, työssä kohtaamat haasteet sekä hänen ratkaisunsa niihin vastaavat kuitenkin joiltain osin omiani, joskin hän on toiminut itseäni järjestelmällisemmin ja soveltanut laskennallisia ratkaisuja myös

joihinkin sellaisiin ongelmiin, joita olen itse lähtenyt ratkomaan käsityönä. Sitä, että olemme päätyneet toisistamme tietämättä osin samoihin ratkaisuihin aineiston käsittelyssä voi kuitenkin mielestäni pitää epäsuorana validaationa sille, että valitsemani menetelmät ovat ainakin näiltä osin periaatteessa mielekkäitä. Vaikka oma kehittämistyöni vastaa osin samoihin haasteisiin kuin Simolan työpaperi, suosittelen silti lämpimästi siinä kuvattuihin menetelmiin lähemmin tutustumista, mikäli lukija harkitsee käyttävänsä kehittämistyöni havaintoja pohjana oman eduskunnan keskustelupöytäkirjojen haku-, analysointi- tms. työkalunsa kehittämiseen.

Lopputuloksen kannalta uskoakseni kaikkein merkittävin ero Simolan ja oman lähestymistapani välillä on, että hän on jakanut prosessin vaiheisiin siten, että hän on ensin tunnistanut pöytäkirjojen keskustelua sisältävät osiot ja vasta sen jälkeen lähtenyt pilkkomaan niitä yksittäisiksi keskusteluiksi (Simola 2020), kun taas itse olen edennyt kunkin pöytäkirjan läpi systemaattisesti rivi riviltä ja pyrkinyt arvioimaan, kuuluuko rivi keskusteluun, korjaten samalla myös joitakin tekstintunnistuksessa tehtyjä virheitä. Tämä on vaatinut tilapohjaista (*state-based*) lähestymistapaa, jossa kullakin hetkellä pidetään muistissa keskeiset tiedot esimerkiksi siitä, onko kyseessä puheenvuoro, kuka kulloinkin äänessä olevasta puhuja on jne. Jälkikäteen arvioiden valitsemani lähestymistapa on huomattavasti alttiimpi virheille, sillä yksittäinen virheellisesti tulkittu tai huomaamatta jäänyt ”tilan” muutos – esimerkiksi vaihdos keskustelusta muuhun sisältöön tai puhujan vaihdos – kertautuu nopeasti ja aiheuttaa virheellisiä tuloksia myös seuraavissa osioissa. Kun tarkastellaan lopputulosta, nähdäänkin, että koodini on tosiaan pudottanut pois merkittävän osan keskusteluista virheellisesti pois (ks. alaluku 5.1.1). Valitettavasti tämän ongelman laajuus on valjennut minulle vasta produktiivisen osan loppupuolella, joten sen korjaaminen ei ole ollut enää mahdollista tämän kehittämistyön puitteissa.

Pyrin seuraavassa erittelemään aineiston keskeiset virheille altistavat tekijät, jotta samaa mahdollisesti myöhemmin yrittävät voisivat välttyä tekemästä samoja virheitä.

Sekä oman että Simolan (2020) käyttämän lähestymistavan ytimessä on ollut tunnistaa puheenvuorojen aloitusrivit etsimällä kaikki sellaiset rivit, jotka ovat kappaleen ensimmäisiä ja joissa esiintyy kaksoispiste. Itse olen ratkaissut puhujan tunnistuksen niin, että jos kaksoispistettä edeltävä osio rivistä alkaa ”Edustaja”, ”Ed.”, ”Puhemies”, ”Varapuhemies”, ”Ikäpuhemies”, ”Senaattori” tai sanalla, johon sisältyy ”ministeri”, tutkitaan tarkemmin, onko kyseessä todellakin uuden puheenvuoron alku, vai esimerkiksi vastauspuheenvuoroon sisältyvä viittaus edeltävään puhujaan (”Ed. Niemiselle sen sijaan sanoisin: …”) sitä pidetään kandidaattina uudeksi puheenvuoroksi ja koitetaan selvittää, vastaako osion sisältö jonkun ajankohtaan sopivan kansanedustajan nimeä.

Ennen kuin puhuja yritetään tunnistaa (ks. seuraava alaluku) kaksoispistettä edeltävästä osiosta poistetaan ilmaisut kuten ”(vastauspuheenvuoro)”, ”joka puheenvuoron saatuaan lausui”, ”sai puheenvuoron ja lausui”, ”-n puheenvuoro”, ”kysyy”, ”jatkaa”, ”toteaa” ym., joita varsinkin vanhimmissa pöytäkirjoissa usein esiintyy. Lisäksi olen tehnyt listan n. löytämistäni 450 sanasta tai fraasista, jotka esiintyvät jossain kohdin tekstiä edellä mainittujen alkutunnisteiden kuten ”Edustaja” ja kaksoispisteen välissä, mutta jotka eivät ole kansanedustajien nimiä.

Asiaa hankaloittavat satunnaiset painovirheet, skannausvaiheessa kriittiseen paikkaan osuneet roskat sekä tekstintunnistuksen tekemät virheet. Esimerkiksi lyhenteestä ”ed.” (edustaja) olen löytänyt aineistosta 29 erilaista virheellistä muotoa: ”ed ”, ”ed-”, ”fd”, ”hd”, ”kd”, ”rd”, ”fa.”, ”fa ”, ”fä ”, ”fg.”, ”ea.”, ”eä.”, ”eä ”, ”td.”, ”bd”, ”md”, ”pd”, ”od.”, ”id.”, ”tid.” ”edä”, ”fed”, ”fdä”, ”eid”, ”edi ”, ”f4.”, ”f4d.”, ”fidd.” ja ”fida”. Erilaisia sanan ”puhemies” virheellisiä muotoja olen löytänyt 83. Myös lukuisat muut virheet, epäjohdonmukaisuudet ja vaihtelevat käytännöt hankaloittavat tehtävää. Olen pyrkinyt ratkaisemaan tällaiset ongelmat siten, että kutakin koodiversiota on testattu joukolla keskustelupätkiä, katsottu pystytäänkö jokaiselle tunnistetulle puheenvuorolle tunnistamaan myös puhuja (ks. seuraava alaluku) ja mikäli ei, muokattu koodia, kunnes kaikki näin havaitut virheet ja puutteet on saatu korjattua. Tätä on iteroitu siihen asti, kunnes koodi ei enää vaikuta uusia keskustelupätkiä analysoitaessa tuottavan virheellisiä tuloksia, vaan tekstipätkien puhujat pystytään pääosin tunnistamaan, tai tekstipätkät luokittelemaan ei-keskusteluksi. Epäsystemaattinen lähestymistapa on kuitenkin johtanut siihen, että koodi on monimutkaista ja täynnä harvinaisten poikkeustapausten ja virheiden korjaamisen vaatimia kömpelöitä ratkaisuja, mikä ei jälkeenpäin arvioiden ole paras tapa toimia, varsinkin kun erilaiset tiettyä ongelmaa varten kehitetyt *ad hoc* -ratkaisut saattavat aiheuttaa toisissa kohdin hankalasti ennakoitavia uusia virheitä. Simola (2020) on toiminut tässä suhteessa toisin ja pyrkinyt luomaan vain kourallisen johdonmukaisia sääntöjä, joilla saavutettavaa tulosta hän validoi suhteessa satunnaisesti valittuihin pöytäkirjoihin. Hän tyytyy siihen, että pääsääntöjä soveltamalla päästään yli 90 prosentin tarkkuuteen eikä edes lähde yrittämään harvinaisimpien erikoistapausten käsittelyä.

4.3.2 Keskustelijan tunnistaminen

Puheenvuorojen erottelun lisäksi keskeinen vaihe aineiston analysoinnissa ja järjestämisessä on sen tunnistaminen, kuka kansanedustaja kullakin hetkellä puhuu.

Vuoden 1999 valtiopäivien 2. pöytäkirjaan (istunnot 29.10. asti) saakka pöytäkirjoissa noudatetaan tiivistä ilmaisutapaa, jossa puhuva kansanedustaja tai ministeri ilmaistaan lähtökohtaisesti pelkällä sukunimellä (esim. ”Ed. Lehtimäki”, ”Ulkoministeri Hackzell”).

Tätä uudemmissa pöytäkirjoissa puhujien nimet kirjoitetaan virhetulkinnoille vähemmän alttiissa muodossa ”Koko nimi /eduskuntaryhmän lyhenne”. Vuosien 1989–2010 valtiopäiviltä on lisäksi tuotettu hakemisto, jossa on lueteltu kaikki kunkin kansanedustajan täysistunnoissa pitämät puheenvuorot (Puttonen et al. 2012), mutta koska näiden sinänsä huomattavasti selkeämpien puhujatietojen huomioonottaminen olisi vaatinut koodimuutoksia, päätin rajata tarkasteluni niihin vain dokumentteihin, joissa on käytössä vanha, tiiviimpi tapa, eli pöytäkirjoihin I/1907–II/1999.

Jos samoilla valtiopäivillä on ollut useita kansanedustajia, joilla on sama sukunimi, erotellaan samannimiset kansanedustajat pöytäkirjoissa etunimen ensimmäisen kirjaimen avulla (esim. ”Ed. M. Mattila”, ”Ed. V. Mattila”), tai jos myös etunimen ensimmäinen kirjain on sama, koko nimellä (esim. ”Ed. Jari Koskinen, ”Ed. Johannes Koskinen”). Muutaman kerran kansanedustajina on ollut samanaikaisesti myös täyskaimoja, jolloin pöytäkirjoissa heihin viitataan molempien/kaikkien etunimien alkukirjaimilla (esim. ”Ed. T. M. Koivisto”, ”Ed. T. T. Koivisto”). Vuosien 1907–08 pöytäkirjoissa etunimikirjain ilmoitettiin sukunimen jälkeen. Käytäntö vanhimmissa pöytäkirjoissa muutenkin hieman erilainen kuin myöhemmissä pöytäkirjoissa, esim. Oskari Laineeseen viitataan nimellä ”Ed. Laine, O.” ja hänen lähes-kaimaansa Oskar Laineeseen nimellä ”Ed. Laine, O. F.”.

Jos ei huomioida tekstintunnistuksessa tapahtuneita virheitä (joista lisää alla), useimpien kansanedustajien tunnistaminen onnistuu kohtuullisen helposti vertaamalla sukunimeä tai nimikirjain–sukunimi-yhdistelmää kansanedustajien biografiatietoihin, huomioiden siis eri kansanistujien toimikaudet ja suomen muuttunut oikeinkirjoitus, eli W-kirjaimen korvautuminen V:llä 1900-luvun alkupuolella. Jonkin verran haasteita syntyy kuitenkin kansanedustajien muuttuneista nimistä, sekä siitä, että kaikki eivät käytä kutsumanimenä ensimmäistä etunimeään. Muuttuneet sukunimet jakautuvat avioliiton tai avioeron vuoksi muuttuneisiin naiskansanedustajien nimiin, joissa useimmissa tapauksissa on kyse yhdistelmäsukunimen ottamisesta ja yhdistelmäosan pudottamisesta sukunimestä, sekä muihin sukunimimuutoksiin. Jälkimmäinen ryhmä koostuu pääosin fennomanian aikaisista sukunimien suomennoksista (esim. Serlachius → Särkilahti), mutta joukossa on myös pari esimerkkiä varhaisten vasemmistokansanedustajien vaihtelusta ns. sotanimien (*nom de guerre*) ja syntymänimien välillä (esim. Edvard Valpas, alkuperäiseltä nimeltään Hänninen – Valpas käytti välillä nimestään myös muotoa Valpas-Hänninen).

Olen koonnut taulukkoon liitteessä 1 ne pöytäkirjaan merkityt nimet, joissa kansanedustajan biografiatietoihin merkittyä nimeä ei ole voinut helposti suoraan päätellä edellä pelkästään etunimen alkukirjaimen, sukunimen ja toimikauden perusteella, vaan oikean kansanedustajan identiteetin selvittäminen on vaatinut enemmän salapoliisityötä.

315	**Helenelunmd**	Helenelund
316	**Helenmiwvs**	Helenius
317	**Heleniu s**	Helenius
318	**Helen ius**	Helenius
319	**Helemius**	Helenius
320	**Helemnius**	Helenius
321	**Holenius**	Helenius
322	**H e1o**	Helo
323	**H e1 o**	Helo
324	**H e 1 o**	Helo
325	**H elo**	Helo
326	**Belo**	Helo
327	**Helö**	Helo
328	**Hello**	Helle
329	**Häidenheimo**	Hiidenheimo
330	**Hiiddenheimo**	Hiidenheimo
331	**Hiidemheimo**	Hiidenheimo
332	**Hiidenbeimo**	Hiidenheimo
333	**Hiidenmheimo**	Hiidenheimo
334	**Hildön**	Hildén
335	**Hikdén**	Hildén
336	**Hildätn**	Hildén
337	**Hildéx**	Hildén
338	**Hilkdön**	Hildén
339	**Hielt**	Hjelt
340	**Honen**	Homén
341	**Homäen**	Homén
342	**Homäön**	Homén

Kuva 6. Esimerkkejä tekstintunnistuksen tuottamista kansanedustajien nimien virheellisistä muodoista.

Myös kansanedustajien nimissä esiintyy runsaasti tekstintunnistuksen, skannausroskien ja painovirheiden aiheuttamia virheellisiä muotoja. Olen koonnut aineistosta käsityönä yli 1 700 virheellistä muotoa, mutta tämä lukema on varmasti alakanttiin, sillä olen koostanut erikseen virheellisiä muotoja myös tekstintunnistukselle hankalista merkkijonoista, joita esiintyy useissa eri nimissä (esim. ”-backa”, ”-én”, ”-qvist”), jotka käydään läpi ennen kuin koodi alkaa etsiä yksittäisten nimien virheellisiä muotoja. Vaikka suhteellisesti eniten ongelmia on itsenäisyyden alkuvuosikymmenten painolaadultaan heikompien pöytäkirjojen teksteissä ja ruotsin- tai vieraskielisissä nimissä, varsinkin jos niissä esiintyy kirjaimia, joita ei suomenkielisissä sanoissa käytetä kuten C, É, F, Q, Z ja Å, myös uudemmissa aineistoissa esiintyy jonkin verran virheellisiä muotoja ja myös aivan tavallisista suomalaisista nimistä mahtuu mukaan virheellisiä kirjoitusasuja, jos ne esiintyvät aineistossa usein. Esimerkiksi nimen ”Vennamo” virheellisiä muotoja olen aineistosta löytänyt peräti 33 kpl – varmasti pitkälti siksi, että SMP:n Veikko Vennamo on yksi eniten eduskunnan historiassa äänessä olleista kansanedustajista.

Päädyin poistamaan aineistosta välihuudot, mikä onnistui pääosin helposti, sillä ne on pöytäkirjoissa kirjattu sulkeisiin. Joskus jompikumpi sulkeista kuitenkin puuttuu virheellisen tekstintunnistuksen vuoksi, asia muuttuu heti hankalammaksi. Myöskään jatkopuheenvuorojen – esim. jos puhemies keskeyttää ensin puhujan ja antaa tämän sitten jatkaa – yhdistäminen oikeaan puheenvuoroon ei yleensä onnistunut oikein, sillä jatkopuheenvuoron alussa ei pöytäkirjaan ole merkitty kuka puhuu. Asia ilmaistaan useimmiten ilmaisulla ”Puhuja jatkaa”, mutta välillä myös jollain muulla tavalla, esimerkiksi ”Senjälkeen pääministeri jatkaa” tai ”Puhuja jatkaa lavalta”. Sanapari ”Puhuja jatkaa” voi kaiken lisäksi esiintyä myös muissa konteksteissa, esimerkiksi puhemiehen kehotuksena tai pöytäkirjaan merkittynä toteamuksena, esim. ”Puhuja jatkaa edelleen puhettaan.”

Kun puheenvuorot on saatu erotettua muusta aineistosta ja puhujan nimi on saatu selville, ne tallennetaan TSV-muodossa[3]. Tiedoston sarakkeina ovat itse puheenvuoron lisäksi istunnon päivämäärä, puhujan nimi, etunimen perusteella päätelty sukupuoli[4], eduskuntaryhmä, asema (rivikansanedustaja vai esim. ministeri) ja sivunumero.

4.3.3 Perusmuotoistaminen ja sanaluokittelu suomen kielen jäsentimellä

Kun puheenvuorot on saatu erotettua muusta aineistoista ja järjestettyä puhujittain TSV-tiedostoiksi, puheenvuorot käydään vielä läpi suomen kielen jäsennintä hyödyntäen sanojen perusmuotojen ja kieliopillisten piirteiden selvittämiseksi. Samassa yhteydessä tekstistä poistetaan hukkasanat, eli sisällöllisesti vailla itsenäistä merkitystä olevat yleiset side- ja täytesanat.

Suomi on voimakkaasti taipuva kieli, jossa sama sana voi saada sen roolista lauseessa riippuen kymmeniä, teoriassa jopa tuhansia (Suominen 2010) erilaisia muotoja: esimerkiksi sana ”hallitus” esiintyy eduskunnan keskustelupöytäkirjoissa muodoissa ”hallituksen”, ”hallitusta”, ”hallitukseen”, ”hallituksesta” jne. Keskustelujen datajournalistisen analysoinnin kannalta olisi perusteltua, että saman sanan eri taivutusmuotojen esiintymät laskettaisiin yhteen sen sijaan, että jokainen taivutusmuoto tulkittaisiin omaksi sanakseen ja niiden esiintymät laskettaisiin erikseen. Tämä edellyttää tekstin normalisointia (ks. alaluku 3.2) ja suomenkielisen tekstin normalisointimenetelmäksi soveltuu parhaiten perusmuotoistaminen eli lemmaus.

Koska aineistossa on sekä suomen- että ruotsinkielisiä puheenvuoroja, ennen perusmuotoistamista on jokaisesta puheenvuorosta selvitettävä, kumpaa kieltä puhuja on käyttänyt. (Muunkielisiä – esimerkiksi saamen- tai romaninkielisiä – puheenvuoroja en aineistosta ole löytänyt, mutta projektin myöhemmässä vaiheessa olen huomannut, että varsinkin uudemmissa pöytäkirjoissa on paljon kaksikielisiä puheenvuoroja, joissa kansanedustaja lukee etukäteen valmistellun puheenvuoron ensin ruotsiksi ja sitten suomeksi. Vanhempien pöytäkirjojen perusteella tehty oletus siitä, että kukin puheenvuoro

[3] CSV on taulukkomuotoisen tiedon tallentamiseen tarkoitettu yksinkertainen tiedostomuoto, jossa kentät erotetaan toisistaan pilkuilla ja rivit rivivaihdoilla. TSV on sen variantti, jossa erottimena käytetään pilkkujen sijasta sarkainmerkkejä (Koponen 2021). Molempia muotoja hyödynnetään laajasti avoimen lähdekoodin ohjelmistoissa ja datajournalismissa.

[4] Kansanedustaja Kaino Haapasen (mies) ja Kaino Oksanen (nainen) sukupuoli määriteltiin käsin, sillä Kaino-etunimi ei kerro henkilön sukupuolta.

olisi yksikielinen ei siis pidä paikkaansa, mikä on ongelmallista, sillä aineisto on kokonaisuudessaan käsitelty olettaen, että puheenvuorot ovat aina yksikielisiä.)

Päädyin toteuttamaan kielen tunnistuksen laskemalla paljonko puheenvuorossa esiintyy suomen- ja ruotsinkielisiä hukkasanoja ja päättelemään, että teksti on suomenkielinen jos suomenkielisiä sanoja on enemmän tai päinvastoin. Hukkasanojen tunnistamisessa on hyödynnetty Stopwords ISO -pakettiin (Stopwords ISO 2016) kuuluvia hukkasanalistoja, joita on täydennetty noin parilla sadalla omalla lisäyksellä. Lisätyt hukkasanat ovat pääosin tekstintunnistuksen virheellisesti tunnistamia muotoja (*niimt*, *mähända*) tai vanhahtavia ilmaisuja (*elikkä*, *öfver*) joilla on sama merkitys kuin jollain hukkasanalistalla valmiiksi olevalla sanalla, mutta joukossa on myös muutamia eduskunnan keskustelutyyliin liittyviä erikoisuuksia, kuten ”vasemmalta” ja ”oikealta”, jotka esiintyvät lähes aina vain välihuutojen yhteydessä.

Kun puheenvuorot on jaoteltu suomen- ja ruotsinkielisiin, suomenkieliset puheenvuorot perusmuotoistetaan morfologisen jäsentimen avulla. (Edellisessä alaluvussa kuvailtujen ruotsinkielisen tekstin tekstintunnistusongelmien vuoksi päädyin lopulta olemaan tekemättä lainkaan lemmausta ruotsinkielisille puheenvuoroille.) Tarkoitukseen on käytettävissä ainakin neljä avoimen lähdekoodin ohjelmistoa (Hämäläinen & Alnajjar 2021; John Snow Labs 2022):

1. Helsingin yliopiston yleisen kielitieteen laitoksella kehitetty **Omorfi**, jonka ”paranneltu painos” on aiemmin samalla laitoksella tutkijoina toimineiden Miikka Silfverbergin ja Teemu Ruokolaisen kehittämä **FinnPos**
2. Turun yliopiston TurkuNLP-tutkimusryhmän kehittämä **Turku-neural-parser-pipeline (TNPP)** ja sen vastaava paranneltu versio **Finnish-dep-parser**
3. Uudempi, 2019 julkaistu Helsingin yliopiston tutkijoiden Mika Hämäläisen ja Khalid Alnajjarin kehittämä **UralicNLP**
4. Yhdysvaltalaisen John Snow Labs -yhtiön kehittämä monikielinen avoimen lähdekoodin **Spark NLP** -jäsennin tukee myös suomenkielisen tekstin jäsentämistä ja lemmatisointia

Avoimen lähdekoodin jäsentimien lisäksi suomen kielen perusmuotoistamiseen on tarjolla myös joitakin kaupallisia ohjelmia, esim. Lingsoft-yhtiön kehittämä FINTWOL-jäsennin (Koskenniemi 2022).

Pirinen (2019) on vertaillut Omorfin ja TNPP:n suorituskykyä perusmuotoistamisessa. Omorfi osasi perusmuotoistaa koeaineiston sanoista oikein vain n. 83 %, kun TNPP ylsi lähes 96 % tarkkuuteen. Tämä puoltaisi TNPP:n tai Finnish-dep-parserin valitsemista perusmuotoistajaksi, mutta en valitettavasti ollut tietoinen Pirisen tutkimuksesta (enkä UralicNLP- ja Spark NLP-jäsentimien olemassaolosta) siinä vaiheessa kun toteutin kehitystyön produktiivista osuutta ja olen päättynyt käyttämään perusmuotoistajana hänen

vertailussaan huonommin menestyneeseen Omorfiin perustuvaa FinnPos-jäsennintä. FinnPos mahdollistaa jäsentämiseen käytetyn neuroverkon kouluttamisen myös itse, mutta olen käyttänyt asennuspaketin mukana tulevaa Omorfin valmiiksi koulutettua mallia.

Yhtenä keskeisenä syynä FinnPosin valintaan oli se, että olin käyttänyt sitä aiemminkin ja saanut sen onnistuneesti asennettua käyttämääni macOS-ympäristöön. Usein Linux-ympäristöön suunniteltujen avoimen lähdekoodin ohjelmien käyttöönotto ei valitettavasti ole aina aivan mutkatonta Mac-koneilla. (Tämän kehittämistyön liitteessä 2 on laatimani ohjeet FinnPos-jäsentimen asentamisesta macOS-käyttöjärjestelmään, mikäli niistä on joillekin lukijoille hyötyä.)

Prosessi alkaa saneistamalla TSV-tiedostoihin tallennetut puheenvuorot yksinkertaisella algoritmilla, joka lähinnä vain pilkkoo tekstin sanoiksi välilyöntien kohdalta ja tekee muutamia pieniä korjauksia kuten poistaa väli- ja erikoismerkit. Saneet pilkotaan kukin omalle rivilleen ja jos puheenvuoro on suomenkielinen, ohjataan FinnPosin käsiteltäväksi.

FinnPosiin ei ole tehty Python-rajapintaa, vaan sitä käytetään tavallisesti *komentoriviltä* eli ns. komentotulkin (esim. macOS:n Pääte tai Windowsin PowerShell) avulla. Olen ratkaissut asian omalta osaltani siten, että koodini käyttää Pythonin subprocess-moduulia, joka mahdollistaa rivikomentojen antamisen Pythonista. Perusmuotoistettava teksti tallennetaan `temp.txt`-nimiseen tilapäistiedostoon, jonka jälkeen subprocess kutsuu FinnPosiin kuuluvaa ftb-label-moduulia Unix-komennolla `cat temp.txt | ftb-label` eli listaa tiedoston sisällön cat-komennolla ja ohjaa komennon palauttaman merkkijonon jäsentimen käsiteltäväksi. Moduulin palauttama tulos otetaan talteen käsittelyä varten.

```
valtiopäiväjärjestyksen _    valtiopäiväjärjestys    [POS=NOUN]|[NUM=SG]|[CASE=GEN]   _
sanotaan    _   sanoa   [POS=VERB]|[VOICE=PSS]|[MOOD=INDV]|[TENSE=PRESENT]|[PERS=PE4]   _
lippuäänestyksestä  _   lippuäänestys   [POS=NOUN]|[NUM=SG]|[CASE=ELA]  _
näin    _   näin    [POS=ADVERB]|[SUBCAT=DEMONSTRATIVE] _
lippuäänestyksessä  _   lippuäänestys   [POS=NOUN]|[NUM=SG]|[CASE=INE]  _
käytettäköön    _   käyttää [POS=VERB]|[VOICE=PSS]|[MOOD=IMPV]|[PERS=PE4]   _
lippuja _   lippu   [POS=NOUN]|[NUM=PL]|[CASE=PAR]  _
joihin  _   joka    [POS=PRONOUN]|[SUBCAT=RELATIVE]|[NUM=PL]|[CASE=ILL] _
on  _   olla    [POS=VERB]|[VOICE=ACT]|[MOOD=INDV]|[TENSE=PRESENT]|[PERS=SG3]   _
painettuna  _   painaa  [POS=VERB]|[VOICE=PSS]|[PCP=NUT]|[NUM=SG]|[CASE=ESS]    _
jaa _   jaa [POS=PARTICLE]|[SUBCAT=INTERJECTION]    _
taikka  _   taikka  [POS=PARTICLE]|[SUBCAT=CONJUNCTION]|[CONJ=COORD]    _
ei  _   ei  [POS=VERB]|[SUBCAT=NEG]|[VOICE=ACT]|[PERS=SG3]  _
ja  _   ja  [POS=PARTICLE]  _
```

Kuva 7. Esimerkki FinnPosin ftp-label-moduulin palauttamasta tuloksesta. Alkuperäisen sanan lisäksi näytetään sanan perusmuoto, sanaluokka (POS, *part of speech*) sekä taivutusmuodot (esim. kieliopillinen luku NUM ja sijamuoto CASE).

Aluksi yritin tehdä perusmuotoistamisen yksi puheenvuoro kerrallaan, mutta tämä osoittautui erittäin hitaaksi, sillä FinnPosin käynnistäminen joka rivikomennon yhteydessä uudestaan vie melko paljon aikaa – koko aineiston prosessointi olisi vienyt jopa kuukausia. Päädyin liittämään tilapäistiedostoon useita puheenvuoroja peräkkäin ja ohjaamaan ne jäsentimen käsiteltäväksi vasta kun tietty määrä puheenvuoroja tuli täyteen. Kokeilemalla hyväksi määräksi osoittautui sata puheenvuoroa kerralla, jolloin koko aineiston prosessointiaika lyheni n. 12 tuntiin late 2018 MacBook Pron yhdellä prosessoriytimellä. Puheenvuorojen välissä käytetään tunnuksena merkkijonoa ” ——” jota seuraa juokseva numero välillä 0–99. Näitä merkkijonoja käytettään erottamaan eri puheenvuoroihin liittyvät sanat toisistaan jäsennetyn datan käsittelyvaiheessa. FinnPos palauttaa sanan perusmuodon lisäksi myös sanaluokan ja taivutusmuodot, jotka myös otetaan talteen myöhempää käyttöä varten.

FinnPos tekee perusmuotoistuksessa tiettyjä systemaattisia virheitä. Se esimerkiksi tulkitsee virheellisesti sanan ”paras” olevan sanan ”parka” taivutusmuoto. Varsinkin erisnimet ovat FinnPosille haastavia – jostain syystä jäsennin tulkitsee monet sukunimet verbeiksi, esim. ”andersson” → ”anderssondä”. Olen koonnut perusmuotoistamisessa ja tekstintunnistuksessa tai niiden yhdistelmänä syntyneistä yleisimmistä virheellisistä muodoista lähes 3 000 suomenkielisen sanan luettelon, jota vastaan FinnPosin palauttamia tuloksia verrataan. Jos tuloksissa on luettelosta löytyvä virheellinen muoto, se korvataan samasta luettelosta löytyvällä oikealla muodolla ja tiedolla sanan oikeasta sanaluokasta. Vastaavassa ruotsinkielisessä luettelossa on vain n. 60 sanaa, koska edellä kuvattujen ruotsinkielisen tekstin tekstintunnistuksen ongelmien vuoksi olen päättänyt keskittyä kehittämistyössäni pääosin suomenkielisten puheenvuorojen käsittelyyn. Tutkin alustavasti myös mahdollisuutta lemmata ruotsinkieliset puheenvuorot Lemmy-perusmuotoistajalla (Lind 2019), mutta päädyin lopulta jättämään myös tämän vaiheen pois.

Jäsentämisen ja virheiden korjaamisen jälkeen puheenvuorot tallennetaan uuteen TSV-tiedostoon, johon otetaan alkuperäisen puheenvuorotekstin lisäksi luettelo jäsennettyjen sanojen alkuperäisistä muodoista luettelona, niiden perusmuodot, sanaluokat ja taivutusmuodot.

4.3.4 Ryvästäminen Word2Vec-algoritmilla

Viimeisenä käsittelyvaiheena kokeilin vielä muodostaa aineistosta sanaryppäitä eli -klustereita Word2Vec-algoritmin avulla. Word2Vec on Googlen kehittämä ja luonnollisen kielen analyysissä yleisesti käytetty työkalu, joka analysoi tietyssä korpuksessa esiintyvien

sanojen keskinäistä samankaltaisuutta muuttamalla ne vektoreiksi[5] (Mumbi 2021). Algoritmi perustuu ohjaamattomaan oppimiseen (ks. alaluku 3.1), eli se ei kerro ”oikeita” vastauksia samaan tapaan kuin vaikkapa edellä käsitelty FinnPos, vaan ainoastaan paljastaa aineistossa esiintyviä piirteitä, esimerkiksi mitkä sanat esiintyvät usein yhdessä. Vektoreilla voidaan tehdä esimerkiksi erilaisia laskutoimituksia, ja jos mallin kouluttamiseen käytetty aineisto on riittävän suuri, niiden tulokset voivat paljastaa jotain mielenkiintoista aineistossa esiintyvien sanojen suhteesta. (Google 2013) Esimerkiksi eduskunnan keskustelupöytäkirjoista koostamallani aineistolla koulutetulla malleilla voidaan suorittaa seuraavanlaisia laskutoimituksia ja saada lähimmäksi tulokseksi tässä lihavoidulla tekstillä kirjoitettu sanavektori:

kenraali – puolustusvoimat + tasavalta = **presidentti**
eduskunta – puhemies + yleisradio = **hallintoneuvosto**
vennamo – smp + kokoomus = **ilaskivi**

Kokonaisuudessaan eduskunnan keskustelujen muodostama korpus on kuitenkin liian pieni, jotta useimmat tällaiset laskutoimitukset antaisivat kiinnostavia tuloksia.

Käytin Word2Veciä avoimen lähdekoodin Gensim-ohjelmistopaketin (Řehůřek 2014) avulla. Sen avulla mallin kouluttaminen on äärimmäisen helppoa. Muodostin korpuksen keräämällä edellisessä vaiheessa perusmuotoistetut puheenvuorot peräkkäin yhteen ilmitekstitiedostoon. Korpuksesta muodostetaan yksinkertainen Python-iteraattori ja annetaan se parametriksi Gensimin Word2Vec-komennolle, joka kouluttaa mallin aineiston perusteella alle 10 minuutissa. Valmis malli voidaan tallentaa tiedostoksi ja hyödyntää sitä myöhemmin uudestaan ilman, että sitä tarvitsisi kouluttaa joka kerta uudestaan.

Hyödynsin Word2Vecin st_similar-toimintoa merkitykseltään samankaltaisten sanojen ryppäiden eli klustereiden selvittämiseen. Muodostin ryppäät aineiston 25 000 yleisimmälle perusmuotoistetulle sanalle, mikä paljasti suuren joukon tekstintunnistuksessa tai lemmauksessa syntyneitä virheellisiä muotoja, sillä sekä oikein että virheellisesti tunnistetut sanat esiintyvät tekstissä usein samankaltaisissa yhteyksissä, joten algoritmi yhdistää ne samaan klusteriin. Keräsin näitä virheellisiä muotoja

[5] Vektori on matemaattinen suure, jolla on suunta ja pituus. Fysiikassa esimerkiksi voimia kuvataan vektoreina. Vektoreilla voidaan suorittaa monia laskutoimituksia kuten yhteen-, vähennys- ja kertolaskuja. (Tran 2021)

edellisessä alaluvussa kuvailemaani virheellisten muotojen listaan ja suoritin perusmuotoistamisen uudestaan niin, että virheet korjattiin. Tämän jälkeen koulutin mallin ja tein ryvästämisen uudestaan. Keräsin uusia paljastuneita virheellisiä muotoja listaan ja toistin prosessin muutamia kertoja kunnes listan alkupäässä (kaikkein yleisimpien sanojen kanssa samoisa ryppäissä) ei enää virheellisiä muotoja näkynyt.

Tämän jälkeen kävin ryppäät läpi käsin yleisyysjärjestyksessä etsien sellaisia klustereita, joissa algoritmi oli koonnut yhteen merkitykseltään samankaltaisia sanoja, jotka saattoivat olla synonyymejä (esim. ”aktiivisuus” ja ”aloitteellisuus”) tai muulla tavoin merkitykseltään lähellä toisiaan (esim. ”afganistan” ja ”afganistanilainen”). Koostin vajaan tuhannen ryppään luettelon ja lisäsin joihinkin niistä vielä käsin joitain yleisiä synonyymejä. Ajatuksena on, että näin muodostettuja sanaryppäitä voidaan käyttää keskustelun teemojen tunnistamiseen paremmin kuin yksittäisiä sanoja. Samaan tarkoitukseen voisi käyttää myös esimerkiksi aihemalleja (ks. alaluku 3.2), mutta rajasin aihemallien käytön tietoisesti pois jo kehittämistyön alkuvaiheessa, jotta projektissa käytettävien tekniikoiden määrä pysyisi hallittavana.

4.4 Aineiston tallettaminen tietokantaan

Edeltävissä aineiston keräämisen ja analysoinnin vaiheissa tietoja käsiteltiin irrallisina TSV- tai ilmitekstitiedostoina. Seuraava työvaihe kehittämistyössäni oli siirtää aineisto tietokantaan.

En ollut aikaisemmin itse pystyttänyt tietokantaa – tai edes kovin paljoa käyttänyt jonkun muun pystyttämää tietokantaa muutoin kuin graafisen käyttöliittymän avulla –, joten tämä oli minulle aivan uusi aihepiiri ja vaati asioiden opettelua alkeista saakka. Ensimmäinen selvitettävä asia oli, mikä tietokantaohjelmisto sopisi parhaiten tarkoitukseen. Koska halusin oppia käyttämään myös relaatiotietokantojen käsittelyssä yleistä SQL-ohjelmointikieltä, valinta tapahtui avoimen lähdekoodin SQL-pohjaisten ohjelmistojen välillä, joista yleisimmin käytetyt ovat MySQL, PostgreSQL ja SQLite.

Päädyin valitsemaan PostgreSQL-tietokannan ennen kaikkea sen hyvän Python-tuen vuoksi. PostgreSQL:ää pidetään myös hyvin skaalautuvana – ts. se soveltuu myös vaativampiin, suuria tieto- ja/tai käyttäjämääriä palveleviin ratkaisuihin (esim. Hristozov 19.7.2019). Vaikka skaalautuvuudella ei kehittämistyöni kannalta ole merkitystä, pidän kuitenkin perusteltuna opetella käyttämään ohjelmistoa, joka on käyttökelpoinen myös sellaisissa projekteissa, joissa sillä on. (Eri SQL-ohjelmistot eroavat hieman toisistaan siinä, miten SQL-kielen komentoja niissä tulkitaan, joten esimerkiksi MySQL-ympäristöön

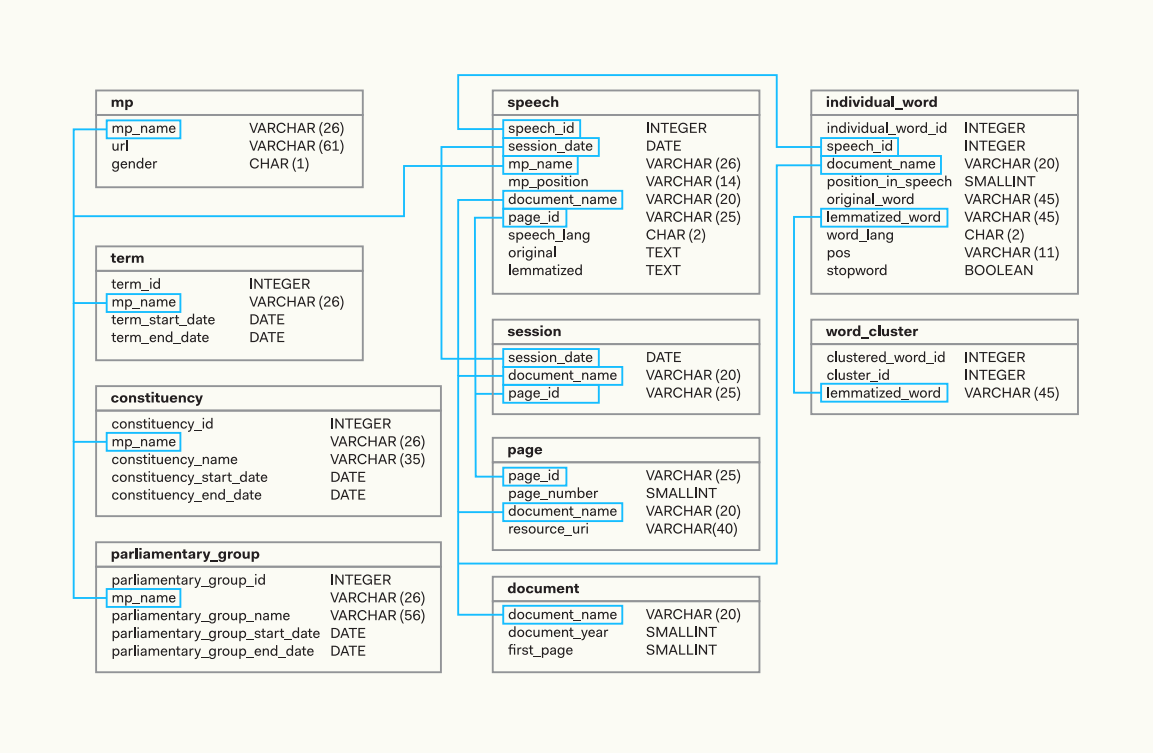
kirjoitettu koodi ei välttämättä aivan sellaisenaan toimi PostgreSQL-ympäristössä ja päinvastoin.)

4.4.1 Tietokannan pystytys ja aineiston siirto kantaan

Alkuvaiheessa asensin PostgreSQL:n omalle koneelleni ja käsittelin sitä paikallisesti. Myöhemmin, kun olin saanut tietokannan toimimaan omalla koneellani halutulla tavalla, tein paikallisesta tietokannastani kopion pilveen selainpohjaisen käyttöliittymän pystyttämistä varten (ks. alaluku 4.5.1). Käyttämäni ohjelmistoversio on macOS:lle helposti asennettava Postgres.app (Postgres.app s.a.), jonka valintaa puolsi myös, että se sisältää paikkatiedon käsittelyyn PostgreSQL:llä tarvittavan PostGIS-laajennukseen. (Kehittämistyöhöni ei sisälly paikkatiedon käsittelyä, mutta päivätyöhöni sisältyy.)

Puheenvuorojen ja kansanedustajien biografiatietojen siirto kantaan tapahtui Pythonin psycopg2-moduulilla. Hyödynsin tässä moduulin kehittäjien valmista esimerkkikoodia, jota muokkasin omiin tarkoituksiini paremmin soveltuvaksi. Siirron yhteydessä tein käsin joitakin korjauksia täysistuntojen painovirheen tai tekstintunnistuksen virheen vuoksi virheellisiin päivämäärätietoihin. Kansanedustajan tietoihin lisättiin sukupuoli, joka pääteltiin etunimen perusteella (vrt. alaluku 4.3.2).

4.4.2 Tietokantarakenne



Kuva 8. Tietokannan rakenne, käytetyt tietotyypit ja yhteydet tietokantataulujen välillä.

Useiden iteraatioiden jälkeen päädyin yllä olevan kaavion kuvaamaan tietokantarakenteeseen. Kuvailen seuraavassa lyhyesti taulujen sisällön:

mp sisältää kansanedustajan nimen siinä muodossa kuin se on eduskunnan verkkosivuilla, linkin kyseiselle sivulle ja kansanedustajan sukupuolen.

term sisältää kunkin kansanedustajan yhtäjaksoisen toimikauden tai -kaudet, joita voi olla useampi kuin yksi, jos kansanedustaja on pudonnut vaaleissa tai lähtenyt muusta syystä pois eduskunnasta ja palannut sinne myöhemmin. Jokaisen kansanedustajan kukin toimikausi on oma rivinsä, jolla on oma id-numeronsa ja joka linkitetään kansanedustajaan käyttäen avaimena tämän nimeä ja josta on tallennettu alku- ja loppuajankohta.

constituency ja **parliamentary_group** -taulut kertovat kansanedustajan edustaman vaalipiirin tai -piirit ja eduskuntaryhmän tai -ryhmät, joihin hän on kuulunut. Ne muistuttavat rakenteeltaan läheisesti **term**-taulua, mutta niissä on myös kenttä vaalipiirin tai eduskuntaryhmän nimelle.

speech sisältää kaikki aineistossa olevat puheenvuorot. ID-numeron lisäksi taulun sarakkeina ovat avaimina muihin tauluihin käytettävät täysistunnon päivämäärä ja kansanedustajan nimi, sekä puheenvuoron sisältävän pöytäkirjan tiedostonimi ja sen yksittäisen sivun, jolta puheenvuoro alkaa id-numero. Muita kenttiä ovat puhujan rooli (rivikansanedustaja tai esim. puhemies tai ministeri), puheen kieli (suomi tai ruotsi) sekä alkuperäinen ja perusmuotoistettu puheen teksti.

session sisältää täysistuntojen päivämäärät ja tiedon siitä, minkä nimiseen tiedostoon tallennettuun pöytäkirjaan se kuuluu ja miltä sivulta alkaa.

page sisältää tiedot kustakin yksittäisestä pöytäkirjan sivusta: juokseva sivunumero pöytäkirjan alusta lukien (ei välttämättä sama kuin sivun kulmaan merkitty sivunumero, ks. seuraavan taulun kuvaus), pöytäkirjatiedoston nimi ja linkki skannattuun kuvaan sivusta. (Skannatut kuvat on tallennettu Amazonin S3-palveluun, ks. alaluku 4.5.2.)

document sisältää perustiedot pöytäkirjasta: tiedostonnimen, vuosiluvun ja ensimmäisen sivun sivunumeron. (Saman vuoden pöytäkirjoissa on käytössä juokseva numerointi, niin että vuoden toisen pöytäkirjan numerointi alkaa siitä, mihin ensimmäisen päättyy jne.)

individual_word sisältää jokaisen aineistosta löytyvän yksittäisen sanan, tiedon mihin puheenvuoroon ja mihin pöytäkirjaan se sisältyy, monesko puheenvuoron sana se on, sanan alkuperäisen ja perusmuotoistetun muodon, sanan kielen, sanaluokan sekä tiedon, onko sana hukkasana vai ei.

word_cluster sisältää tiedot merkitykseltään samankaltaisten sanojen ryppäistä (ks. alaluku 4.3.4). Perusmuotoistetut sanat yhdistetään toisiinsa cluster_id-avaimen avulla.

Tietotyyppien kuvaukset:

CHAR(n) on merkkijono (tekstinpätkä, jonka pituus on **n** merkkiä
VARCHAR(n) on merkkijono, jonka pituus on *korkeintaan* **n** merkkiä
TEXT on merkkijono, jonka pituudelle ei ole asetettu ylärajaa
INTEGER on kokonaisluku
SMALLINT on pieni kokonaisluku (itseisarvo pienempi kuin 2^{15} eli korkeintaan 32 767)
DATE on päivämäärä
BOOLEAN on totuusarvo (ns. Boolen arvo), eli joko tosi tai epätosi

CHAR-tietotyypin käyttö VARCHARin sijaan nopeuttaa tietokannan toimintaa, mutta sama teksti vie CHAR-muodossa kantaan tallennettuna yleensä enemmän tilaa kuin VARCHARina, joten sitä kannattaa käyttää vain hyvin lyhyisiin, vakiomittaisiin teksteihin.

```
eduskunta=# \dt+
                            List of relations
 Schema |        Name          | Type  |  Owner   |  Size   | Description
--------+----------------------+-------+----------+---------+-------------
 public | constituency         | table | postgres | 368 kB  |
 public | document             | table | postgres | 56 kB   |
 public | individual_word      | table | postgres | 3525 MB |
 public | mp                   | table | postgres | 408 kB  |
 public | page                 | table | postgres | 26 MB   |
 public | parliamentary_group  | table | postgres | 360 kB  |
 public | session              | table | postgres | 512 kB  |
 public | speech               | table | postgres | 622 MB  |
 public | term                 | table | postgres | 232 kB  |
 public | word_cluster         | table | postgres | 184 kB  |
(10 rows)
```

Kuva 9. Tietokannan taulut asiakasohjelmassa (*client*) lueteltuna.

Kuten yllä olevasta taulujen luettelosta ilmenee, ylivoimaisesti suurin taulu on yksittäiset sanat sisältävä individual_word, jonka koko on n. 3,5 gigatavua. Myös speech-taulu on melko suuri, yli 600 megatavua. Muut taulut ovat verrattain pieniä.

4.5 Verkkopohjaisen käyttöliittymän rakentaminen

Käyttöliittymä on se rajapinta, jonka kautta ihminen pystyy omaksumaan ja käsittelemään tietojärjestelmän kuten tietokannan sisältämää tietoa (Koponen 14.8.2018). Käyttöliittymän suunnittelu eroaa esimerkiksi taideteollisesta muotoilusta siinä, että käyttöliittymän ei ole tarkoitus vetää huomiota itseensä sen enempää hyvässä kuin pahassakaan, vaan ainoastaan mahdollistaa vuorovaikutus ihmisen ja tietojärjestelmän

välillä. Hyvin suunniteltu käyttöliittymä ikään kuin katoaa taustalle ja antaa käyttäjän keskittyä työn alla olevaan tehtävään (Shneirderman & Plaisant 2005, s. 12). Pidän hyvänä analogiana tälle ajatukselle hyvän käyttöliittymän näkymättömyydestä Beatrice Warden (2005) esittämää ajatusta typografiasta eli tekstin ulkoasun muotoilusta ”viinilasina”. Erilaisilla ja eri tyylisillä laseilla voi olla erilaisia ominaisuuksia, mutta kaikille niille yhteistä on sama käyttötarkoitus: viinin kuljettaminen. Viini on aina tärkeämpi kuin lasi, johon se on kaadettu – samoin teksti on tärkeämpi kuin kirjaimet, joilla se on ladottu. Erilaiset käyttöliittymät voivat palvella eri tarkoituksia tai toimia välineenä saman tehtävän suorittamiseen, mutta käyttöliittymä itsessään on toissijainen – sen tarkoitus on vain mahdollistaa yhteistyö ihmisen ja koneen välillä.

Shneirderman & Plaisant (2005, s. 16) luettelevat viisi keskeistä käyttöliittymän toimivuuden mittaria:

1. Käytön oppimisen vaatima aika
2. Käytön nopeus
3. Käyttäjien tekemien virheiden määrä
4. Muistissa pysyminen, eli miten helppo tauon jälkeen palaavan käyttäjän on muistaa, miten käyttöliittymä toimii
5. Subjektiivinen miellyttävyys

He toteavat, että täydellinen onnistuminen jokaisessa kategoriassa samanaikaisesti ei useinkaan ole mahdollista, vaan eri toimivuuden indikaattoreiden välillä joudutaan yleensä tekemään kompromisseja. Omassa kehittämistyössäni tämä Shneidermanin ja Plaisantin esittämä järjestys on muovautunut myös prioriteettijärjestykseksi: olen keskittynyt käyttöliittymän suunnittelussa kaikkein eniten siihen, että käyttöliittymä olisi nopea oppia ja muihin mittareihin laskevassa prioriteettijärjestyksessä, unohtamatta kuitenkaan listan viimeistäkään kohtaa. Jää käyttäjän arvioitavaksi, kuinka hyvin olen tässä tehtävässä onnistunut.

Berezhnoi (2019) pitää käyttöliittymän suunnittelun kolmena keskeisenä tehtävänä visuaalista suunnittelua, vuorovaikutussuunnittelua ja informaatioarkkitehtuurin suunnittelua. Visuaalisen suunnittelun alaan hän lukee mm. kirjaintyyppien, värien ja kuvien käytön, vuorovaikutussuunnitteluun erilaiset vuorovaikutteisten elementtien toiminnan suunnittelun ja informaatioarkkitehtuuriin sivun sisältämien tietoaineistojen nimiönnin, rakenteen suunnittelun ja järjestämisen, mutta myös hakukoneoptimoinnin. Vaikka vuorovaikutteisen käyttöliittymän suunnitteluun väistämättä kuuluvat todellakin kaksi ensimmäistäkin osa-aluetta, jouduin kehittämistyössäni keskittymään olosuhteiden pakosta ylivoimaisesti eniten kolmanteen osa-alueeseen, sillä käyttöliittymän tekninen

toteutus osoittautui paljon etukäteen arvioitua työläämmäksi, kuten seuraavissa alaluvuissa kuvailen.

4.5.1 Datan siirtäminen Amazonin pilveen

Kun olin saanut tietokannan toimimaan halutulla tavalla omalla koneellani, seuraava vaihe oli sen siirtäminen pilveen, jotta se olisi saavutettavissa selainpohjaisen käyttöliittymän kautta. Pilvipalveluksi valikoitui Amazonin RDS (db.t3.micro-palvelutaso) lähinnä kustannussyistä. Tietokannan ja taustaohjelmiston (ks. seuraava alaluku) yhteenlasketut käyttökustannukset ovat olleet n. 25 €/kk (alv 0), kun kilpailevilla palveluilla kustannus näin suuren PostgreSQL-kannan isännöinnille olisi ollut moninkertainen. Minulla oli entuudestaan hyviä kokemuksia Amazonin AWS-brändin alla toimiviin pilvipalveluihin lukeutuvan S3-säilytyspalvelun käytöstä verkkosivujen isännöintiin, mikä myös osaltaan puolsi Amazonin valintaa palveluntarjoajaksi.

Tietojen kopioiminen RDS-isännöityyn kantaan oli helppoa: se onnistui edellistä työvaihetta varten kirjoittamallani psycopg2-moduulia hyödyntävällä ohjelmalla, kun vain tietokannan asetukset sisältävään db.ini-tiedostoon vaihdettiin paikallisosoitteen (*localhost*) tilalle RDS-tietokannan http-osoite ja kirjautumistiedot (käyttäjätunnus ja salasana) päivitettiin.

4.5.2 Taustaohjelmiston ja rajapintojen rakentaminen

Kehittämistyöni teknisesti haastavimmaksi osuudeksi osoittautui tietokannan ja käyttäjän selaimen välissä toimivan taustaohjelmiston (*backend*) ja sen tarvitseman rajapinnan rakentaminen. Amazonin valitseminen palveluntarjoajaksi alkoi hetkittäin tuntua väärältä päätökseltä, sillä vaikka palvelut ovat edullisia ja monipuolisia, niiden käyttö on teknisesti monimutkaista. Isoksi ongelmaksi muodostui, että AWS-palveluiden tekninen dokumentointi on puutteellista ja enimmäkseen suunnattu ammattimaisille ohjelmistokehittäjille itseni kaltaisten aloittelijoiden sijaan. Kuvaan seuraavassa yksinkertaistetusti taustaohjelmiston teknisen ratkaisun ja muutamia kaikkein suurimpia haasteita sen kehittämisessä, mutta kuvaus on huomattavan paljon virtaviivaistettu versio todellisuudessa tahmeasti edenneestä, työläästä ja hetkittäin jopa kaoottisesta prosessista.

Suunnitelmani oli pystyttää RESTful-rajapinta (Gupta 2021), joka mahdollistaisi tietokantahakujen tekemisen selainpohjaisen käyttöliittymän kautta RDS-palvelussa isännöidystä kannasta. Ensimmäinen tehtävä oli luoda tämä rajapinta sekä sen ja tietokannan välissä taustaohjelmistona toimiva ns. Lambda-funktio.

Lambda-funktio on verkossa pyörivä ”piensovellus”: yhden tai muutaman yksittäisen asian suorittava koodinpätkä. Palvelu mahdollistaa ohjelmoinnin useilla eri ohjelmointikielillä. (Tero 2016.) Omassa käyttötapauksessani sen tehtävänä on käsitellä RESTful-rajapinnasta tulevia tietokantakyselyitä, ohjata ne oikeaan paikkaan ja palauttaa kyselyn tulokset rajapintaan. Halusin toteuttaa funktion Python-ohjelmointikielellä, koska se oli minulle tuttu tapa työskennellä. Koska minulla ei ollut aiempaa kokemusta Lambda-funktioista tai rajapinnan pystyttämisestä, seurasin toteutuksessa AWS Database Blog -blogista löytämääni ohjetta (Holmer 19.3.2018), jota sitten sovelsin omiin tarpeisiini. Esivalmistelujen ja edellisessä alaluvussa kuvatun RDS-tietokannan pystyttämisen jälkeen työn vaiheet olivat seuraavat:

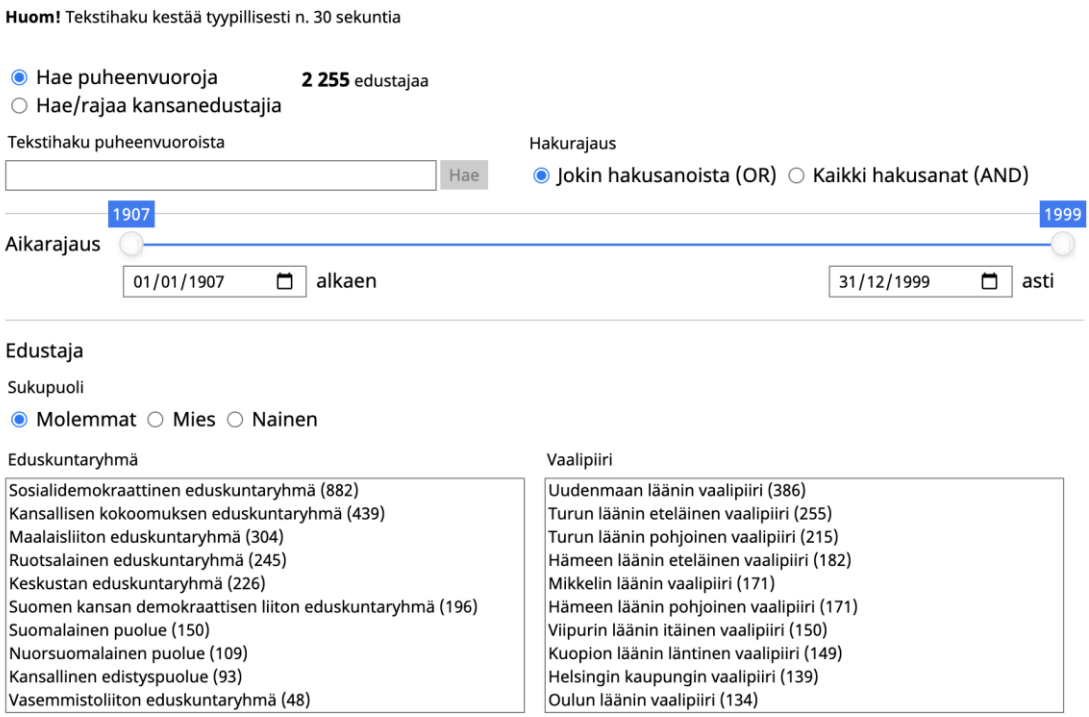
1. Luodaan EC2-virtuaalipalvelin.
2. Lisätään virtuaalipalvelin tietokannan turvallisuusryhmän sallittujen sisäänpäin suunnatun liikenteen (*inbound rules*) lähteiden joukkoon.
3. Kopioidaan Amazonin GitHub-sivulta valmis serverless-query.py-ohjelmakoodi, siirretään se EC2-palvelimelle ja muokataan palvelimen ja tiedoston asetuksia niin, että koodi voidaan suorittaa.
4. Luodaan yhteys RDS-tietokantaan virtuaalipalvelimen käyttöympäristön asetukset oikein asettamalla.
5. Muokataan koodia halutulla tavalla.
6. Luodaan käyttöönottopaketti (*deployment package*).
7. Otetaan käyttöön sopiva S3-säilytyspalvelun virtualisointiympäristö (*instance*) ja kopioidaan käyttöönottopaketti sinne.
8. Luodaan CloudFormation-hallintapalvelun pino (*stack*), joka yhdistää käyttöönottopaketin ja tietokantayhteydet. Tämän toimenpiteen seurauksena AWS tekee käyttöönottopaketin koodista Lambda-funktion ja luo sille tarvittavan RESTful-rajapinnan Amazonin API Gateway -palveluun.

Lähteenä käyttämäni kirjoitus (Holmer 19.3.2018) sisältää myös ohjeet koodin tuottamien virheilmoitusten tarkistamiseen ongelmatilanteissa CloudWatch-lokitiedoista, mikä muodostuikin erittäin tarpeelliseksi, sillä ongelmia oli paljon. Yhdeksi suurimmista osoittautui, että vuonna 2018 kirjoitettu ohje olettaa, että uusi, ”pakasta vedetty” EC2-palvelin olisi varustettu Python-ohjelmointikielen versiolla 3.6, kun tosiasiassa uusissa EC2-virtualisointiympäristöissä käytettiin työn toteuttamisaikaan jo Pythonin versiota 3.7. Vaikka versionumeron ero voi vaikuttaa pieneltä, osa työssä tarvittavista Python-moduuleista ei kerta kaikkiaan toimi uudemmalla versiolla. Eri ratkaisuvaihtoehtoja kokeiltuani onnistuin lopulta suurella vaivalla asentamaan Pythonin vanhemman 3.6-version EC2-palvelimelle, jonka jälkeen tarvittavat moduulit saatiin vihdoin toimimaan.

Kehittämistyön loppuvaiheessa ilmeni, että Amazon on luopumassa Python 3.6-version teknisestä tuesta vaiheittain 18.7.2022 alkaen (Amazon Web Services, Inc. 15.4.2022), jolloin myös kehittämäni käyttöliittymä lakkaa toimimasta.

Yksi työtä keskeisesti nopeuttanut oivallus oli, että Lambda-funktion koodin päivittäminen ei vaadi yllä kuvatun prosessin suorittamista vaiheesta 5 alkaen uudestaan, kuten alkuun luulin, vaan Amazon tarjoaa Lambdalle oman selainpohjaisen käyttöliittymän, jossa funktion koodia voi muokata suoraan ja uuden version käyttöönotto tapahtuu yhdellä napinpainalluksella. Alkuun käyttämässäni työskentelymallissa jokaisen pienenkin koodimuutoksen käyttöönotto vaati useita hankalia rivikomentoja ja useamman minuutin odottelun ennen kuin päivitetty koodi oli palvelimella käytössä.

4.5.3 Hakulomakkeen käyttöliittymän rakentaminen



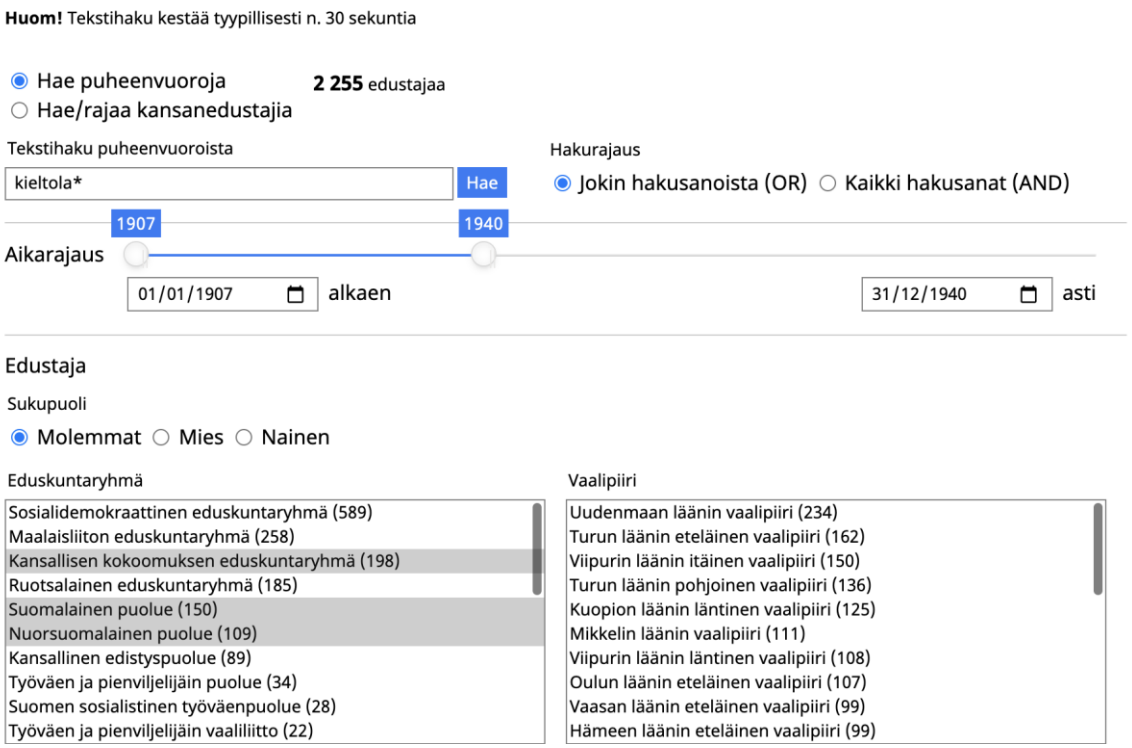
Kuva 10. Selainpohjaisen käyttöliittymän aloitusnäkymä.

Kun lopulta sain rajapinnan ja taustaohjelmiston alustavasti toimimaan, ryhdyin rakentamaan selainpohjaista käyttöliittymää, jonka avulla tietokantaan tallennetuista keskusteluista pystyisi tekemään hakuja. Toimiva käyttöliittymä sijaitsee verkko-osoitteessa http://demo.koponen-hilden.fi/eduskunta-keskustelut/

Käyttäjälle näkyvä osuus käyttöliittymästä syntyi kohtuullisen helposti, sillä minulla on runsaasti kokemusta selainkäyttöliittymien suunnittelusta ja teknisestä toteutuksesta. Sivu on toteutettu ilman verkkosovelluskehystä (*web framework*) kuten esim. Svelte, Vue.js tai React, suoraan html- ja javascript-koodina, koska tämä työskentelytapa on mielestäni joustavampi ja soveltuu hyvin kehittämistyöni tapaiseen pienimuotoiseen projektiin. Sivun elementtien päivityksessä on hyödynnetty D3.js-javascript-kirjastoa, jonka pääasiallinen käyttötarkoitus on yleensä tiedon visualisointi, mutta jonka ydinosan muodostavat

käyttöliittymän toteutuksessa hyödylliset ns. dokumenttioliomallin *(Document Object Model*) eli html-sivun sisältörakenteen muokkaukseen tarkoitetut toiminnot. D3.js sisältää lisäksi tietojen lukemiseen ulkoisesta lähteestä tarkoitettuja toimintoja, joita olen hyödyntänyt selainkäyttöliittymän ja RESTful-rajapinnan välisen liikenteen toteuttamiseen. Lisäksi olen käyttänyt noUISlider- ja wNumb-javascript-kirjastoja vaihteluvälisäätimen toteuttamiseen ja numeromuotoiluihin.

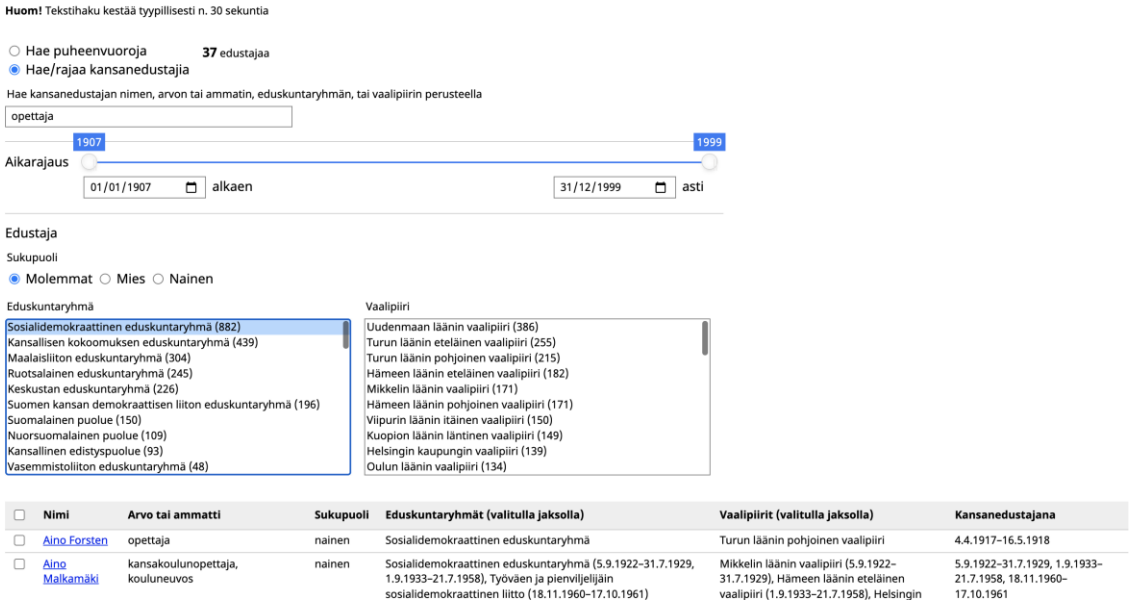
Käyttöliittymässä näkyvät, hakurajauksiin käytettävissä olevat eduskuntaryhmien ja vaalipiirien nimet haetaan tietokannasta sivua ladattaessa. Rajausvalinnat vaikuttavat toisiinsa siten, että esimerkiksi vain ne eduskuntaryhmät, joilla on ollut jäseniä valitulla aikarajauksella tulevat näkyviin Eduskuntaryhmä-valintalaatikkoon.



Kuva 11. Esimerkki hakurajauksesta.

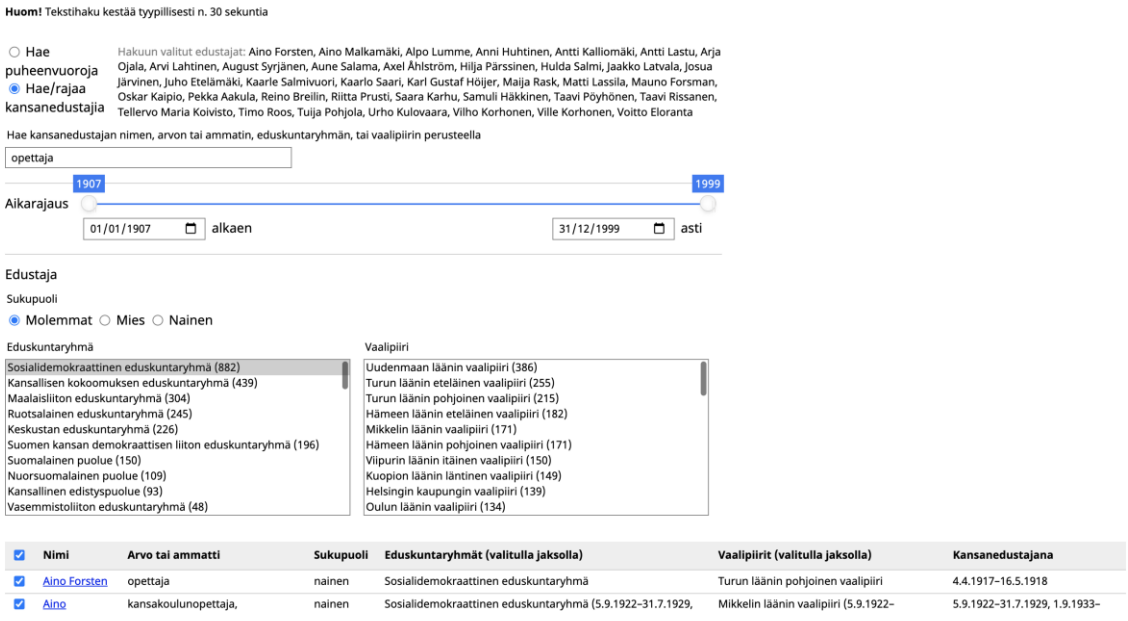
Hakurajauksessa tehdyt valinnat päivittyvät ha selaimen osoiteriville, joten haun voi helposti kopioida linkkinä jaettavaksi. Tämä mahdollistaa esimerkiksi sen, että juttuun sisällytetyistä taulukoista, grafiikoista ja muista elementeistä voi linkata suoraan siihen hakuun, jolla tiedot on kerätty.

Haku kohdistuu sekä sanojen alkuperäisiin että lemmattuihin muotoihin. Jokerimerkin * käyttö on myös sallittu, eli esim. hakuehto ”hallitu*” löytää kaikki ”hallitu”-alkuiset sanat (hallitus, hallituksen, hallitusmuoto, hallitusohjelman jne.).



Kuva 12. Esimerkki kansanedustajahausta.

Käyttöliittymä mahdollistaa myös kansanedustajista hakemisen samoja hakurajauksia käyttäen kuin varsinaisista puheenvuoroista haettaessa. Kansanedustajaluettelosta on myös mahdollista valita valintaruutujen (*checkbox*) avulla vain tietyt kansanedustajat tekstihaun kohteeksi.



Kuva 13. Esimerkki kansanedustajien valitsemisesta tekstihakuun.

Yksinkertaisuuden vuoksi käyttöliittymään ei sisälly minkäänlaista käyttöoikeuksien todennusta eli autentikointia esimerkiksi käyttäjätunnuksella ja salasanalla, vaan kuka hyvänsä, joka tietää palvelun osoitteen voi käyttää sitä. (Tietokantayhteyden käyttämä

käyttäjätunnus ja salasana on kuitenkin määritelty kiinteästi taustaohjelmistossa, eli niihin käyttäjä ei pääse käsiksi selaimensa kautta.)

4.5.4 Selaimen ja rajapinnan välinen yhteys

Itse SQL-muotoinen tietokantakysely muokataan pitkälti selaimessa. Esimerkiksi yllä (Kuva 4) näkyvän ruutukaappauksen mukainen hakurajaus lähettää seuraavanlaisen pyynnön rajapinnalle:

```
select=*&from=speech&params=INNER JOIN individual_word ON speech.speech_id =
individual_word.speech_id INNER JOIN parliamentary_group ON speech.mp_name =
parliamentary_group.mp_name WHERE speech.session_date >= '1907-01-01' AND
speech.session_date <= '1940-12-31' AND ((individual_word.lemmatized_word
LIKE 'kieltola%' OR individual_word.original_word LIKE 'kieltola%')) AND
parliamentary_group.parliamentary_group_name IN ('Kansallisen kokoomuksen
eduskuntaryhmä', 'Suomalainen puolue', 'Nuorsuomalainen puolue')
```

Kuten huomataan, SQL-kyselyn rakenne on pyynnössä lähes valmiina ja se välitetään rajapinnan kautta taustaohjelmistolle sellaisenaan. Tämä **ei** edusta ohjelmistokehityksen parhaita käytäntöjä: ratkaisu on tietoturvan kannalta potentiaalisesti ongelmallinen, sillä pahantahtoinen hyökkääjä voi ujuttaa kyselyyn mukaan vahingollista koodia, joka esimerkiksi vahingoittaa tietokantaa jollain tavoin. Oikeaoppinen tapa olisi muodostaa SQL-kysely vasta taustaohjelmistossa. Tätä tietoturvan kannalta ongelmallista ratkaisua olen yrittänyt paikata siten, että jos pyyntö sisältää puolipisteen[6], tai jonkin kiellettyjen komentojen luettelossa[7] mainitun merkkijonon, sitä ei suoriteta. Pidän silti varsin todennäköisenä, että tämän jälkeenkin kokonaisuuteen on jäänyt jokin tietoturva-aukko, jota hyökkääjän olisi mahdollista väärinkäyttää. Päädyin tähän ratkaisuun pitkälti siksi, että minulla on paljon enemmän selaimessa kuin palvelimella tapahtuvasta tiedon käsittelystä, joten tällaisen ratkaisun koodaaminen oli minulle helppoa. Koska palvelun väärinkäyttöpotentiaali on vähäinen, sen osoite ei ole julkisesti saatavilla ja kyseessä on opinnäytteenä tehty työ, joka poistuu verkosta työn arvioinnin jälkeen, tietoturvakysymykset eivät mielestäni ole demototeutuksen kannalta kriittisiä, mutta olen

[6] Puolipiste merkitsee SQL-kielessä komennon päättämistä, eli sen jälkeen alkaa uusi komento. Jos puolipiste olisi sallittu, hyökkääjä voisi sulkea tietokantakyselyyn käytettävän SELECT-komennon ja antaa sen jälkeen omia komentojaan tietokantaohjelmistolle.

[7] Kiellettyjen komentojen luettelossa ovat tärkeimmät tietokannan muokkaamiseen käytettävät komennot: DROP TABLE, ALTER TABLE, CREATE TABLE, INSERT INTO, UPDATE ja DELETE FROM.

hyvin tietoinen siitä, että jos kehittämistyöni pohjalta joskus toteutettaisiin julkiseen käyttöön tarkoitettu versio, tämä seikka olisi syytä ratkaista toisin.

Käyttöliittymän ja edellisessä alaluvussa käsitellyn RESTful-rajapinnan välisen tietoliikenteen toteuttaminen osoittautui luultua haastavammaksi ja sen toimimaan saaminen vaati laajoja muutoksia sekä rajapintaan että taustaohjelmistoon.

Aluksi koitin toteuttaa liikenteen hyödyntämällä http-yhteyskäytännön GET-pyyntöä, joka on teknisesti yksinkertaisin tapa ja tavanomainen menettely selaimen ja rajapinnan välisten kyselyiden toteuttamiseen. Tällöin kysely välitetään sellaisenaan url-verkko-osoitteen osana. Yksinkertainen rajapintakysely voisi näyttää esimerkiksi tältä:

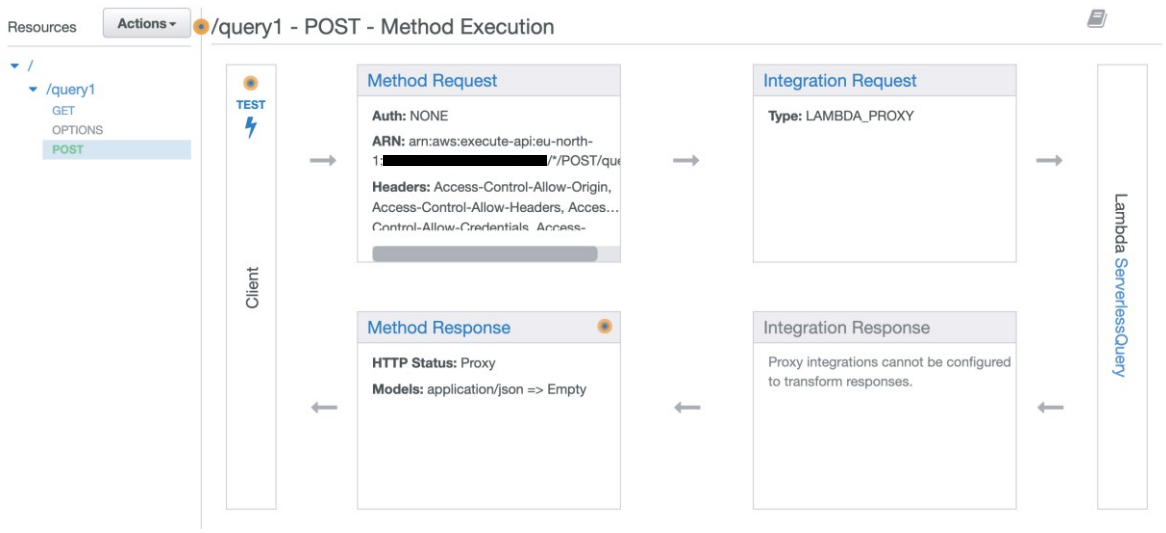
```
https://xxxxxxxxxx.execute-api.eu-north-1.amazonaws.com/beta/query1?
select=*&from=speech&params=WHERE%20individual_word.lemmatized_word='etyk'
```

Tässä kysymysmerkkiä edeltävä osuus on Amazonin API Gateway -palvelussa isännöidyn rajapinnan osoite (olen peittänyt ensimmäiset 10 merkkiä, jotka kertovat rajapinnan tarkan osoitteen Amazonin palvelimella) ja sitä seuraava osuus sisältää varsinaisen kyselyn. Kyselyn eri parametrit on erotettu toisistaan &-merkeillä, %20-merkkijono taas koodaa välilyöntiä, joka on kielletty merkki url-osoitteissa. Tämän osoitteen voi liittää sellaisenaan vaikkapa selaimen osoiteriville, jolloin rajapinta välittää kyselyn taustaohjelmistolle, joka suorittaa tietokantakyselyn ja palauttaa tulokset rajapinnan kautta suoraan selaimeen.

Ongelmaksi kuitenkin muodostuu, että varsinkin yksittäisten valittujen kansanedustajien puheita ja/tai monesta hakusanasta koostuvia hakuja haettaessa SQL-kyselyn pohjalta muodostetusta url-osoitteesta voi tulla niin pitkä, että se ylittää esimerkiksi Chrome-selaimen salliman maksimipituuden, 2 048 merkkiä (magichat 2021), eikä sitä pystytä enää välittämään GET-pyyntönä. Niinpä kysely on lähetettävä POST-pyyntönä, jossa maksimipituuden määrittelee palveluntarjoaja. Amazonin määrittelemä maksimikoko Lambda-funktioiden käsittelemille POST-pyynnöille on 6 megatavua eli 6 miljoonaa merkkiä (Amazon 2022).

Yhteystavan muuttaminen POST-pyynnöksi tuotti kuitenkin uusia ongelmia, jotka liittyvät eri lähteistä tulevien pyyntöjen jakamiskäytännön eli CORS:n (*cross-origin request sharing*) tietoturvasyistä asettamiin rajoitteisiin javascript-kielellä tehtyjen POST-kyselyiden käsittelyyn. (Helsingin yliopisto 2021; Jokimies 2016.) Jotta rajapinta suostuisi vastaamaan Amazonin sisäisen verkon ulkopuolelta tulevaan POST-pyyntöön ja selain käsittelemään saadun vastauksen, täytyy rajapinnasta ja taustajärjestelmästä määritellä joukko asetuksia, kuten pyynnön sallittu alkuperä, sallitut metodit ja sallitut otsikkokentät.

Lisäksi eräitä muitakin rajapinnan muita asetuksia oli säädettävä, jotta yhteys saatiin toimimaan.



Kuva 14. Rajapinnan asetukset Amazon API Gatewayn käyttöliittymässä. Rajapinnan yksilöivä tunniste on kuvassa peitetty mustalla palkilla.

Koska minulla ei ollut aiempaa kokemusta CORS-otsikkokenttien määrittelystä ja aiheesta verkosta löytämäni tieto oli ristiriitaista, tämä periaatteessa sinänsä yksinkertainen tehtävä muodostui yllättävän haastavaksi. Sain yhteyden lopulta toimimaan seuraavilla asetuksilla:

Rajapinnan asetukset:

```
Access-Control-Allow-Methods: GET, OPTIONS, POST
Access-Control-Allow-Headers: 'Content-Type,X-Amz-Date,Authorization,X-Api-
Key,X-Amz-Security-Token'
Access-Control-Allow-Origin: '*'
```

Taustajärjestelmän palauttamat http-otsikot:

```
"statusCode": 200,
"headers": {
    "Access-Control-Allow-Origin": "*",
    "Access-Control-Allow-Headers": "Content-Type",
    "Access-Control-Allow-Methods": "OPTIONS,POST,GET"
},
"isBase64Encoded": "false"
```

POST-pyyntö välittää siis selaimessa tehdyt hakuvalinnat miltei valmiina SQL-kyselynä taustaohjelmistolle, joka suorittaa tietokantakyselyn Pythonin psycopg2-moduulin avulla ja palauttaa kyselyn tulokset json-muodossa. JSON on yleisesti käytetty tekniikka tiedon siirtämiseen esimerkiksi selaimen ja palvelimen välillä. Tietorakenne koostuu

aaltosulkeiden väliin sijoitetuista nimi/arvo-pareista ja sen käsittely esimerkiksi selaimessa on helppoa. (Ylärinne 2013.)

Seuraava ratkaistava asia oli tämän json-muodossa olevan hakutuloksen välittäminen selaimelle. Tavallisempaa olisi ratkaista asia käsittelemällä hakutulokset jo taustaohjelmistossa ja palauttaa selaimelle vain sen verran tietoa, kuin käyttäjälle on tarkoitus näyttää, mutta koska olen tottuneempi käsittelemään tietoa selaimessa kuin palvelimella, päädyin hieman kömpelöön ratkaisuun, jossa tiedot välitetään eteenpäin sellaisenaan ja niiden käsittely esityskelpoiseen muotoon tapahtuu selaimessa.

Tässä ratkaisussa on kuitenkin sellainen ongelma, että tietokannasta saadut käsittelemättömät hakutulokset sisältävät tyypillisesti varsin paljon tietoa: taustaohjelmiston eteenpäin lähettämän json-tiedoston koko voi helposti olla useita kymmeniä megatavuja. Tämä on valitettavasti enemmän, kuin Amazonin Lambda-funktioiden POST-kyselyille asettama kuuden megatavun maksimi, joten hakutulosten palauttaminen kokonaisuudessaan ei onnistu http-yhteyskäytännön avulla.

Päädyin ratkaisemaan tämän ongelman siten, että hakutulokset tallennetaan json-tiedostona Amazonin S3-säilytyspalveluun ja vastauksena POST-kyselyyn selaimelle palautetaan ainoastaan linkki tähän tiedostoon, josta selain käy sen sitten noukkimassa. Tiedoston nimi muodostetaan käytettyjen hakukriteerien perusteella ja hakutuloksia säilytetään 24 tuntia haun tekemisestä. Jos käyttäjä tekee haun, jota vastaavan niminen tiedosto S3-palvelun hakutuloskansiosta jo löytyy, hakua ei suoriteta uudestaan, vaan käyttäjälle palautetaan aiemmin tehdyn haun tulokset, mikä tiputtaa hakuajan tyypillisestä noin puolesta minuutista noin yhteen sekuntiin. Tämä tarjoaa sivutuotteena kohtuullisen ratkaisun myös siihen ongelman, että mikäli haun käsittely Lambda-funktiossa kestää yli 29 sekuntia, Amazonin API Gateway katkaisee yhteyden automaattisesti, eikä tätä aikarajaa pysty pidentämään. Yhteyden katkaisu ei kuitenkaan keskeytä itse Lambda-funktiota, vaan sen jatkaa hakutulosten käsittelyä taustalla ja tallentaa valmistuneet tulokset json-tiedostoon kuten normaalistikin. Niinpä jos haku epäonnistuu aikakatkaisun vuoksi, käyttäjä voi odottaa hetken ja tehdä uuden haun samoilla kriteereillä, jolloin Lambda-funktio palauttaa jo aiemmin valmistuneet haun tulokset heti. Ratkaisun huonona puolena on mm. se, että koska S3 asettaa tiedostonimen ja sen sisältävän kansion nimen yhteispituudelle 255 merkin maksimirajan, aivan täyttä hakukriteeristöä ei välttämättä pystytä tiedostonnimeen tallentamaan, joten aiemmin tehdystä hausta vain hieman

poikkeavaa uusi haku voi virheellisesti palauttaa aiemman haun tulokset, jos eroavaisuudet hakukriteereissä eivät mahdu 241 ensimmäiseen merkkiin.[8]

Toinen keskeinen ongelma tässä lähestymistavassa liittyy siihen, miten Amazon laskuttaa S3-palvelun käytöstä: se perii jokaisesta tiedostolatauksesta pienen lähtevän liikenteen maksun (*egress fee*). Asialla ei ole suurta merkitystä käyttömäärien ollessa pieniä, mutta jos aktiivisia käyttäjiä olisi tuhansia, maksut saattaisivat nousta nopeasti suuriksi. Projektin mahdollisissa myöhemmissä iteraatioissa olisikin luultavasti perusteltua siirtää json-tiedosto selaimelle Lambda-funktion kanssa saman virtuaalisen yksityisen pilven (VPC, *virtual private cloud*) sisälle pystytetyn virtuaalisen Nginx-välityspalvelimen kautta, sen sijaan, että se ladataan suoraan S3:sta. Amazon ei nimittäin peri lähtevän liikenteen maksua virtuaalisen yksityisen pilven sisällä tapahtuvasta liikenteestä eikä Nginx-palvelimen ja ulkoisen verkon välisestä liikenteestä. (Jalonen 7.11.2021.)

4.5.5 Hakutulosten näyttäminen ja pöytäkirjojen selaus

Hakutulokset Lataa hakutulokset TSV-muodossa

Haku kesti 29,6 s.

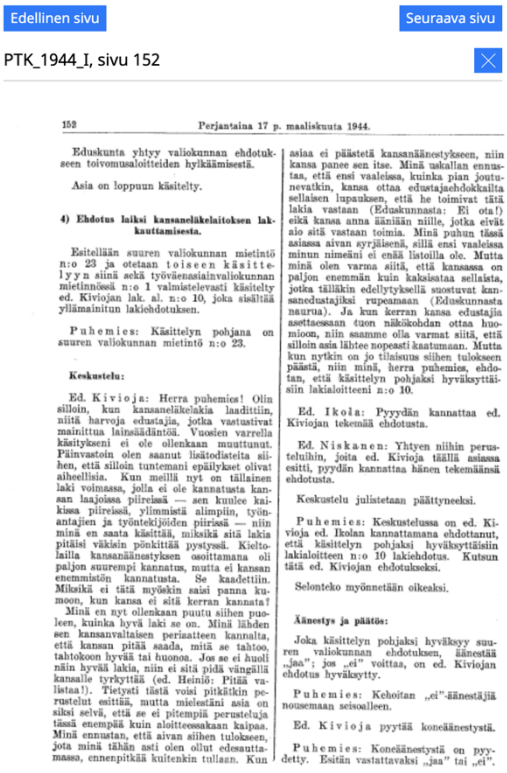
Hakutulosten määrä 364 kpl

Id	Päivämäärä	Pöytäkirja	Sivu	Edustaja	Eduskuntaryhmä	Vaalipiiri	Kieli	Puheenvuoro	Näytä sivu pöytäkirjassa
868	14.6.1907	PTK_1907_II	243	Nestor Huoponen	Nuorsuomalainen puolue	Viipurin läänin itäinen vaalipiiri	fi	Johdonmukaisena sille esittämälleni toivomukselle, että vain anomusehdotusten ponnet luettaisiin ja noudattaen ed. Räsäsen esimerkkiä, en tahdo vaivata eduskuntaa lukemalla niitä perusteluja, joita tämä esitysehdotus sisältää, sitä suuremmalla syyllä kun huomaan että tuskin /s jäsenistä on täällä saapuvilla ...	
870	14.6.1907	PTK_1907_II	244	J.K. Paasikivi	Suomalainen puolue	Turun läänin pohjoinen vaalipiiri	fi	Minä olen aikoinani ollut jonkun verran osallisena sen ehdotuksen tekemisessä tai oikeastaan tarkastamisessa, jonka pohjalle nämä kaikki esitysehdotukset ovat valmistetut ja joiden joukossa myöskin on se esitysehdotus, jonka ed. Soininen julkiluki, ja joka on meidän ryhmästä annettu. Kun kuitenkin muiden ...	
875	14.6.1907	PTK_1907_II	247	Ernst Palmén	Suomalainen puolue	Uudenmaan läänin vaalipiiri	fi	Niin suurella mielenkiinnolla kuin seurasin ed. Paasikiven esitystä, yhtä vähän minä käsitin, mitä aihetta ed. Nuortevalla oli muistutukseensa. Asianlaita on se, että jokainen eduskunta maailmassa, kun ratkaisu on kysymyksessä, on pakotettu keskustelemaan ja esittämään syitä. Kun asioita on lähetettävä ...	
877	14.6.1907	PTK_1907_II	248	J.K. Paasikivi	Suomalainen puolue	Turun läänin pohjoinen vaalipiiri	fi	Minä luulen, että ed. Valpas on sinne taakse kuullut aivan väärin, mitä minä puhuin. Minä en ole tahtonut ottaa meidän puolueellemme kieltolakiasiassa mitään kunniaa. Minä tiedän sen, että sosiaalidemokraattinen puolue on ensimäisenä ottanut kieltolakiasian ajaakseen. EFi tässä ollut ollenkaan puhe ...	
878	14.6.1907	PTK_1907_II	248	Nestor Huoponen	Nuorsuomalainen puolue	Viipurin läänin itäinen vaalipiiri	fi	Pyydän anteeksi, että talousvaliokunnan jäsenenä, johon nämät eduskuntaesitykset lähetetään, katson velvollisuudekseni pyytää puoheenvuoroa, koska täällä näkyy tulevan riita kunniasta, kuka ennemmin ja enemmän on kieltolakiasiaa ajanut. Minä puolestani en tahtoisi ed. Valppaankaan väitteelle myöntää ...	
879	14.6.1907	PTK_1907_II	249	Kaarlo Kares	Suomalainen puolue	Mikkelin läänin vaalipiiri	fi	Kyllä on koko lailla joutavaa, että ne miehet, jotka ovat olleet raittiusväen riveissä, rupeavat tänä hetkenä, jolloin todella ensi kerran yleinen kieltolakiesitys on esitettävä, väittelemään: minä olen enemmän tehnyt kuin sinä. Minä huomautan vain, että koko raittiuskansa on ollut aivan	

Kuva 15. Esimerkki hakutuloksista selainkäyttöliittymässä.

Onnistuneen haun jälkeen käyttöliittymä näyttää hakutulokset taulukkona, jota voi tarkastella selaimessa (ks. kuva 14 yllä) tai ladata tiedostona omalle koneelleen.

[8] Tämä merkkimäärä (241) on 255 vähennettynä json-tiedoston sisältävän S3-kansion koko polun merkkimäärällä (14).



Kuva 16. Pöytäkirjan sivu selausnäkymässä.

Taulukon oikeanpuolimmaisessa sarakkeessa näkyy myös pienoiskuva siitä skannatusta sivusta, jolta riville kirjattu puheenvuoro alkaa. Kuvan klikkaaminen avaa *selausnäkymän*, jonka tarkoituksena on mahdollistaa alkuperäisen dokumentin relevantin kohdan tarkastelu. Sivun ylälaidan nappien avulla voidaan skannattua pöytäkirjaa selata eteen- tai taaksepäin.

Koska aineistossa on runsaasti tekstintunnistuksessa ja tekstin käsittelyssä tapahtuneita virheitä, alkuperäisten skannattujen dokumenttien tarkastelumahdollisuus toimii hyödyllisenä apuvälineenä hakutulosten validoimiseen. Lisäksi se vahvistaa verkkopalvelun käyttökelpoisuutta ”tavanomaisen” tekstihaun välineenä sen lisäksi, että hakutulosten massalataus mahdollistaa erilaiset datajournalistiset lähestymistavat aineistoon.

5 Toteutuksen ja tulosten arviointi

5.1 Datajournalististen juttujen laatiminen tietokannan tietojen pohjalta

Testasin rakentamani tietokannan käyttökelpoisuutta journalistisen työn apuvälineenä keräämällä sen avulla datan kahta erityyppistä datajournalistista juttukokonaisuutta varten.

5.1.1 Viestintäteknisten termien maininnat suuressa salissa

Ensimmäinen datajournalistinen juttu on interaktiiviseksi suunniteltu aikajana eri viestintäteknologioiden maininnoista eduskunnan täysistuntojen suomenkielisissä keskusteluissa ja siihen liittyvä lyhyt juttuteksti. Staattinen versio visualisoinnista on nähtävissä tämän kehittämistyön liitteessä 3.

Keräsin juttua varten mainintoja eri viestintävälineistä ja -tekniikoista alaluvussa 4.5.3 esiteltyä hakulomaketta hyödyntäen. Käytin jokerihakua (esim. ”sanomaleh*” ja ”matkapuheli*”), jotta myös mahdolliset virheellisesti perusmuotoistetut muodot sanoista tulisivat mukaan aineistoon. Rajasin pois ruotsinkieliset puheenvuorot ja merkitykseltään ilmeisen eriävät muodot (esim. ”radio*” haku löytää myös radioaktiivisuutta koskevia mainintoja ja ”posti*” tarttuu myös esim. Postipankkia ja postiryöstöjä käsitteleviin keskusteluihin). Sen jälkeen yhdistin samaa merkitsevät hakusanat (esim. ”puhelin” ja ”telefoni”) ja laadin grafiikan, joka näyttää eri välineiden maininnat kunakin vuonna, sekä niiden välineiden, joita ei esiinny vielä vuoden 1907 pöytäkirjassa ensimmäiset maininnat.

Kuviossa vasemmalla ovat perinteiset, paperimuotoiset viestintävälineet ja oikealla sähköiset. Ajatuksena on, että jutun interaktiivisessa versiossa käyttäjä saisi klikkaamalla nähtäväkseen myös jokaisen vuoden kunkin välineen yksittäiset maininnat, ja desktop-versiossa hiiren päälle vieminen korostaisi sarjan, kertoisi sen nimen ja mainintojen määrän tarkasteluvuonna.

Visualisointia varten tehty tiedonkeruu antoi mahdollisuuden validoida tietokannan sisältöä suhteessa alkuperäisiin, tekstintunnistuksen käyneisiin pöytäkirjoihin. Keräsin samat tiedot ohjelmallisesti myös suoraan alkuperäisistä, tekstintunnistuksen läpi käyneistä pöytäkirjoista sekä puheenvuoroiksi jäsennellyistä TSV-tiedostoista (ks. alaluvut 4.3.1–4.3.2). Tulokset ovat nähtävissä taulukossa 1, eivätkä ne valitettavasti ole järin mairittelevat.

Taulukko 1. Viestintäteknisten termien runko-osien maininnat tekstintunnistuksen läpi käyneissä pöytäkirjoissa, jäsennellyissä TSV-tiedostoissa ja tietokannassa.

Sana	Maininnat pöytäkirjoissa	Maininnat TSV:ssä	Maininnat tieto-kannassa	Muutos pöytäkirja → TSV	Muutos TSV → tietokanta
posti	26 467	8 425	7 561	-68 %	-10 %
kirje	26 194	7 083	6 345	-73 %	-10 %
sanomaleh	12 653	4 778	4 667	-62 %	-2 %
radio	8 842	2 683	2 353	-70 %	-12 %
puheli	8 547	2 500	2 404	-71 %	-4 %
tv	7 666	640	293	-92 %	-54 %
televisio	6 268	2 444	2 252	-61 %	-8 %
elokuv	5 292	1 723	1 668	-67 %	-3 %
lehdistö	4 483	1 607	1 570	-64 %	-2 %
lennät	3 820	790	715	-79 %	-9 %
film	1 628	676	554	-58 %	-18 %
video	1 513	630	588	-58 %	-7 %
kirjapaino	1 388	400	388	-71 %	-3 %
telefo	937	318	42	-66 %	-87 %
aikakausleh	664	244	232	-63 %	-5 %
sähkösanom	488	188	187	-61 %	-1 %
internet	208	95	80	-54 %	-16 %
iltapäiväleh	195	96	94	-51 %	-2 %
kännyk	118	71	68	-40 %	-4 %
teleks, telex	115	23	21	-80 %	-9 %
sähköt	93	26	26	-72 %	0 %
radiopuhel	81	31	30	-62 %	-3 %
sähke	80	32	32	-60 %	0 %
matkapuheli	55	31	31	-44 %	0 %
sähköpost	41	25	25	-39 %	0 %
painokone	28	7	7	-75 %	0 %
faks, fax	38	16	16	-58 %	0 %
telefaks, telefax	29	11	10	-62 %	-9 %
tietokonepeli	4	2	2	-50 %	0 %
autopuheli	2	2	2	0 %	0 %
tekstiviest	1	1	1	0 %	0 %

Koska maininnat on kerätty teknisesti erilaisilla menetelmillä, ei ole syytäkään olettaa, että lukemat vastaisivat toisiaan täsmällisesti. TSV-tiedostoista löytyvien mainintojen tietokannasta poimittuja mainintoja suurempi määrä selittynee pääosin sillä, että ruotsinkieliset puheenvuorot on rajattu viimeksi mainitusta aineistosta pois. Aineiston siirrossa TSV-tiedostoista tietokantaan ja tietokannan siirrossa Amazonin pilveen ei siis vaikuta tapahtuneen ainakaan merkittävää tiedonhukkaa ja selainkäyttöliittymän kautta käytetty hakutoiminto vaikuttaa myös löytävän jokseenkin kaikki kantaan tallennetut tiedot.

Sen sijaan alkuperäisten pöytäkirjojen mainintojen määrä on kautta linjan huomattavasti suurempi kuin TSV-tiedostoihin kertyneiden mainintojen. Osatekijänä on varmasti se, että monet sanat kuten ”posti” tai ”sähke” voivat esiintyä pöytäkirjoissa myös keskustelujen ulkopuolella, esim. kirjallisissa kysymyksissä, lakialoitteissa ja muissa pöytäkirjaan merkityissä asialistan kohdissa. Mutta myös esimerkiksi täysin puhekielisen ”kännykkä”-sanan, joka tuskin esiintyy muutoin kuin keskusteluissa, maininnoista 40 % puuttuu TSV-tiedostoista. Tämä kertoo uskoakseni siitä, että alaluvuissa 4.3.1–4.3.2 kuvailtu puheenvuorojen erottelu muusta tekstistä on osin epäonnistunut eivätkä läheskään kaikki puheenvuorot ole tulleet kirjatuiksi TSV-tiedostoihin kuten olisi pitänyt, ja/tai merkittävä osuus puheenvuoroista on kirjautunut vain osittain – siis katkennut kesken. Oli varsin kiusallista huomata tämä keskeinen virhe vasta kehittämistyön ollessa lähes valmis, jolloin aikataulu ei enää mahdollistanut sen korjaamisen vaatimaa todennäköisesti varsin suuren työmäärän käyttöä. Jälkiviisaasti voi todeta, että eri työvaiheiden välillä siirtymiseen olisi aina pitänyt yhdistää pistokoeluonteisia kokeiluja laajempi aineiston eheyden validointi.

Koska hakutulokset ovat selvästi puutteellisia, visualisointi ei todennäköisesti anna täysin oikeaa kuvaa viestintäteknologioiden maininnoista eduskunnan keskusteluissa, eikä siten soveltuisi julkaistavaksi.

5.1.2 Mistä ja miten eduskunnassa puhutaan?

Toinen datajournalistinen kokonaisuus käsittelee politiikan keskustelun teemoja ja keskustelutapoja. Kyseessä on vasta juttuhahmotelma – lopullinen versio on ajateltu aikakauslehtityyppiseksi pitkähköksi jutuksi, jossa datan perusteella tehtyjä havaintoja avattaisiin suorien sitaattien ja asiantuntijahaastattelujen avulla. Juttuun liittyvää taustatyötä en kuitenkaan ole tehnyt, eikä juttua ole kirjoitettu. Liitteessä 4 on sen sijaan muutamia visualisointeja/taulukoita, joista jutun datajournalistinen sisältö hahmottuu.

Tätä kokonaisuutta varten data on kerätty selainkäyttöliittymän sijaan suoraan tietokannasta SQL-hakujen avulla. Esimerkiksi kunkin vuoden suhteellisesti eniten mainittu sana on selvitetty näin:

```
SELECT lemmatized_word, COUNT(*) AS mentions
FROM individual_word INNER JOIN speech
ON speech.speech_id = individual_word.speech_id
AND speech_lang = 'fi' AND NOT stopword
AND session_date >= 'vuosi-01-01' AND session_date <= 'vuosi-12-31'
GROUP BY lemmatized_word ORDER BY mentions DESC LIMIT 5000;
```

Tietokannasta poimitaan siis kaikkien suomenkielisten puheenvuorojen sellaiset sanat, jotka eivät ole hukkasanoja. Haku toistetaan kullekin vuodelle 1907–1999 siten, että teksti ”vuosi” korvataan yllä olevassa hakulausekkeessa kyseisellä vuodella. Tulokset tallennetaan ja käydään läpi skripteillä, jotka laskevat kunkin vuoden *suhteellisesti* yleisimmin esiintyvän sanan ja kunkin puolueen suhteellisesti eniten käyttämät 10 sanaa. Hakutuloksissa huomioidaan siis toisaalta sanojen absoluuttisten mainintojen määrä kunakin vuonna ja kunkin puolueen edustajien puheenvuoroissa, toisaalta se, kuinka iso osuus kaikista sanan maininnoista on tapahtunut kyseisenä vuonna tai kyseisen puolueen edustajien toimesta. Tarkastelun ulkopuolelle on jätetty pois sellaiset sanat, jotka eivät ole tarkastelun kannalta kiinnostavia: hukkasanojen lisäksi listalta on pudotettu kansanedustajien nimet sekä joukko eduskunnan arkeen keskeisesti liittyviä termejä kuten *hallitus, laki, kysymys, mietintö, valiokunta* jne. Puolueiden top 10 -luetteloista on jätetty lisäksi pois sanat, jotka viittaavat edustajan omaan puolueeseen.

Huolimatta tiedossa olevista puutteista aineistossa, joita on käsitelty laajemmin edellisessä alaluvussa listoista piirtyy mielenkiintoinen kuva politiikan isoista teemoista yli 90 vuoden ajalta. Kriisiaikojen ja suurten murrosten kuten Suomen EU-jäsenyyden alkuvuosien ulkopuolella eduskuntaa tuntuu puhuttavan talous- ja elinkeinopolitiikka, varsinkin metsä- ja maatalous, sekä vähäisemmässä määrin koulutus ja päivähoito. Puoluekohtaiset teemoissa erot ovat suuria – vain SDP:llä ja kokoomuksella on top 10 -sanoissa selvää päällekkäisyyttä. Suurelta osin erot ovat arvattavia – jopa kliseisiin asti –, mutta yllätyksiäkin mahtuu joukkoon: esimerkiksi se, että toinen puolue (SKP) nousee RKP:n suhteellisesti suosituimpien sanojen joukkoon. (Tässä voi tosin taustalla vaikuttaa se, että RKP:llä suomenkielisten puheenvuorojen osuus on pieni, joten sattumalla voi olla osuutensa asiaan.)

Olin kiinnostunut myös kysymyksestä, löytyisikö puolueiden välille eroja paitsi keskustelun teemoissa, myös siinä, millaista kieltä käytetään: esimerkiksi käyttääkö oikeisto tai vasemmisto enemmän tai vähemmän adjektiiveja tai verbejä. Alustavassa analyysissä kuitenkaan kovin merkittäviä puoluekohtaisia eroja ei useimmista tutkimistani puheiden kieliopillisista piirteistä löytynyt. Osasyynsä voi olla sillä, että useimpien puolueiden aikasarjat ovat hyvin pitkiä, jolloin yleinen muutos kielessä peittää alleen pienemmät erot puolueiden tavoissa puhua.

Suurimmat aineistosta löytämäni erot puhetavoissa olivat persoonapronominien käytössä. Ensimmäisessä persoonassa puhuivat lähes yhtä paljon keskusta/maalaisliitto, RKP, vihreät ja kokoomus, vähiten vasemmistoliitto/SKDL ja perussuomalaiset/SMP. Eniten muista puolueista poikkesivat RKP ja kristillisdemokraatit/SKL. Erityisesti viime mainittu yllättää: kristilliset käyttivät vähiten minä- ja eniten me-muotoa. Myös eduskunnassa harvoin kuultua sinä-sanaa he viljelevät tuplasti useammin kuin muut – ehkä Raamatun lainauksissa?

Myös tämän aineiston keruu paljasti selviä virheitä tietokannan sisällössä. Esimerkiksi kaikki pronominien (myös muut kuin persoonapronominit) maininnat kerännyt SQL-kysely löysi tietokannasta kaikkiaan 999 eri FinnPos-jäsentimen pronominiksi päättelemää sanaa, vaikka suomen kielessä on vain noin 50 eri pronominia. Suurimmalle osalle niistä löytyy aineistosta vain kourallinen mainintoja, mutta esimerkiksi sanan *edustaja* FinnPos on kontekstin perusteella tulkinnut pronominiksi yli 125 000 kertaa – useammin kuin esimerkiksi indefiniittipronominit *sama* ja *eräs* esiintyvät aineistossa. Toisaalta tämä on sinänsä mielenkiintoinen havainto, joka paljastaa myös todellisen piirteen eduskunnan kielenkäytöstä: sanaa *edustaja* todella käytetään täysistuntokeskusteluissa usein kuin persoonapronominia!

5.2 Tietokannan käyttökelpoisuuden arviointi juttuprosessin aikana tehtyjen havaintojen perusteella

Tietokannan ja sen käyttöliittymän todellinen tulikaste tapahtui siis vasta kehittämistyön loppupuolella, kun sovelsin kantaan tallennettuja tietoja ensi kertaa journalistiseen käyttöön. Aiemmin tehdyt pistokoeluontoiset kokeilut aineistolla ja hakukäyttöliittymällä olivat paljastaneet joitain ongelmia, joista suurimman osan pystyin prosessin aikana korjaamaan. Vasta tietokannan hyödyntäminen juttuprosessissa paljasti huomattavia puutteita, joita olen käsitellyt edellä.

Aineiston puutteet ovat osin huomattavia, kuten esimerkiksi Taulukko 1 ilmenee – niin huomattavia, että olen taipuvainen ajattelemaan, ettei tietokanta tällaisenaan sovellu datajournalistiseen käyttöön. Haut tuottavat toki suuntaa antavia tuloksia esimerkiksi eri termien suhteellisesta esiintyvyydestä eri rajauksin, mutta mitä täsmällisempiin kysymyksiin niillä yritetään etsiä vastausta, sitä isommaksi ongelmaksi muodostuu, että kantaan tallennettu tieto on isolta osin puutteellista ja virheellistä.

Hakulomake toimii mielestäni sinänsä kohtuullisen hyvin. Suurimpana puutteena on hidas vasteaika, joka käsitykseni mukaan johtuu suurelta osin kustannussyistä valitusta matalasta palvelutasosta (ks. alaluku 4.5.1) ja tietokannan melko suuresta koosta, sekä

vähäisemmässä määrin taustaohjelmiston hieman kömpelöstä teknisestä toteutuksesta (alaluku 4.5.2). Haun suorittaminen kestää tyypillisesti hieman alle 30 sekuntia per haku, mikä on varsin hidasta verrattuna niihin hakutoiminnallisuuksiin, joihin käyttäjät ovat tavallisesti tottuneet. Muutoin käyttöliittymä toimii hyvin ja haku tuottaa huomattavasti parempia tuloksia kuin eduskunnan verkkosivujen oma hakukone. Tämä yhdistettynä mahdollisuuteen tarkistaa puheenvuoron alkuperäinen konteksti selausnäkymän (ks. alaluku 4.5.5) kautta tekee mielestäni siitä käyttökelpoisen perinteisemmässä journalistisessa tiedonhankinnassa, joka ei nojaa aineiston massalatauksiin.

Myös tietokannan käyttö suoraan SQL-kielen avulla sujuu mainiosti. Omalla koneella pyöritettynä tietokanta toimii nopeasti ja valittu tietokantarakenne sopii isoihin poimintoihin hyvin. Toimituskäytössä SQL-hakuihin perustuva lähestymistapa mahdollistaa monipuolisemmat haut kuin verkkolomake, etenkin silloin kun tarkoituksena on tarkastella aineistoa nimenomaan datajournalismin menetelmin.

Näiden kahden datajournalistisen juttuhahmotelman tekeminen paljasti siis merkittäviä laadullisia puutteita siinä, miten tekstin käsittely ja analysointi on onnistunut. Toisaalta se mielestäni myös osoitti, että käytetty lähestymistapa on periaatteellisella tasolla toimiva: aineiston kerääminen kumpaakin juttua varten sujui erittäin helposti. Jos aineiston käsittely olisi onnistunut paremmin, tietokanta ja sen selainpohjainen käyttöliittymä toimisivat erinomaisesti journalistisen tiedonhankinnan välineinä.

6 Yhteenveto ja pohdintaa

Pohdin tässä, kehittämistyöni viimeisessä luvussa, miten hyvin työlle asetetut tavoitteet ovat toteutuneet ja millaisia vastauksia tutkimuskysymyksiini työ on antanut. Kuvailen myös, miten toimisin vastaavassa hankkeessa nyt toisin kaiken prosessin aikana oppimani valossa.

6.1 Asetettujen tavoitteiden toteutuminen

Olen asettanut kehittämistyölleni seuraavat tavoitteet:

- Tutkia, millainen prosessi eduskunnan keskustelupöytäkirjojen analysointi ja siirtäminen hakutoiminnolla varustettuun tietokantaan olisi
- Dokumentoida työn vaiheet ja vaikeudet niin, että kehittämistyötä voidaan käyttää myöhemmin vastaavan projektin apuna ja välttää mahdolliset prosessissa itse tekemäni virheet
- Saada aikaiseksi toimiva verkkopalvelu, jonka avulla voi tehdä hakuja eduskunnan keskustelupöytäkirjoista ja ladata hakutulokset massa-aineistona
- Oppia hyödyntämään luonnollisen kielen käsittelyn menetelmiä ison aineiston käsittelyyn
- Oppia pystyttämään ja hallinnoimaan tietokantaa ja tekemään tietokantakyselyitä SQL-kielen avulla
- Oppia, miten rakennetaan pilvipalveluna toimivaa tietokantaa hyödyntävä verkkopalvelu

Arvioin seuraavaksi, miten hyvin nämä tavoitteet ovat työssä mielestäni toteutuneet.

Ensimmäiset kaksi tavoitetta ovat mielestäni toteutuneet melko hyvin. Taustatutkimus ja työn konkreettinen toteutus ovat antaneet minulle hyvän käsityksen siitä, mitä kaikkia työvaiheita, tehtäviä, haasteita ja virhemahdollisuuksia keskustelupöytäkirjojen analysointi, käsittely ja tallentaminen tietokantaan sisältävät ja vaativat, ja olen kirjannut nämä tiedot ylös kehittämistyöni loppuraporttiin. Prosessin oheistuotteena on syntynyt runsaasti koodia, joka toimii myös omanlaisenaan dokumentaationa.

Kehitystyölle asetetuista tavoitteista heikoimmin on mielestäni onnistunut toimivan hakupalvelun kehittäminen. Vaikka tällainen palvelu syntyikin kehitystyöprosessin tuloksena ja on tätä kirjoittaessa toiminnassa, se on kuitenkin monin tavoin puutteellinen, kuten olen edellä (alaluvussa 5.2) kirjoittanut. Alkuperäisenä toiveena oli tuottaa verkkopalvelu, joka olisi mahdollista avata kenen hyvänsä kiinnostuneen vapaasti käytettäväksi, mutta aiemmin kuvatuista virheistä ja puutteista johtuen en pidä nykyisen version julkista levittämistä tarkoituksenmukaisena. Itse prosessin lopputuote ei siis ole laatutasoltaan sellainen, kuin kehittämistyötä aloittaessani tavoitteenani oli. Kuten olen aiemmin todennut, verkkopalvelun kehittäminen on kuitenkin vain yksi kehittämistyölle

asettamistani tavoitteista, enkä pidä työtä kokonaisuudessaan epäonnistuneena, vaikken lopulta onnistunutkaan siinä niin hyvin kuin olisin toivonut.

Kehittämistyölle asettamieni oppimistavoitteiden katson toteutuneen hyvin. Olen oppinut erittäin paljon luonnollisen kielen menetelmistä ja niiden soveltamisesta datajournalistiseen käyttöön. Olen myös onnistuneesti suunnitellut ja pystyttänyt tietokannan ja oppinut käyttämään SQL-kieltä tietokantakyselyiden toteuttamiseen. Myös kolmas oppimistavoite, oppia rakentamaan pilvipalveluna toimivaa tietokantaa hyödyntävä verkkopalvelu on toteutunut kohtuullisesti, joskin osin negaation kautta – nyt tiedän paljon etenkin siitä, miten tällaista palvelua *ei* kannata toteuttaa.

6.2 Vastaukset tutkimuskysymyksiin

Esitin kehittämistyöni alussa seuraavat tutkimuskysymykset, joihin pyrin seuraavassa työn nyt valmistuttua vastaamaan:

- Millaisia taitoja ja välineitä skannatuista asiakirjoista koostuvan aineiston siirtäminen tietokantamuotoon vaatii?
- Miten journalistit voivat hyödyntää luonnollisen kielen käsittelyä ja tietokantoja omassa työssään?
- Kuinka hyvin tietokanta soveltuu journalistisena sisältönä julkaistavaksi?

Ensimmäiseen kysymykseen voinee perustellusti vastata monella eri tavoin, sillä samaan lopputulokseen voidaan päästä monilla eri välineillä. Oma lähestymistapani nojasi vahvasti Python-ohjelmointikielen ja sen eri moduulien eli laajennusosien käyttöön. Käytin tiedonharavointiin selenium- ja requests-moduuleita, tekstintunnistukseen PyMuPDF- ja pytesseract-moduuleita, tekstin jäsentämiseen ja perusmuotoistamiseen FinnPos-ohjelmistoa Pythonin subprocess-moduulin avulla (ks. tarkemmin alaluku 4.3.3), sanojen ryvästämiseen Word2Vec-algoritmia gensim-moduulin avulla, tietokantayhteyksiin psycopg2-moduulia ja erilaisiin avustaviin toimintoihin pandas-, csv-, os-, sys-, re-, datetime-, string-, math- ja configparser-moduuleita. Pythonin lisäksi tarpeellisia teknisiä taitoja olivat SQL-kielen sekä ns. säännöllisten lausekkeiden (regex) perusteiden hallinta. Huomattavaa etua oli myös Suomen poliittisen historian ja kulttuurin tuntemuksesta ja hyvästä suomen ja tyydyttävästä ruotsin kielen taidosta, jotka auttoivat monissa kohdin huomaamaan ja korjaamaan koodin tekemiä virheitä ja täydentämään puuttuvia tietoja.

Kehittämistyöprosessi vahvisti käsitystäni, jonka mukaan luonnollisen kielen käsittely on potentiaalisesti erittäin hyödyllinen – vaikka toistaiseksi varsin vähän käytetty – menetelmä journalistisessa työssä. Toimituksellisessa työssä on hyvin tavallinen tilanne, että käytettävissä ei ole hyvin jäsenneltyä tilastollista aineistoa vaan sen sijaan kaoottinen

kokoelma rakenteetonta tietoa, esimerkiksi suuri määrä tietovuodon kautta haltuun saatuja asiakirjoja. Luonnollisen kielen käsittelyn menetelmät tarjoavat journalistille mahdollisuuden nopeuttaa isojen dokumenttimassojen läpikäyntiä, oli tarkoitus sitten etsiä raskauttavia todisteita taloudellisista väärinkäytöksistä tai vaikkapa selvittää, mitä veteraanikansanedustaja on puhunut 20 vuotta sitten nyt uudestaan pinnalla olevasta aiheesta. Datajournalismin näkökulmasta menetelmä taas tarjoaa mahdollisuuden muuttaa tekstimuotoista tietoa numeroiksi ja muiksi datarakenteiksi, jotka soveltuvat massa-analyysin, visualisoinnin ja muiden datajournalismin menetelmin keinon käsiteltäväksi.

Olen työskennellyt pitkään erilaisten datasettien kanssa, mutta aiemmin käyttämäni aineistot ovat lähes aina olleet tiedostomuotoisia. Työskentely tietokantamuotoon tallennetun tiedon kanssa oli silmiä avaava kokemus. Kun kyseessä on tässä kehitystyössä käsitellyn kaltainen, huomattavan suurikokoinen datasetti, sen pyörittely sujuu SQL-kielen avulla huomattavasti tiedostomuotoa kätevämmin ja etenkin nopeammin. Koko aineistosta tehtävien poimintojen läpimenoaika pienentyi tietokantamuodossa pieneen murto-osaan siitä, mitä se olisi ollut, jos tieto olisi ollut käytettävissä vain kasana tiedostoja. Tämän kokemuksen perusteella uskaltaisin sanoa, että tietokantojen ja SQL-kielen perusteiden tuntemuksesta olisi datajournalistien lisäksi hyötyä ainakin kaikille tutkiville journalisteille, mutta varmasti myös monille muille isojen tietomäärien kanssa työskenteleville.

Kolmas, tietokannan roolia itsenäisenä journalistisena sisältönä koskeva tutkimuskysymys ei kenties vielä saanut lopullista vastaustaan tämän kehittämistyön puitteissa, sillä edellä käsiteltyjen puutteiden vuoksi kehittämistyön lopputuloksena syntynyt tietokanta ja sen verkkokäyttöliittymä eivät mielestäni sovellu yleisölle julkaistavaksi. Voidaan kuitenkin pohtia, mikä vastaus kysymykseen olisi, jos tietokannan kattavuus olisi kunnossa ja haku toimisi nopeammin.

Vaikka Suomessa on jossain määrin vakiintunut perinne tietokantojen julkaisemisesta sellaisenaan journalistisena sisältönä (erilaiset ”koneet” kuten Verokone, Ylioppilaskone jne.), toistaiseksi näin julkaistut tietokannat ovat olleet yleensä rakenteeltaan ja käyttöliittymältään varsin yksinkertaisia. Kehittämistyössäni laadittu hakutyökalu on ominaisuuksiltaan monipuolisempi, mutta siten valitettavasti myös monimutkaisempi käyttää kuin useimmat suomalaisten tiedotusvälineiden julkaisemat tietokannat, mikä uskoakseni rajaa selvästi sen potentiaalista käyttäjäkuntaa. Vaikka hakukone voi olla erittäin kiinnostava politiikan historiasta syvällisesti kiinnostuneelle lukijalle, keskivertolukija tuskin vaivautuisi sen parissa kovin pitkää hetkeä viettämään, koska

tietokanta ei useimmille tarjoa samanlaista henkilökohtaisesti kiinnostavaa tulokulmaa kuin vaikkapa naapurin tietojen vilkuilu Verokoneesta tai sitten konkreettista hyötyä kuten vaikkapa asuntokauppojen tarkkoja hintatietoja sisältävä tietokanta. Parhaiten hakukäyttöliittymä toimisi uskoakseni verkossa julkaistujen juttujen ”laajennuksena”: jutun tekstistä, taulukoista ja grafiikoista voitaisiin linkata suoraan hakuun, jolla niissä esitettävät tiedot on saatu ja mahdollistaa syvemmälle porautuminen sille lukijoiden vähemmistölle, joka tällaisesta on kiinnostunut. Suurempi merkitys tietokannalla olisi kuitenkin uskoakseni toimituksen sisällä tiedonhankinnan työkaluna sekä muutamien erityisryhmien kuten tutkijoiden ja historianopettajien palvelemisessa. Klikeillä mitaten hakukone tuskin olisi iso hitti, mutta julkisen palvelun toimijalle tai muutoin yhteiskunnallisesti vastuulliselle mediatoimijalle tämän kaiken mahdollistamisella saattaisi olla itseisarvo, jota ei voi suoraan mitata lukijamäärien kautta. Joka tapauksessa on hyvä muistaa, että tietokone yksinään ei synnytä journalismia, vaan journalististen tekojen takana on aina ihmisen työtä niiden ulkoisesta muodosta ja teknisestä rakenteesta riippumatta. ”Teknologia ei itsessään tee mitään. On oltava juttuidea ja taidot ja kiinnostus, jotta teknologia saadaan käyttöön”, kuten Charlotte Observerin tietokantatoimittaja Ted Mellnik on sanonut (Cox 2000).

6.3 Versio 2.0?

Koska työni on ylempään ammattikorkeakoulututkintoon sisältyvä kehittämishanke, sen ei tarvitse välttämättä tuottaa lopputuloksenaan julkaisukelpoista palvelua. Koska eduskunta on sekä merkittävä yhteiskunnallinen vallankäyttäjä että kulttuurinen instituutio, siellä käytyjen keskustelujen soisi kuitenkin olevan helpommin kansalaisten saatavilla kuin mikä nykyinen tilanne on. Puutetta on ilmeisesti tulossa osaltaan paikkamaan vuoden 2022 aikana julkaistava semanttisen webin teknologioihin perustuva Parlamenttisampo-palvelu (SeCo 2021), mutta näen, että myös tiedotusvälineiden omalle ”eduskuntakoneelle” voisi ehkä olla paikkansa. Tai sitten samaa lähestymistapaa voisi käyttää muihin vastaaviin aineistokokoelmiin – esimerkiksi kunnanvaltuustojen keskustelupöytäkirjoihin.

Niinpä haluan lopettaa kehittämistyöni käymällä vielä läpi tiiviisti, mitä kehittämistyön myötä kertyneen kokemuksen valossa tekisin nyt toisin, jos – ja kenties kun – lähden toteuttamaan vastaavaa hanketta uudestaan.

Aivan ensimmäisenä tutustuisin todella huolellisesti Salla Simolan julkaisemattomaan työpaperiin (2020), joka ei valitettavasti ollut käytössäni tietokantaa alun perin toteuttaessani. Simola kuvaa työssään erittäin hyvin eduskunnan keskustelupöytäkirjojen muodostaman korpuksen käsittelyn haasteet ja myös ne menetelmät, joilla hän on ongelmat pystynyt ratkaisemaan. Erityisesti ottaisin opiksi Simolan tavasta jäsentää

työprosessi vaiheisiin (ks. tarkemmin alaluku 4.3.1). Jälkikäteen arvioiden pidän tässä vaiheessa tekemiäni virheitä lopputuloksen kannalta kaikkein kriittisimpinä, ja uskon, että niiltä olisi vältytty, jos olisin toiminut samoin kuin Simola omassa työssään. Häneltä kopioisin myös käytetyn tavan puhujien ja puheenvuorojen tunnistamiseen – tämä oli yksi työn haastavimmista (ja itselläni pahimmin epäonnistuneista) vaiheista.

Projekti kannattaa siis ehdottomasti pilkkoa selkeisiin vaiheisiin, joiden tulokset validoidaan ja joita toistetaan iteratiivisesti, kunnes ollaan varmoja siitä, että suurin osa virheistä on saatu eliminoitua ennen kuin siirrytään seuraavan vaiheen työstämiseen. Kehittämistyössäni käyttämäni työskentelymalli, jossa systemaattisempi validointi tapahtui vasta projektin loppupuolella, osoittautui erittäin huonoksi valinnaksi. Myöhäisessä vaiheessa huomattujen virheiden korjaaminen olisi vaatinut, että erittäin suuri osuus työstä olisi jouduttu tekemään uudestaan, mikä ei aikataulusyistä ollut mahdollista. Jos virheet olisi huomattu aikaisemmin, niiden korjaaminen olisi ollut paljon helpompaa.

Validointitapana tiedonkeruu juttua varten toimi sinänsä erittäin hyvin. Toimiva ratkaisu voisikin olla yrittää keksiä juttuideoita, jotka olisivat toteutettavissa eri vaiheissa käytettävissä olevilla keskeneräisillä versioilla aineistosta ja tietokannasta, ja hyödyntää niitä työn validoinnissa.

Alkuvaiheen tekstintunnistus pitäisi tehdä aineistolle kahteen kertaan – sekä suomen- että ruotsinkielisillä asetuksilla. Tulokset pitäisi myös validoida vähintään pistokoeluontoisesti esimerkiksi sanalistojen avulla. Nyt varsinkin ruotsinkielisten tekstien tunnistus on kautta linjan huonolaatuista johtuen siitä, että Tesseract-ohjelmassa on käytetty suomen kielen mukaisia asetuksia. Puheenvuorojen kielet pitäisi tunnistaa nykyistä paremmin ja tunnistuksessa pitäisi myös ottaa huomioon tieto, että aineistossa on kohtuullisen suuri määrä kaksikielisiä puheenvuoroja – kieli voi siis vaihtua puheenvuoron sisälläkin, mitä tietokannassa ei tällä hetkellä ole huomioitu. Tekstintunnistus ja muut laskennallisesti intensiiviset vaiheet kannattaisi teknisesti toteuttaa rinnakkaislaskentana pilvessä, jolloin ne voitaisiin toteuttaa huomattavasti lyhyemmässä ajassa kuin tässä projektissa, jossa prosessointi iMac-pöytäkoneen yksittäisellä prosessoriytimellä kesti viikkokausia.

Teknisellä puolella korvaisin myös ongelmalliseksi osoittautuneen FinnPos-jäsentimen joko Pirisen (2019) vertailussa selvästi vähemmän virheitä tehneellä Turku-neural-parser-pipelinellä (ks. alaluku 4.3.3), tai sitten uudemmalla UralicNLP:llä, jonka valintaa puoltaisi mm. parempi yhteensopivuus Pythonin kanssa. UralicNLP:n käyttöä puoltaisi sekin seikka, että se pystyy jäsentämään suomen lisäksi myös ruotsinkielistä tekstiä.

Harkitsisin myös vakavasti, onko Amazonin AWS välttämättä paras tekninen alusta verkossa pyörivälle tietokannalle. Vaikka AWS on selvästi edullisin vaihtoehto, palvelun pystyttäminen – ennen kaikkea rajapinnan saaminen toimintaan – oli erittäin hankalaa ja muodostui kehittämistyöni kaikkein työläimmäksi vaiheeksi. Toisaalta vaikeudet johtuivat osittain siitä, että yritin hieman ”oikaista” toteutuksessa hyödyntämällä kokemustani front-end-kehittäjänä (joka ymmärtää back-end-puolta eli serverillä tapahtuvia asioita erittäin huonosti) ja koodaamalla käyttäjän selaimen päässä suoritettaviksi asioita, jotka oikeastaan kuuluisivat palvelimella tehtäviksi. Pelkästään tästä kikkailusta luopuminen ja verkkokehityksen parhaiden käytäntöjen hyödyntäminen back-end-toteutuksessa saattaisi helpottaa teknistä toteutusta.

Hakukäyttöliittymää voisi rikastaa pienillä visualisoinneilla, jotka näyttäisivät esimerkiksi histogrammeina hakutulosten sisäisiä jakaumia ja muita kiinnostavia tietoja. Hakukoneesta pitäisi tehdä myös pelkistetty upotusversio, jonka voisi sisällyttää osaksi muualla julkaistavaa juttua.

Hieman suuremmalla ajallisella alkuinvestoinnilla hakukoneen voisi tuotteistaa – oli kyse sitten kaupallisesta tuotteesta, avoimen lähdekoodin ohjelmistosta tai toimituksen sisäiseen käyttöön tehdystä työkalusta. Ajatuksena olisi tällöin puoliautomatisoida ison skannatuista dokumenteista koostuvan tiedostojoukon tallettaminen hakutoiminnolla varustettuun tietokantaan. Tällaiselle työkalulle olisi varmasti tarvetta monissa toimituksissa.

Lähteet

Amazon 2022. Amazon API Gateway developer guide. Luettavissa: https://docs.aws.amazon.com/apigateway/latest/developerguide/ Luettu 9.1.2022.

Amazon Web Services, Inc. 15.4.2022. Sähköposti.

Andrushchenko, M., Sandberg, K., Turunen, R., Marjanen, J., Hatavara, M., Kurunmäki, J., Nummenmaa, T., Hyvärinen, M., Teräs, K., Peltonen, J., & Nummenmaa, J. 2021. Using parsed and annotated corpora to analyze parliamentarians' talk in Finland. Journal of the Association for Information Science and Technology, 2021, s. 1–15.

Apilo, A. 28.9.2018. Digitoitujen vp-asiakirjojen palvelu. Powerpoint-esitysdiat. Luettavissa: https://avoindata.eduskunta.fi/digitoidut/esittely Luettu: 5.1.2022.

Berezhnoi, R. 2019. What is UI design and why is it important? Luettavissa: https://f5-studio.com/articles/what-is-user-interface-design-and-why-is-it-important/ Luettu: 4.9.2021.

Bell, E. 2012. Journalism by numbers. Luettavissa: https://archives.cjr.org/cover_story/journalism_by_numbers.php Luettu: 4.1.2022.

Bradshaw, P. 2012. What is data journalism? Teoksessa Bounegru, L.; Chambers, L. & Gray, J. (toim.) The data journalism handbook 1. European Journalism Centre. Maastricht. Luettavissa: https://datajournalism.com/read/handbook/one Luettu: 2.1.2021.

COSS ry. (Suomen avoimien tietojärjestelmien keskus) s.a. Avoin lähdekoodi. Luettavissa: https://coss.fi/avoimuus/avoin-lahdekoodi/ Luettu: 6.1.2022.

Cox, M. 2000. The development of computer-assisted reporting. A paper presented to the Newspaper Division, Association for Education in Journalism and Mass Communication, Southeast Colloquium. Pohjois-Carolinan yliopisto Chapel Hillissä. Chapel Hill. Luettavissa: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.631.6220&rep=rep1&type=pdf Luettu 18.1.2022.

Economist, The 29.1.2022. What Spotify data show about the decline of English. Luettavissa: https://www.economist.com/interactive/graphic-detail/2022/01/29/what-spotify-data-show-about-the-decline-of-english Luettu 3.2.2022.

Eduskunta 2018a. Avoin data. Luettavissa: https://avoindata.eduskunta.fi Luettu: 4.1.2022.

Eduskunta 2018b. Digitoidut valtiopäiväasiakirjat 1907–2000. Luettavissa: https://avoindata.eduskunta.fi/#/fi/digitoidut/download Luettu: 6.11.2020.

Eduskunta s.a.-a. Haku: Kansanedustajat. Luettavissa: https://www.eduskunta.fi/FI/search/Sivut/peopleresults.aspx?k= Luettu: 6.11.2020.

Eduskunta s.a.-b. Tarkennettu haku. Luettavissa: https://www.eduskunta.fi/FI/search/Sivut/advanced.aspx Luettu: 29.8.2021.

FIN-CLARIN 2015. Kielipankki. Luettavissa: https://www.kielipankki.fi Luettu: 2.4.2022

Forum Virium 7.4.2015. Kansan muisti: kansanedustajien teot näkyviksi. Luettavissa: https://forumvirium.fi/kansan-muisti-kansanedustajien-teot-nakyvaksi/ Luettu: 4.1.2022.

Fouquenet, M. 7.7.2021. Journalism and machine learning: a poweful tool for investigations. Medium. Luettavissa: https://mfouquenet.medium.com/journalism-and-machine-learning-a-poweful-tool-for-investigations-91f12eff181b Luettu: 8.1.2022.

Gawiser, S. R. & Evans Witt, G. 1994. A journalist's guide to public opinion polls. Praeger. Westport & Lontoo.

Google 2013. Word2vec. Luettavissa: https://code.google.com/archive/p/word2vec/ Luettu: 6.1.2022.

Google s.a. Google Books Ngram Viewer. Luettavissa: https://books.google.com/ngrams/graph?content=data+journalism%2Ccomputer-assisted+reporting&year_start=1970&year_end=2019 Luettu 18.1.2022.

Gupta, L. 2021. What is REST? Luettavissa: https://restfulapi.net Luettu: 9.1.2022.

Helsingin yliopisto 2021. Web-palvelinohjelmointi Java, 2021: opi tekemään verkossa toimivia sovelluksia. Verkkokurssi. Helsingin yliopisto. Helsinki. Luettavissa: https://web-palvelinohjelmointi-21.mooc.fi Luettu 9.1.2022.

Helsingin yliopisto 3.3.2021. Uusilla metodeilla voi analysoida tekstin tunnelatauksia entistä paremmin. Lehdistötiedote. Luettavissa: https://www.sttinfo.fi/tiedote/uusilla-metodeilla-voi-analysoida-tekstin-tunnelatauksia-entista-paremmin?publisherId=3747&releaseId=69902466 Luettu: 8.1.2022.

Knight, M. 2015. Data journalism in the UK: a preliminary analysis of form and content. Journal of Media Practice, 16, 1, s. 55–72.

Holmer, D. 19.3.2018. Query your AWS database from your serverless application. AWS Database Blog. Luettavissa: https://aws.amazon.com/blogs/database/query-your-aws-database-from-your-serverless-application/ Luettu: 19.10.2021.

Holovaty, A. 2006. A fundamental way newspaper sites need to change. Luettavissa: http://www.holovaty.com/writing/fundamental-change/ Luettu: 30.8.2021.

Houston, B. 2019. Data for journalists. A practical guide for computer-assisted reporting. Fifth edition. Routledge. Abingdon-on-Thames.

Hristozov, K. 19.7.2019. MySQL vs. PostgreSQL: choose the right database for your project. Okta Developer. Luettavissa: https://developer.okta.com/blog/2019/07/19/mysql-vs-postgres Luettu: 8.1.2022.

HS 21.5.2012. HS perustaa datajournalismin ryhmän. Helsingin Sanomat. Luettavissa: https://www.hs.fi/kotimaa/art-2000002531942.html Luettu: 19.1.2022.

Humak 2021. Opinnäytetyöopas YAMK. Luettavissa: https://humak.libguides.com/c.php?g=688355 Luettu: 3.11.2021

Hyvönen, E., Sinikallio, L., Leskinen, P., Drobac, S., Tuominen, J., Elo, K., La Mela, M., Koho, M., Ikkala, E., Tamper, M., Leal, R. & Kesäniemi J. 2021. Parlamenttisampo: eduskunnan aineistojen linkitetyn avoimen datan palvelu ja sen käyttömahdollisuudet. Informaatiotutkimus, 40, 3, s. 217–244.

Hämäläinen, M. & Alnajjar, K. 2021. Current state of Finnish NLP. https://arxiv.org/pdf/2109.11326.pdf Luettu: 5.1.2022.

Inha O. 2021. Govt.fi. Luettavissa: https://govt.fi Luettu: 3.11.2021

Jaakkola, J. 1997. Tekstihaun toteutustekniikat. Luettavissa: https://www.cs.helsinki.fi/u/jjaakkol/tutkielma/tutkielma.html Luettu: 5.1.2022.

Jalonen, E. 7.11.2021. Vanhempi ohjelmistokehittäjä. Distrikt 13. Haastattelu. Helsinki.

John Snow Labls 2022. Finnish lemmatizer. Luettavissa: https://nlp.johnsnowlabs.com/2020/05/05/lemma_fi.html Luettu: 7.1.2022.

Jokimies, T. 2016. Tietoturvauhkien torjuminen selainpohjaisilla tietoturvamekanismeilla AngularJS-sovelluksissa. Pro gradu -työ. Helsingin yliopisto. Helsinki. Luettavissa: https://helda.helsinki.fi/bitstream/handle/10138/167478/tuomo-jokimies-gradu.pdf Luettu 9.1.2022.

JSN (Julkisen sanan neuvosto) 2011. Journalistin ohjeet ja liite. Luettavissa: https://www.jsn.fi/journalistin_ohjeet/ Luettu: 5.1.2022.

Kananen, J. 2015. Kehittämistutkimuksen kirjoittamisen käytännön opas: miten kirjoitan kehittämistutkimuksen vaihe vaiheelta. Jyväskylän ammattikorkeakoulu. Jyväskylä.

Kansan muisti ry. 2010. Kansan muisti. Luettavissa: https://kansanmuisti.fi/ Luettu: 31.5.2021.

Keefe, J., Zhou, Y. & Merrill, J. B. 12.5.2021. The present and potential of AI in journalism. Knight Foundation. Luettavissa: https://knightfoundation.org/articles/the-present-and-potential-of-ai-in-journalism/ Luettu: 8.1.2022.

Kelleher, J. D. & Tierney, B. 2021. Datatiede. Suom. Kimmo Pietiläinen. Terra Cogita. Helsinki.

Klein, S. 16.3.2016. Infographics in the time of cholera. The ProPublica Nerd Blog. Luettavissa: https://www.propublica.org/nerds/infographics-in-the-time-of-cholera Luettu 18.1.2022.

Koponen, J. 14.8.2018. Informaatiomuotoilu – ihmisen ja tiedon rajapinnassa. Esitys Finnan 5-vuotisjuhlaseminaarissa. Helsinki.

Koponen, J. 2021. Aineiston tuominen ja vieminen. Sähköinen opetusmateriaali. Haaga-Helia. Helsinki. Luettavissa: http://demo.koponen-hilden.fi/journalistin-datankäsittelytaidot-k2021/Aineiston_tuominen_ja_vieminen.html Luettu: 3.5.2021.

Koponen, J., Hildén J. & Vapaasalo, T. 2019. Tieto näkyväksi. Kolmas painos. Aalto ARTS Books. Helsinki.

Koskenniemi, K 2022. FINTWOL. Luettavissa: http://www2.lingsoft.fi/cgi-bin/fintwol Luettu: 7.1.2022.

Kukkamäki, R. & Lehtomäki, E. 2020. Luonnolisen kielen ymmärtäminen koneellisesti. Pro gradu -tutkielma. Jyväskylän yliopisto. Jyväskylä.

Kuutti, H. 2011. Johdatus tietokantajournalismiin. Teoksessa Kuutti, H. (toim.) Julkisuusjournalismi, s. 598–623. Jyväskylän yliopisto. Jyväskylä.

Laaksonen, A. 2020. Tietokantojen perusteet, kevät 2020: johdatus suurten tietomäärien käsittelyyn. Verkkokurssi. Helsingin yliopisto. Helsinki. Luettavissa: https://tikape-k20.mooc.fi Luettu: 4.1.2022.

Lind, S. 2019. Lemmy. Luettavissa: https://github.com/sorenlind/lemmy Luettu: 27.11.2021.

Loosen, W. 2002. The second-level digital divide of the web and its impact on journalism. First Monday, 7, 8.

magichat 2021. Maximum length of a URL in different browsers. GeeksforGeeks. Luettavissa: https://www.geeksforgeeks.org/maximum-length-of-a-url-in-different-browsers/ Luettu: 9.1.2022.

Makkonen, K. & Loukasmäki P. 2019. Eduskunnan täysistunnon puheenaiheet 1999–2014. Miten käsitellä LDA-aihemalleja? Politiikka, 61, 2, s. 127–159.

Mervola, P. 1995. Kirja, kirjavampi, sanomalehti. Ulkoasukierre ja suomalaisten sanomalehtien ulkoasu 1771–1994. Väitöskirja. Suomen Historiallinen Seura & Jyväskylän yliopisto. Helsinki ym.

Mumbi, W. 2021. What is Word2Vec? Luettavissa: https://www.section.io/engineering-education/what-is-word2vec/ Luettu: 8.1.2022.

Mäkinen, E. 2.3.2011. HS avaa vaalikoneensa tiedot. Helsingin Sanomat. Luettavissa: https://www.hs.fi/kulttuuri/art-2000004791160.html Luettu: 19.1.2022.

Mäkinen, E. 28.6.2012. Kunnanluoja kehitettiin HS Openissa. Helsingin Sanomat. Luettavissa: https://www.hs.fi/kaupunki/art-2000002540786.html Luettu 19.1.2022.

Mäkinen, E. 16.11.2021. Toimituspäällikkö. Esitys Datajournalismi Suomessa 10 vuotta -seminaarissa. Helsingin Sanomat. Seminaariesitys. Helsinki.

Pirinen, T. A. 2019. Neural and rule-based Finnish NLP models: expectations, experiments and experiences. Proceedings of the fifth Workshop on Computational Linguistics for Uralic Languages, 104–144.

Poikola, A., Kivekäs, O., Kettunen, J., Polo, T., Laine S., Aaltonen J., Moilanen J., Korhonen J., Honkanen M., Kekäläinen, O. & von Willebrand, M. 2014. Avoimen rajapinnan määritelmä. Luettavissa: http://avoinrajapinta.fi Luettu: 6.1.2022.

Postgres.app s.a. Postgres.app: the easiest way to get started with PostgreSQL on the Mac. Luettavissa: https://postgresapp.com Luettu: 15.8.2020.

Puttonen, K., Erkkilä, P., Pakarinen, M., Peuhkurinen, T. & Voutilainen, H. 2012. Suomen valtiopäiväasiakirjat. Käytön opas. 2. uudistettu painos. Eduskunta. Helsinki.

Rantanen, T. 17.3.2015. Kansan muisti ei nuku. Voima. Luettavissa: https://voima.fi/artikkeli/2015/kansan-muisti-ei-nuku-2/ Luettu: 4.1.2022.

Řehůřek, R. 2014. Word2Vec tutorial. Luettavissa: https://rare-technologies.com/word2vec-tutorial/ Luettu: 4.11.2020.

Sanastokeskus TSK s.a. TEPA-termipankki. Luettavissa: https://termipankki.fi/tepa/ Luettu: 3.1.2022.

Sanoma Media Finland 27.10.2015. Helsingin Sanomille kolme uutta osastoa: uutisdeski, datadeski ja HSTV. Lehdistötiedote. Luettavissa: https://www.sanoma.fi/news/2015/news-imported-from-wp/helsingin-sanomille-kolme-uutta-osastoa-uutisdeski-datadeski-ja-hstv2/ Luettu: 3.1.2022.

SeCo (Semantic Computing Research Group) 2021. Parlamenttisampo. Eduskunta semanttisessa webissä. Luettavissa: https://seco.cs.aalto.fi/projects/semparl/ Luettu: 5.11.2021.

Shiab, N. 2015. On the ethics of web scraping and data journalism. Luettavissa: https://gijn.org/2015/08/12/on-the-ethics-of-web-scraping-and-data-journalism/ Luettu: 20.5.2020

Shneiderman, B. & Plaisant, C. 2004. Desigining the user interface: strategies for effective human–computer interaction. Fourth edition. Addison-Wesley. Boston ym.

Simola, S. 2020. Century of partisanship in Finnish political speech. Työpaperi. Luettavissa: https://drive.google.com/file/d/1ZhUiW1c-eKg5Bi3_nNJ7FICiw2fiNAQU/view Luettu: 4.1.2022.

Stray, J. 2010. How The Guardian is pioneering data journalism with free tools. Luettavissa: https://www.niemanlab.org/2010/08/how-the-guardian-is-pioneering-data-journalism-with-free-tools/ Luettu: 3.1.2022.

Stopwords ISO 2016. Stopwords ISO. Luettavissa: https://github.com/stopwords-iso/stopwords-iso Luettu: 1.6.2020.

STT s.a. STT Pikkulintu seuraa verkkosivuja puolestasi. Luettavissa: https://stt.fi/product/pikkulintu/ Luettu: 8.1.2022.

Suominen, P. 2010. Suomenkielisten tekstien morfologinen analyysi. Helsingin yliopisto. Helsinki. Luettavissa: https://www.cs.helsinki.fi/u/leino/opetus/aineisto-s10/esitykset/ps.pdf Luettu: 4.1.2022.

Tero, T. 2016. Amazon AWS -pilvipalveluiden integrointi olemassa olevaan sovellukseen. Insinöörityö. Metropolia-ammattikorkeakoulu. Helsinki. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/121846/Tero_Toomas.pdf?sequence=1 Luettu: 9.1.2022.

Tesseract OCR 2015. Tesseract OCR. Luettavissa: https://github.com/tesseract-ocr/tesseract Luettu: 5.1.2022.

Tieteen termipankki 4.1.2022. Kielitiede: korpus. Luettavissa: https://tieteentermipankki.fi/wiki/Kielitiede:korpus Luettu: 4.1.2022.

Tilastokeskus 2020. Iterointi. Luettavissa: https://www.stat.fi/meta/kas/iterointi.html Luettu: 3.11.2021.

Toikkanen, I. 2014. Pinnallisista klikkikartoista yhteiskunnalliseen merkittävyyteen. Datajournalismin työprosessi Helsingin Sanomissa ja Ylessä. Pro gradu -tutkielma. Jyväskylän yliopisto. Jyväskylä.

Tran, T. 2021. MAA4: analyyttinen geometria ja vektorit. Luettavissa: https://matikka.omaantahtiin.com/pitkamatikka/maa4 Luettu: 8.1.2022.

Tuomi, S. 2021. Opinnäytetyön ohjaajan käsikirja. Luettavissa: https://oppimateriaalit.jamk.fi/yamk-kasikirja/ Luettu: 3.1.2022.

TurkuNLP (Turku neural parser pipeline) 2021. Turku neural parser pipeline. Luettavissa: https://turkunlp.org/Turku-neural-parser-pipeline/ Luettu: 4.1.2022.

Uskali, T. & Kuutti, H. 2016. Datajournalismin työkäytännöt. Vastapaino. Helsinki.

Warde, B. 2005. Kristallimalja, eli typografian tulee olla näkymätöntä. Teoksessa Hinkka, J. (toim.) Beatrice Warde – Kristallimalja, s. 21–32. Grafia, Helsinki.

Ylärinne, T. 2013. Helsinki Region Infoshare: avoimen datan julkaisutekniikat. Koulutusmateriaali. Sovelto Oyj. Helsinki. Luettavissa: https://www.hel.fi/hel2/tietokeskus/data/dokumentit/koulutusmateriaali/Avoimen_datan_julkaisutekniikat.pdf Luettu: 9.1.2022.

Yrjölä, J. 4.1.2021. Perustaja. Kansan muisti ry. Haastattelu pikaviestiohjelman välityksellä. Helsinki.

Liitteet

Liite 1. Pöytäkirjoissa esiintyvät kansanedustajien nimien kirjoitusasut, joita ei pystytä suoran yhdistämään biografiatietoihin

Muoto pöytäkirjassa	Nimi biografiatiedoissa
A. Aalto	Artturi Aalto
A. H. Virkkunen	Artturi Virkkunen
A. K. Wirtanen	Atos Wirtanen
A. L. Jokinen	Anna-Liisa Jokinen
Aalle	Ida Aalle-Teljo
Ala-Kapee	Pirjo Ala-Kapee-Hakulinen
Alfthan (ennen v. 1910)	Kristian von Alfthan
Alfthan (v. 1910 jälkeen)	Alex Alfthan
Arffman	Kusti Arhama
Borgman-Sundman	Margit Borg-Sundman
E. A. Saari	Eero Saari
E. S. Yrjö-Koskinen	Sakari Yrjö-Koskinen
E. Setälä	Edvard Setälä
E. W. Lehtinen	Emil Lehtinen
F. J. Leino	Jussi Leino
Forselles (ennen v. 1919)	Jenny af Forselles
Forselles (v. 1919 jälkeen)	Arthur af Forselles
Forstadius	Jalmari Pilkama
Hahl	Eero Hatva
Hänninen (ennen v. 1919)	Edvard Valpas
Hirvelä	Inger Hirvelä López
Hurskainen-Leppänen	Sinikka Hurskainen
J. Annala	Jussi Annala
J. E. Partanen	Eemil Partanen
J. F. Aalto	Johan Aalto
J. Laine	Jermu Laine
J. Leino	Jalmari Leino
J. V. Vainio	Vihtori Vainio
Järvisalo-Kanerva	Riitta Järvisalo
K. E. Linna	Eemil Linna
Karttunen-Raiskio	Marjukka Karttunen

Kreivi Berg	Erik Berg
Kuusinen-Leino	Hertta Kuusinen
Kuuskoski-Vikatmaa	Eeva Kuuskoski
L. Linna	Lauri Linna
L. Myllymäki	Lauri Myllymäki
Laine, Maria	Maria Paaso
Laine, O.	Oskar Laine
Laine, O. F.	Oskari Laine
Leino (ennen v. 1939)	Oskari Leino
Löthman	Tilda Löthman-Koponen
Numminen-Härmä	Laura Härmä
O. Korhonen	Ville Korhonen
O. Laine	Oskari Laine
Perho-Santala	Maija Perho
Serlachius	Allan Särkilahti
Sventorzetski	Reinhold Svento
T. M. Koivisto	Tellervo Maria Koivisto
T. T. Koivisto	Tellervo Koivisto
Tavastähti	Elli Hiidenheimo
Tichanoff	Vasili Suosaari
Tiekso	Anna-Liisa Korpinen
Turpeinen-Kajanoja	Pirkko Turpeinen-Saari
Ursin	Nils Robert af Ursin
V. A. Virtanen	Viljo Virtanen
V. I. Rytkönen	Veikko Ilmari Rytkönen
V. J. Rytkönen	Veikko J. Rytkönen
V. K. Turunen	Varma Turunen
V. Korhonen	Vilho Korhonen
V. R. Virtanen	Viljo Virtanen
V. Vainio	Ville Vainio
Valpas-Hänninen	Edvard Valpas
Vikatmaa	Eeva Kuuskoski
Viljomaa	Marja-Leena Viljamaa

Liite 2. Ohje FinnPOS-jäsentimen asentamiseen macOS-käyttöjärjestelmään

Ohjeistus on testattu macOS Mojave -käyttöjärjestelmällä.

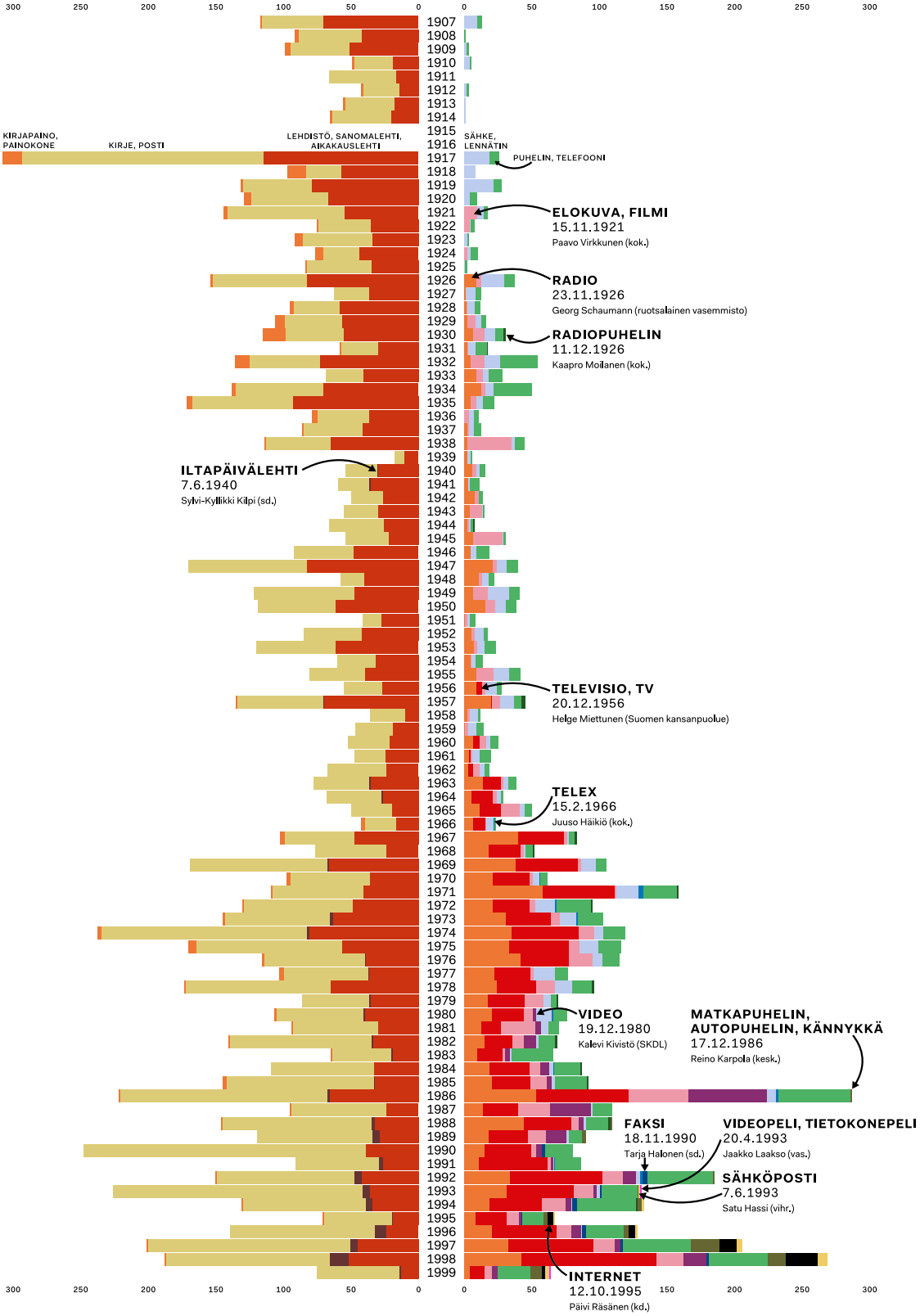
Tässä ohjeistuksessa oletetaan, että haet FinnPosin asennuspaketin omaan latauskansioosi (**Downloads**). Mikäli haluat asentaa sen toisesta kansiosta käsin, hakemiston polku `~/Downloads` on alla komennoissa 2 ja 10 korvattava sen hakemiston polulla, johon haluat FinnPosin asennuspaketin ladata. Asennukseen tarvitaan pääkäyttäjän oikeudet.

1. Avaa **Terminal**-ohjelma
2. Siirry latauskansioon kirjoittamalla `cd ~/Downloads`
   (FinnPosin asennuspaketti ladataan tähän kansioon)
3. Lataa FinnPosin asennuspaketti komennolla
   `git clone https://github.com/mpsilfve/FinnPos`

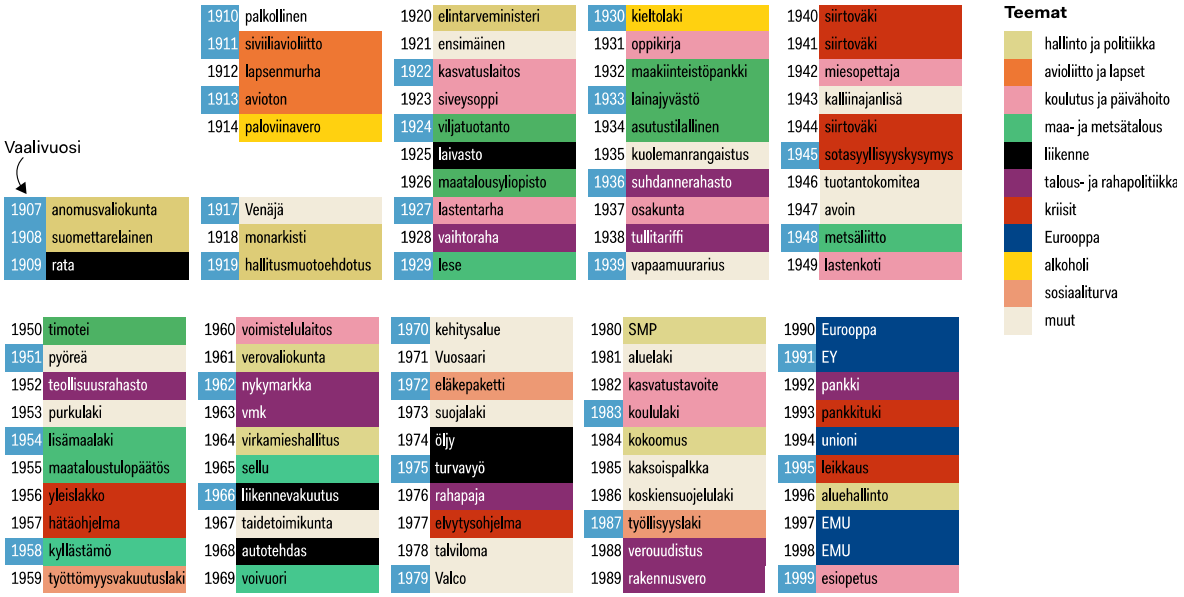
   Jos FinnPosin asentamiseen käytettävä **git**-ohjelma ei ole valmiiksi asennettuna koneellesi, tämän komennon suorittaminen käynnistää automaattisesti asennusohjelman ("*Welcome to the Git 2.20.1 Installer*" jne.). Tällaisessa tapauksessa suorita ensin asennusohjelma loppuun ja syötä sen jälkeen kohdassa 3 mainittu komento uudestaan **Terminal**-ohjelman komentoriville
4. Avaa selain, mene osoitteeseen https://github.com/mpsilfve/FinnPos/releases ja lataa **Assets**-otsikon alta gzip-paketti **morphology.omor.hfst.gz**
5. Avaa **Finder** ja mene latauskansioosi. Riippuen käyttöjärjestelmän asetuksista lataamasi gzip-paketti on saattanut heti latauduttuaan purkautua automaattisesti. Mikäli näin ei ole käynyt, pura paketti tuplaklikkaamalla sitä.
6. Siirrä Finderissä paketista purettu **morphology.omor.hfst**-tiedosto FinnPosin asennuspaketin alakansioon **share/finnpos/omorfi/**
   Tämän voi tehdä myös Terminal-ohjelmassa komennolla
   `mv morphology.omor.hfst FinnPos/share/finnpos/omorfi/`
7. Avaa koodieditorissa (esim. **Sublime Text** tai vaikka macOS:n mukana tuleva **TextEdit**) FinnPosin asennuspaketin alakansiosta **src/tagger/** löytyvä tiedosto **UnorderedMapSet.hh**
8. Muokkaa **UnorderedMapSet.hh**-tiedoston rivejä 5 ja 6 poistamalla niistä kummastakin merkkijono `tr1/`
   Rivi `#include <tr1/unordered_map>` muuttuu siis muotoon
   `#include <unordered_map>` jne.
9. Muokkaa saman tiedoston rivejä 12 ja 13 vastaavasti poistamalla niistä kummastakin merkkijono `tr1::`
   Rivi `using std::tr1::unordered_map;` muuttuu muotoon
   `using std::unordered_map;` jne.

10. Avaa **Terminal**-ohjelma ja siirry komennolla `cd ~/Downloads/Finnpos` FinnPosin asennuspaketin pääkansioon
11. Suorita komento `make`
12. Suorita komento `make ftb-omorfi-tagger` (Kesto 5–10 min.)
13. Suorita komento `sudo make install` (tässä kohdin joudut syöttämään salasanasi)
14. Suorita komento `sudo make install-models` (Kesto 5–10 min.)
15. Kokeile suorittaa komento `echo "junassa" | ftb-label`
    Jos saat virheilmoituksen `-bash: ftb-label: command not found`, suorita vielä komento `export PATH=$PATH:/usr/local/bin`
16. Asennus on nyt valmis. Lopuksi voit vielä avata **Finderin** ja poistaa latauskansiosta kohdassa 2 ladaatun FinnPosin asennuspaketin sekä kohdassa 4 ladatun gzip-paketin

Liite 3. Visualisointi viestintäteknologioiden maininnoista eduskunnan täysistuntojen suomenkielisissä keskusteluissa



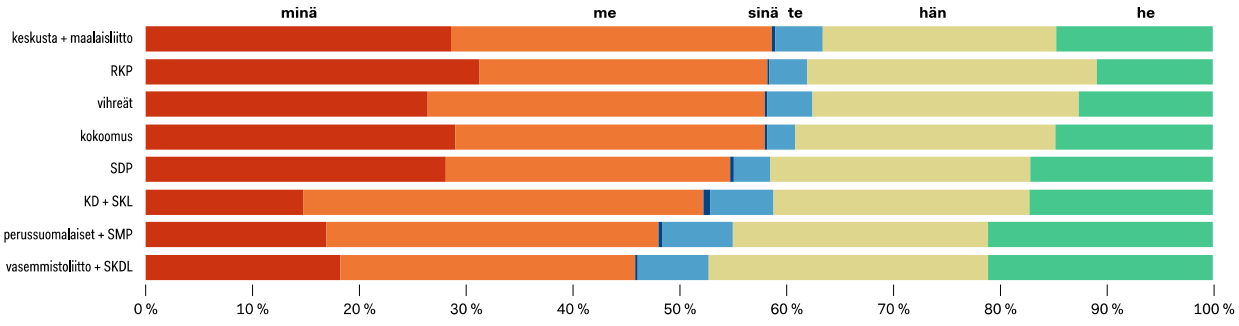
Liite 4. Visualisointeja eduskunnan keskustelun teemoista



Kunkin vuoden suhteellisesti eniten eduskunnan keskusteluissa käytetty sana

	SDP	kokoomus	keskusta maalaisliitto	RKP	vasemmistoliitto SKDL	perussuomalaiset SMP	vihreät	KD SKL
1	maa	opetusministeri	vaalikausi	Vaasa	suurpääoma	EDP	ydinvoima	Raamattu
2	porvarillinen	haltija	vasten	pohjoismainen	metsätyöläinen	veteraani	prosessi	Alko
3	valtio	vapaaehtoinen	viljelys	kiittää	liitetty	kuluttajahinta	ympäristöministeriö	lähetysjärjestö
4	työväki	valtio	maatalousväestö	läsnä	rakennustyö	kenttä	maapallo	tupakointi
5	markka	poikkeuksellinen	kulkulaitosvaliokunta	peräänkuuluttaa	henkilöstö	pienviljelmä	ympäristönsuojelu	valitsijamies
6	työ	elinkeinoelämä	metsätalousvaliokunta	meri	toimeentulotuki	hämmästyä	ihmisoikeus	avioliitto
7	huomio	puolustusvoimat	jatkossa	SKP	palkankorotus	päättynyt	eläin	työvoimakustannus
8	mieli	maa	metsänomistaja	kehitysapu	työttömyysturva	öljy	ympäristövaikutus	alkoholijuoma
9	kunta	markka	merkittävästi	koskien	peruspäiväraha	peli	ydinvoimala	sukunimi
10	tärkeä	professori	alueellinen	pakolainen	valtionyhtiö	aikaansaaminen	ympäristövero	puoliso

Kunkin puolueen 10 suhteellisesti eniten käytettyä sanaa



Käytettyjen persoonapronominien jakauma puolueittain