Tyhjien tunnistaminen automaattisesti – miksi se on niin vaikeaa?

Anssi Jääskeläinen
Tutkimuspäällikkö
XAMK
Digitalia

Reko Etelävuori

Kansallisarkisto

Tuomo Räisänen
IT-asiantuntija
XAMK
Digitalia

Mika Koistinen

Kansallisarkisto

Tässä artikkelissa kerromme kansantajuisesti Digitalian ja Kansallisarkiston yhteistyönä toteuttamasta pilotoinnista, jossa sekä keinoälyn (AI) että sääntöihin perustuvan päättelyn avulla pyrittiin erottelemaan skannatuista asiakirjoista (bittikarttakuvista) tyhjät sivut mahdollisimman luotettavasti. Lisäksi tutkittiin mahdollisuutta erotella toisistaan konekirjoitetut sivut, käsin kirjoitetut sivut ja näitä molempia sisältävät sivut. Kyseessä oli siis aineiston luokittelutehtävä. Tulokset ovat lupaavia ja parantavat merkittävästi aineistojen käytettävyyttä.

Taustat

Kansallisarkiston tehtävänä on varmistaa kansalliseen kulttuuriperintöön kuuluvien asiakirjojen säilyminen ja saatavuus sekä edistää niiden tutkimuskäyttöä. Tämän pilottiprojektin aikana pyrittiin löytämään uusia keinoja edistää ja helpottaa tutkimuskäyttöä vaarantamatta kuitenkaan aineistojen säilymistä. Kuvatut toimenpiteet kohdistuvat siis käyttökappaleisiin tai niistä tehtyihin käsittelykopioihin.

Seuraavaksi on todettava, että yhdelläkään automaattisella menetelmällä ei tulla koskaan pääsemään 100 % tarkkuuteen. Kuitenkin sekä Digitalian että Kansallisarkiston tulokset osoittavat, että varsin hyvään 97–99 % tarkkuuteen tyhjien tunnistuksessa voidaan päästä monellakin eri menetelmällä.

Kansallisarkistossa on tehty digitointia vuodesta 1999. Digitointiprosessia on kehitetty sen koko olemassaolon ajan sekä virkatyönä että erillisissä projekteissa. Nykyisellään Kansallisarkiston digitointitoiminta jakautuu kahteen toimintoon, takautuvaan digitointiin ja massadigitointiin. Takautuva digitointi tarkoittaa Kansallisarkistoon analogisessa muodossa vuosien saatossa siirrettyjen aineistojen digitaaliseksi muotoon muuttamista, kun taas massadigitointi tarkoittaa edelleen aineiston luoneen viranomaisen hallussa olevan analogisen aineiston digitointia analogisen aineiston siirron yhteydessä. Digitoinnissa tärkeintä on tallentaa analogisista aineistoista tieto täydellisenä digitaaliseen muotoon – tässä vaiheessa ei ole tarkoitus tehdä sen suurempia tulkintoja siitä, onko jokin tieto digitaalisessa kuvassa merkittävää vaiko ei.

Pilottiprojektissamme Digitalian henkilökuntaa ei tarkastettu Suojelupoliisin toimesta, mikä olisi vaadittu massadigitoitavan aineiston

käsittelyyn. Keskityimme sen sijaan takautuvan digitoinnin vanhempaan aineistoon, joka oli annettavissa tutkimuskäyttöön kevyemmällä hallinnolla. Saimme käyttöömme runsaat 10 000 sivua sangen monimuotoista materiaalia: pöytäkirjoja suomen ja ruotsin kielellä, erilaisia lomakkeita sekä allekirjoituksilla tai merkinnöillä varustettuja sivuja kokonaan tai osittain käsin kirjoitettuna. Mukana oli myös kopioita kokonaisista aukeamista ja tietenkin myös tyhjiä sivuja. Osa aineistosta oli ruutupaperille kirjoitettu ja osassa oli repeytymiä.

Ongelma

Ihmiselle luokitteluongelmamme on sangen helppo ja silmäilemällä ratkottavissa. Kansallisarkiston tapauksessa materiaalin määrä lasketaan kuitenkin hyllykilometreissä, joten tarvittavien resurssien määrän voi lukija itse kuvitella. Jokaisen sivun tallentaminen tallekappaleena sekä käyttökappaleina ja metatietoina vie tilaa. Vuodessa kumuloituvan aineiston kohdalla voidaan puhua kymmenien, jopa satojen teratavujen määristä. Tyhjiä ei kuitenkaan voida Kansallisarkiston tapauksessa hävittää, ellei tyhjän määritelmä ole selkeä ja tunnistus varma.

Optimaalisessa tilanteessa tyhjä sivu sisältää pelkkiä samanvärisiä pikseleitä. Entäpä valkoisen eri sävyt, kohinaa sisältävät sivut, tyhjän taulukon sisältävä sivu ja niin edelleen? Kuvassa 1 on joitakin esimerkkejä tyhjistä sivuista:

Ihmissilmälle sivut ovat luonnollisesti tyhjiä, koska niissä ei ole asiasisältöä, mutta koneelle tilanne ei ole helppo. Jos kuvia parannetaan, taustapuoli ja viivat tehostuvat, jolloin niistä löytyy ”tekstiä”. Jos taas kuvia ei tehosteta, haaleimmista oikeasti tietoa sisältävistä sivuista ei tunnisteta mitään. Jos tyhjän tunnistus ei ole varma, väärä poisto olisi riski tiedon eheyden suhteen.

Älä tule paha (kriittinen) virhe, tule hyvä virhe (tai mielummin ei sitäkään)

Pilotissa pyrimmekin tunnistamaan tyhjät ja merkitsemään ne, jotta ne voitaisiin piilottaa käyttöliittymässä. Näin voidaan helpottaa käyttäjien tiedon etsintää ja parantaa käyttäjäkokemusta. Piilotetut sivut olisivat kuitenkin tarvittaessa nähtävissä myös käyttöliittymässä siltä varalta, jos tunnistus on tehnyt virheen.

Kokeiluissa havaittiin, että tyhjien sivujen tunnistus on kaikesta huolimatta “helpointa”. Niinpä päädyimme karsimaan aineistoa siten, että ensi vaiheessa erotellaan tyhjät ja muu aineisto menee sen jälkeen jatkokäsittelyyn. Jotta tämä voitaisiin tehdä automaattisesti, kriittisiä virheitä ei saisi tulla. Kriittisellä virheellä tarkoitetaan tässä yhteydessä sitä, että informaatiota sisältävä sivu tulkitaankin tyhjäksi.

Tekniikkaa

Peruslähtökohtanamme Digitalian kaikkeen tekemiseen on seuraava motto: ”Jos jotain pitää tehdä useammin kuin kerran, se pitää automatisoida”. Näin laajan casen kanssa muuta vaihtoehtoa ei olekaan. Ihmistyönä jo 10 000 sivun lajitteleminen kestäisi helposti päiviä. Automatiikka hoitaa asian tunneissa ja minuuteissa, joskin hieman enemmän virheitä tehden. Automatiikan eduksi on laskettava se, että se toimii joka kerta täsmälleen samalla tavalla ja samalla tehokkuudella.

Kuva 1. Esimerkkejä pilotissa kohdatuista tyhjistä sivuista.

Toisena etuna automaattisella käsittelyllä on sen skaalautuvuus: lisäämällä prosessointitehoa saadaan käsittelyaikoja lyhennettyä tarkkuuden kärsimättä. Ihminen puolestaan toimii tyhjien suhteen hieman eri tavalla etenkin silloin, kun tyhjää sivua ei ole tarkasti määritelty. Näin ollen ihmisresurssien lisääminen johtaisi suurempaan vaihteluun laadussa.

Sääntöpohjaisuus

Ensimmäinen osa Digitalian kehittämästä ratkaisusta pohjautuu Python-ohjelmointikieleen, OpenCV -kirjastoon[1] sekä Tesseract OCR -ohjelmaan[2]. Työnkulku on lyhyesti kuvattuna seuraava. Jokainen kuva käsitellään erikseen ja rinnakkaisia käsittelyjä on yhtä monta kuin käytössä olevia prosessoriytimiä. Ensimmäisenä kuvasta rajataan jokaisesta reunasta 45 pikseliä, jonka jälkeen kuvalle tehdään kohinan poisto ja adaptiivinen mustavalkoiseksi muuttaminen (binärisöinti). Kuva 2 näyttää samaisen työnkulun kuvina.

Kuvanparannuksien ja mustavalkoiseksi muuntamisen jälkeen kuva syötetään ensimmäisen kerran Tesseractille, joka tekee kuvalle nopean OSD (Orientation and Script Detection) -tunnistuksen. Jos tämä tunnistus löytää kuvasta jotakin, kuva tulkitaan tietoa sisältäväksi. Jos OSD ei saa tulosta, samainen sivu syötetään uudelleen Tesseractille, joka tekee tällä kertaa syvemmän analysoinnin sivulle. Tämä tunnistus palauttaa lähes sivusta kuin sivusta jotakin, joten Tesseractin tuottamia numeerisia tuloksia on analysoitava lisää ohjelmallisesti, tai lähes kaikki tyhjät sivut tunnistuvat tietoa sisältäviksi.

Tesseractin tuottamista tuloksista hyödynnetään tässä työnkulussa vain tekstiksi tunnistetut osat ja tunnistuksen luotettavuus. Näihin sovelletaan sekä määriin, keskiarvoihin että yksittäisiin tietokenttiin perustuvia jaottelumenetelmiä. Lopputuloksena lähes kaikki oikeasti tyhjät sivut saadaan eroteltua joukosta, mutta virheitäkin luonnollisesti tulee. Toteutettujen kuvanparannustoimenpiteiden lisäksi kokeiltiin myös viivojen poistoa ja sivujen suoristamista, mutta kumpikaan ei parantanut tunnistustarkkuutta, viivojen poisto pikemminkin heikensi sitä. Lisäksi pilotin aikana kokeiltiin kuvan syöttämistä Tesseractille jokaisen välivaiheen jälkeen ja eri järjestyksissä. Kokeiluissa parhaaksi osoittautui valitsemamme työnkulku, jossa ensin poistetaan reunukset, sitten tehdään kohinan poisto ja viimeisenä binärisöinti.

Käytettäessä virtuaalipalvelinta ja 32 virtuaali-CPU:ta ajoaika per sivu on noin 0,7s. Jos CPU-määrä pudotetaan puoleen, ajoaika on noin 1,2s per sivu. Seuraavaksi työlistallamme onkin tutkia, voiko prosessia tehostaa hyödyntämällä GPU:n CUDA-ytimiä suorituksessa. Tähän liittyen palvelimeemme on juuri asennettu Nvidia Quadro M4000, jossa on 1664 CUDA-ydintä. Lisäksi hankintalistalla on RTX 3090 -korttiin perustuva itsenäinen GPU-laskentatietokone, joka nostaisi CUDA-ytimiemme määrän yli kymmeneen tuhanteen.

1 https://opencv.org/

2 https://github.com/tesseract-ocr/tesseract

Kuva 2. Digitalian sääntöpohjainen työnkulku kuvina.

Sääntöpohjaisuuden ongelmakohdat

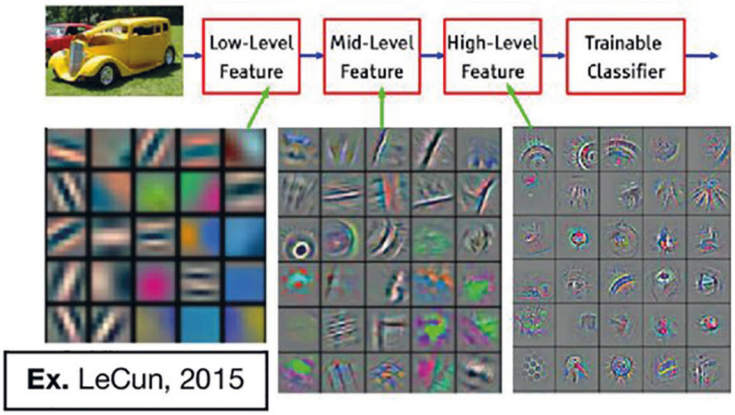
Erityisen ongelmallisia sivuja ovat sellaiset, joissa on ainoastaan hieman käsinkirjoitettua materiaalia, etenkin jos käsiala on epäselvää. Toinen selkeä ongelmakohta Tesseractille ja sääntöpohjaiselle päättelylle ovat sivut, jossa tekstiä on jonkinlaisen värillisen laatikon sisällä. Molemmissa tapauksissa Tesseract saattaa tulkita sisällön niin väärin, että sääntömme, joille Tesseractin tulokset syötetään, tulkitsevat sivun tyhjäksi. Sääntöjä muuttamalla tätä voidaan luonnollisesti korjata, mutta sääntöjen korjaaminen toisesta päästä aiheuttaa virheitä toiseen päähän, jolloin enemmän tyhjiä sivuja tunnistuu tietoa sisältäviksi.

Digitalian keinoälyratkaisu

Digitalian ratkaisun toinen osa pohjautuu keinoälyyn. Tesseract toimii itsekin valmiiksi opetetun LSTM (Long Short Term Memory) -neuroverkon päällä, joten periaatteessa jo ensimmäinenkin osa oli tekoälyratkaisu.

Aluksi Scikit Learniin[3] perustuvaa keinoälyratkaisuamme ajettiin rinnakkain sääntöpohjaisen päättelyn kanssa. Tällöin saavutettiin tyhjien osalta parhaimmillaan noin 99 % tarkkuus kokonaistarkkuuden ollessa noin 80 %. Kriittisiä virheitä tuli kuitenkin enemmän kuin sääntöpohjaisella päättelyllä, joten laitoimme keinoälyratkaisun tutkimaan paremmin tyhjien tunnistuksesta selviytyneen sääntöpohjaisen päättelyn tuottamaa muuksi materiaaliksi tunnistettua aineistoa.

Keinoälyratkaisumme tutkii siis Tesseractin antamaa numerotietoa, joka sisältää mm. jokaiselle tunnistetulle sanalle annetun luotettavuusarvon (confidence rate). Se ilmaisee todennäköisyyden, jolla sana on tunnistettu. Karkeasti pelkistäen: iso prosentti indikoi konekirjoitettua tekstiä, mutta todellisuudessa sivukohtaista numerodataa päättelyyn tarvittiin paljon enemmän. Lisäsimme malliimme erilaisia tilastollisia arvoja, joita pyöriteltiin yrityksen ja erehdyksen kanssa Scikit learn -paketista löytyvien luokittimien avulla. Luokittimien vaihdoilla saatiin aikaan eroja tuloksissa, mutta lopputulos jäi kuitenkin siihen, että kokonaistarkkuus pyöri 80%:n molemmin puolin. Tässäkin tapauksessa neuroverkkopohjainen ratkaisu vaikuttaisi toimivan paremmin, joskin on todettava, että Digitalian ja Kansallisarkiston käyttämät opetusmateriaalit erosivat toisistaan eivätkä keinoälyratkaisut siksi ole keskenään vertailukelpoisia.



Kuva3. Yleinen esimerkki neuroverkkojen piirteiden oppimisesta eri kerroksilla, Yann LeCun, 2015.

Kansallisarkiston keinoälyratkaisu

Kansallisarkistolla kokeiltiin kouluttaa tyhjien tunnistamisen malli esikoulutetulla konvoluutioneuroverkolla. Emme pitkästytä Failin lukijoita neuroverkkoteorioilla, mutta asiasta kiinnostuneet voivat lukea lisää suomeksi esim. http://urn.fi/URN:NBN:fi-fe2020120198991 tai tieteellisemmin https://arxiv.org/pdf/1512.03385.pdf.

Mallina tässä kokeilussa käytettiin 18-kerroksista ResNet18-verkkoa siirretyllä oppimisella. Piirteinä käytettiin 224x224 kokoon pienennettyä kuvaa. Mallin luomisessa käytimme PyTorch-kirjaston[4] päälle rakennettua Fastai-kirjastoa[5]. Opetusaineistoa oli yhteensä noin 25000 kuvaa, joista ResNet18 säätää kohdilleen noin 11 miljoona parametria opetusvaiheessa. Kun aineistot on kerätty haluttuihin luokkiin, kuvatun mallin opetus kestää noin 3-5 tuntia tehokkaalla GPU-koneella.

Painotimme muita kuvia koulutuksessa, koska ne eivät saisi tunnistua tyhjiksi. Näin saimme vähennettyä kriittisiä virheitä. Painotuksesta seurasi kuitenkin tyhjien tunnistamistuloksen heikkeneminen. Tämä tapahtuu luultavasti siksi, että luokitin valitsee epäselvissä tapauksissa yleensä painotetumman luokan.

3 https://scikit-learn.org/

4 https://pytorch.org/

5 https://www.fast.ai/

Digitalian sääntöpohjainen malli				
Oikea luokka	**N**	**Muu-luokkaan tunnistuneet**	**Tyhjä-luokkaan tunnistuneet**	**Tarkkuus**
muu	17359	17169	190	98,905 %
tyhjä	11928	287	11641	97,594 %

Kansallisarkiston ResNet18 malli				
Oikea luokka	**N**	**Muu-luokkaan tunnistuneet**	**Tyhjä-luokkaan tunnistuneet**	**Tarkkuus**
muu	17359	17338	21	99,879 %
tyhjä	11928	1076	10852	90,979 %

Taulukko 1. Digitalian sääntöpohjainen ja Kansallisarkiston AI-ratkaisu numeroina.

Kuten sääntöpohjaisessakin menetelmässä, erityisen ongelmallisia sivuja ovat sellaiset, joissa on ainoastaan hieman käsinkirjoitettua materiaalia, etenkin epäselvällä käsialalla.

Tulokset

Ajoimme sääntöpohjaisen mallin sekä ResNet18-mallin läpi samoilla testiaineistoilla ja tulokset esitetään taulukossa 1. Muu-luokkaan kuuluvia tiedostoja testissä oli 17359 ja tyhjä-luokkaan kuuluvia tiedostoja 11928. Taulukossa oikein menneet on esitetty vihreällä ja väärin menneet valkoisella pohjalla. Paras kokonaistarkkuus on korostettu vaaleansinisellä värillä.

Yhteenveto

Pilotin tuloksien valossa opetettu ResNet18-neuroverkkopohjainen ratkaisu teki vähiten kriittisiä virheitä. Sääntöpohjainen malli olisi parempi, jos kokonaistarkkuus olisi määrittelevä tekijä. Neuroverkkopohjaista mallia voidaan myös säätää luokittimen antamien confidence-arvojen avulla tarvittaessa, mutta silloin ”hyvien” virheiden osuus kasvaa entisestään.

Muu-luokan virheet sisälsivät pääasiassa kuvia, jotka sisälsivät muutamia sanoja epäselvää käsinkirjoitusta. Samankaltainen kuvien korjaaminen, jota sääntöpohjaisessa päättelyssä tehtiin, saattaisi parantaa tuloksia myös neuroverkkopohjaisessa ratkaisussa. Tämä jää seuraavien hankkeiden tutkittavaksi. Lisäksi kokeilimme ja koulutimme neuroverkon luokittamaan muu-luokan sisältä teksti-, yhdistetty- ja käsinkirjoitettuja sivuja. Alustavat tulokset tämän suhteen ovat myös lupaavia pienellä testiaineistolla (yli 90 % tarkkuus, N=2100).

Neuroverkkojen opettamisessa yksi isoimmista haasteista on kerätä tarpeeksi valmiiksi luokiteltua opetusaineistoa. Neuroverkkoja voidaan pitää ns. “Black Box” -malleina, koska ei voida tutkia, mitä sääntöjä se on oppinut opetusvaiheessa. Sääntöpohjaiset menetelmät eivät välttämättä tarvitse paljoa opetusaineistoa, mutta myös yleispätevien sääntöjen kehittäminen on hankalaa. Aineistosta löytyi loppuvaiheessa kuvia, jotka eivät sisältäneet tekstiä, mutta sisälsivät kuvia. Eli käytännössä tekstintunnistuksen käyttäminen ainoana tyhjyyttä määrittelevänä aineistona on huono idea. Aineisto ja sen ongelmakohdat tulisi projektin alkuvaiheessa tuntea niin hyvin kuin mahdollista, mutta on oltava myös valmis reagoimaan kehittämisen aikana tavattuihin uusiin ongelmiin.

Onkin tapauskohtaisesti ratkaistava, kumpaan suuntaan virheitä voidaan tehdä. Kansallisarkiston tapauksessa tärkein lähtökohta oli, että asiaa sisältäviä tiedostoja ei saisi tunnistaa tyhjiksi. Toiseen suuntaan tapahtuvat virheet - tyhjien tunnistaminen sisällöksi - on vähemmän haitallista tiedonhallinnan ja sen hyödyntämisen näkökulmasta. On myös hyvä muistaa, että tässä esitetyt toimet kohdennetaan ”käyttökappaleisiin”, jotka tarjotaan Kansallisarkiston infrastruktuurista käytettäväksi. Tallekappaleita ei luonnollisesti pyritä lajittelemaan ennen niiden pitkäaikaissäilytykseen viemistä.

Loppukaneettina todettakoon, että tyhjät ja aineistoa sisältävät saadaan eroteltua toisistaan 97–99 % tarkkuudella alle sekunti per sivu nopeudella, joka on jo merkittävä harppaus eteenpäin siitä, että kesäharjoittelija siirtelee tiedostoja kansiosta toiseen käsityönä.