Markus Frosti

TEKOÄLYMENETELMÄ PAKETTIEN SUODATUKSEEN

Opinnäytetyö
Tieto- ja viestintätekniikka

2020

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkinto	Aika
Markus Frosti	Tieto- ja viestintätekniikka (AMK)	Syyskuu 2020
Opinnäytetyön nimi TEKOÄLYMENETELMÄ VERKKOPAKETTIEN SUODATUKSEEN		53 sivua
Toimeksiantaja Innocode Oy		
Ohjaaja Yliopettaja Martti Kettunen Tietoturvainsinööri Jani Hellberg		

Tiivistelmä

Tämän työn tavoite oli kaksiosainen. Työssä selvitetään tekoälyn merkitystä yleisellä tasolla sekä halutaan hyödyntää joitakin tekoälyn oletettuja ominaisuuksia. Näistä ominaisuuksista pyritään ohjelmoimaan koodia, joka poistaa tarpeettomaksi katsottua materiaalia verkkoliikenneraporteista.

Ensimmäinen tutkimusongelma oli se, pystyykö verkkoanalysoijien raporteista suodattamaan pois turhaa tekstiä tekoälyn avulla. Tähän kysymykseen vastaus osoittautui myönteiseksi ja sen selvittämiseksi käytettiin NewtonSoft-nimistä ohjelmointikirjastoa, johon soveltuivat erityisesti JSON-muodossa olevat raportit. Toinen tutkimusongelma oli se, mitä kaikkea tekoäly tarkoittaa. Sitä pyrittiin tutkimaan erilaisten demonstraatioiden ja teorioiden avulla, jotka löytyivät valtaosin The Elements of AI -verkkokurssilta saaduista tiedoista ja TensorFlow-tekoälyrakennussovelluksen oppikirjasta.

Tutkimusmenetelmänä käytettiin kehittämistutkimusta, eli tavoitteena oli parantaa aikaisempaa toimintaa laadukkaammaksi matemaattisluonnontieteellisin keinoin. Tässä tapauksessa pyrittiin siis suunnittelemaan aikaisempi pakettien etsintä ja poisto raporteista helpommaksi ohjelmoimalla siihen sopiva koodi. Työ alkoi sopivia ohjelmointikirjastoja etsimällä. Lopulta huomattiin NewtonSoft-nimisen kirjaston hyödyt. Kirjaston toimintojen tutkimisen jälkeen lopputuloksena syntyi heikkoa tekoälyä mukaileva ohjelma, joka pystyy käyttäjän antamien sanojen perusteella poistamaan raporteista pakettien ilmentymiä. Annettavina sanoina toimivat raporttien pakettien tietoja kuvaavat sanat ja näiden tietojen arvot. Siinä on myös käänteistoiminto niin, että ohjelma säilyttää annettuja sanoja sisältävät paketit poistaen kaiken muun.

Johtopäätösten mukaan ohjelma on luotettava ja edelleen kehittämiskelpoinen sovellus, joka täyttää opinnäytetyön teoriaosuudessa kuvaillut ominaisuudet heikosta tekoälystä. Työn kehittämistä odotetaan jatkettavan TensorFlow-sovelluksen avulla. Opittua tekoälyn teoriaa pyritään soveltamaan tuleviin projekteihin. Koska työ oli oletettua haastavampi ja laajempi, siinä opittiin sivuhyötynä myös asiakirjojen muokkaamista erilaisissa formaateissa ohjelmoinnin avulla.

Asiasanat
koneoppiminen, ohjelmointi, suodatus, kehittäminen

South-Eastern Finland University of Applied Sciences

Author (authors)	Degree	Time
Markus Frosti	Bachelor of Engineering	September 2020

Thesis title

AN AI METHOD FOR FILTERING NETWORK PACKETS

53 pages

Commissioned by

Innocode Oy

Supervisor

Senior Principal Martti Kettunen
Cyber Security Engineer Jani Hellberg

Abstract

The objective of the thesis was divided in two parts. The meaning of AI (artificial intelligence) in general will be examined in the work and some assumed features in AI will be utilized while programming code that erases unnecessary material from network traffic reports.

The first research problem was doing the filtering of unnecessary text from reports made by network traffic analyzers with AI. Another research problem was what AI actually means. The latter problem was to be researched with the help of different demonstrations and theories, which were mostly based on the material of an online course called “The Elements of AI” and the guide to an AI application called “TensorFlow”. The former problem was researched via building the actual program.

The research method was a design-based research. In this case, it meant that earlier invented packet search and removal from reports were made easier by programming the right code for them. The programming library called “NewtonSoft” was the most important factor for the code. After having investigated the documentation of the library, a program simulating weak AI that could remove instances of packets from JSON reports via inputs started to develop.

The conclusion was that the program was reliable with potential for further development, which met the requirements for a weak AI method described in the theory chapter of the thesis. The development of this program might be continued with the help of TensorFlow. The theory of AI studied in the thesis was found applicable to future projects. Due to the thesis having been wider and more challenging than initially assumed, editing documents in different formats with programming was also studied.

Keywords

machine learning, programming, filtering, development

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on selvittää, pystyykö WireShark-verkkoliikenneanalysoijan raportin tekstistä poistamaan automaattisesti tekoälyn avulla tekstiä, joka ei kuulu ollenkaan itse raportin sisältöön ja joka haittaa raportin lukemista. Jos työ sujuu onnistuneesti, lopputuloksena syntyy sovellus, joka poistaa verkkopaketteja raportista.

1.1 Tutkimuksen syyt

Syyt tämän työn valintaan ovat yrityksen tarpeet ja toimeksianto. Tietotekniikkayritykset käyttävät tyypillisesti verkkoanalysoijia työssään. Näiden sovellusten tuottama informaatio voidaan siirtää raporttiin luettaviksi. Verkkoliikenne voi kuitenkin sisältää runsaasti informaatiota ja joskus joitakin osia tästä informaatiosta ei tarvita. Tarpeettomat tiedot haittaavat lukijaa ja siten ratkaisu asiaan löytyisi suodatinohjelmasta, joka poistaa tarpeettomat tiedot ohjelman käyttäjän pyytäessä sitä. Siten raportin lukeminen ja ne työt, joissa tällaista raporttia käytetään, helpottuisivat huomattavasti.

Edellisen syyn lisäksi tämä työ tarjoaa lukijoille lisää tietoa koodin ja verkkoliikenteen toiminnasta sekä lisävarmistuksen siihen, kykeneekö ihmisen lukemiselle tarpeetonta pakettikirjoitusta todella poistamaan verkkoliikenneraportista.

ICT-alan yritys Innocode Oy on toimeksiantaja tässä työssä ja sen henkilökunta on suositellut ohjelmia nimeltä Owasp ZAP, Zenmap ja WireShark projektin suorittamista varten. Innocode on myös yksi niistä yrityksistä, jotka käyttävät WireShark-ohjelmaa työssään, joten heille tämä työ tulee olemaan merkittävä.

1.2 Lähteiden ja muun avun hankinta

Tarvittaessa jokaiseen ohjelmaan selvitettiin apua sosiaalisesta mediasta tai foorumeilta. Lähteiden etsintään käytetään Google Scholar -hakupalvelua. Kirjallisuutta on saatavilla WireShark-dokumentista ja ohjelmoinnista, mutta itse paketinsuodatuksesta sitä on vaikeampaa löytää. Työskentely tapahtui pääasiassa yksin, mutta tilanteen mukaan neuvoa voi kysyä myös työtovereilta ja lähipiiriltä. Pakettiensuodatuksen aikaisemmasta tutkimuksesta on todisteena

WireShark-ohjelman työkaluihin kuuluva suodatusominaisuus (Sanders 2018), joten se lisää mahdollisuuksia tekoälyominaisuuden sisällyttämisen rakennettavaan ohjelmaan.

1.3 Tutkimuskysymykset

Tutkimuskysymyksiin kuuluvat seuraavat olennaiset kysymykset: Pystyykö verkkoanalysoijien raporteista suodattamaan pois turhaa tekstiä tekoälyn avulla? Tämä on olennaisin kysymys, sillä se on koko projektin pääasiallinen tutkimus. Toinen kysymys on, että mitä tekoälyllä tarkoitetaan. Tekoäly otettiin käyttöön terminä vuonna 1956 (Deshpande & Kumar 2018, 16), mutta sitä on suhteellisen vaikeaa määritellä, koska sillä ei ole yleisesti hyväksyttyä määritelmää, ja sen määrittely kehittyy koko ajan. Esimerkiksi tietyt epävarman tiedon käsittelymenetelmät laskettiin aikaisemmin tekoälyksi, mutta nykyään ne määritellään tilastotieteeksi tai todennäköisyyslaskennaksi. On erittäin hankalaa arvioida, riittääkö tehtävän haasteellisuus antamaan synteettiselle ja toimintakykyiselle tehtävänsuorittajalle tekoälyluokituksen. Myös monet populaarikulttuurin osat antavat liioittelevan kuvan siitä, millainen tekoäly olisi.

Tekoälyn olennaisiksi osiksi määritellään nykyään autonomisuus ja adaptiivisuus. Autonomisuus tarkoittaa kykyä suorittaa tehtäviä monimutkaisessa ympäristössä ilman käyttäjän jatkuvaa vaikuttamista ja adaptiivisuus tarkoittaa suorituskyvyn parantamista oppimalla kokemuksesta. Valmistuva suodatin suuntautuu enemmän autonomisuuden puolelle ja jos siitä aiotaan tehdä adaptiivisempi, sen suodatusprosessi olisi toistettava useamman kuin yhden kerran, sillä muuten oppimisesta huolimatta suodatin voi poistaa raportista tai jättää raporttiin ne paketit, joita ei ole tarkoitus poistaa tai jättää.

Tekoälylaitteelle annettava sana saattaa ikävässä tapauksessa olla matkalaukkusana eli sana, jolla voi olla useampi merkitys kuin se, mitä puhuja tarkoitti. Esimerkiksi sanalla ”älykäs” on niin monta haaraa, että sitä on hankala määritellä. Samaa voisi verrata myös siihen, että pystyykö tekoälylaite erottamaan kaksi samaa sanaa, joista toinen olisi toivottu ja oikeassa paikassa raporttia ja toinen toivottu ja väärässä paikassa. Tässä työssä tekoälyn on jotenkin löydettävä keino erottaa tällaiset sanat toisistaan, jos siihen tilanteeseen päädytään.

1.4 Tekoälyn kestävä kehitys

Tekoälyä kehitellään koko ajan niin, että siihen voi sisällyttää yhä useampia ihmiselle suunnattuja toimintoja (Grandi passi in avanti… 2020). Syynä tähän on se, että tekoälyn tavoite yleisesti on laajentaa ja täydentää ihmisten elämää (Deshpande & Kumar 2018, 21). Jotkut toiminnot on suunniteltu jopa ylittämään ihmisen tarkkuudessa, esimerkiksi ruoan suunnittelu ulkoilman lämpötilan mukaan (Marr 2019, 94) tai laadun tarkkailu (mts. 106). Tekoälylle on olemassa kaksi luokitusta: yleinen tekoäly, jonka kohteena on useita älyllisiä toimintoja suorittavat laitteet ilman tarkempaa erikoistumista, ja kapea tekoäly, jonka kohteena on vain kerran tietyn toiminnon suorittavat laitteet. Niin kuin kaikki muutkin nykypäivän tekoälylaitteet, myös tämän työn sovellus tulee edustamaan kapeaa ja heikkoa tekoälyä (Deshpande & Kumar 2018, 325). Kapeat tekoälyt itsessään ovat jo merkittävä teknologinen saavutus, mutta jos yleiset tekoälylaitteet tulevat voimaan, niille voisi keksiä muutakin hyötyä kuin vain niiden toiminnat. Esimerkiksi kun kapeiden tekoälylaitteiden ja muiden työlaitteiden määrä vähenee niin, että yleiset tekoälylaitteet tulevat tilalle, yleisillä tekoälylaitteilla säästetään resursseja. Tosin tekoälylaitteiden lisääntyessä ihmisten tarve erilaisille töille vähenee ja vähenisi entisestään, jos yleisiä tekoälylaitteita keksittäisiin. Yleisille tekoälyille voisi kehittää myös omat työvuorot.

Ihmisten töiden ja elinkeinojen suuntautuessa tulevaisuudessa enemmän tekoälyyn heidän kykynsä mukautetaan enemmän tekoälyn eri aloihin, kuten rakentamiseen ja testaamiseen. Tämä työ oli hyvää esivalmistelua tällaiseen tilanteeseen.

1.5 Koneoppimisen lajit

Koneoppiminen tarkoittaa järjestelmien suorituskyvyn parantumista tietyssä tehtävässä kokemuksen tai datan kertyessä. Koneoppiminen on myös hyvä lähestymistapa tekoälyyn liittyviin ongelmiin. Koneoppimisen olennaisimmat juuret löytyvät tilastotieteestä, joiden osa-alueet voivat olla hyvinkin vanhoja. Koneoppiminen kuuluu eräänlaiseen teknologiseen sukupuuhun; syväoppiminen on koneoppimisen osa-alue, koneoppiminen tekoälyn osa-alue ja tekoäly sekä informaatioteknologia tietojenkäsittelytieteen osa-alue. Toisaalta taas

Deshpanden ja Kumarin mukaan (2018, 21) koneoppiminen on syvän oppimisen osa-alue, joka taas on tekoälyn osa-alue. Tämä vaikuttaa epäloogisemmalta ensiksi mainittuun hierarkiaan verrattuna, sillä syvä oppiminen on viralistettu koneoppimisen alueelle ja sillä tarkoitetaan nimensä mukaisesti koneoppimista syvempää oppimista. Syvä oppiminen on myös vain yksi koneoppimisen haara muiden joukossa, kuten ohjatun ja ei-ohjatun oppimisen.

Ohjatussa koneoppimisessa tutkittavat asiat jaetaan luokkiin tai lukuarvoihin, joista koneen olisi sitten osattava päätellä, edustaako tietty objekti tiettyä luokkaa tai lukuarvoa vai ei. Tämä pohjautuu yksinkertaisimmillaan binäärisiin luokitteluongelmiin. Ohjattua koneoppimista varten on olemassa opetusdataa, joka tarkoittaa joukkoa oikealla vastauksella merkittyjä esimerkkejä. Tällä lailla luodaan sopiva ennustinfunktio (Deshpande & Kumar 2018, 55), jonka avulla tekoälymenetelmä ”oppii” tunnistamaan automaattisesti oikeat vastaukset annettujen esimerkkien kohdalla (Gulli & Pal 2017, 16). Jonkun on sitä varten ohjattava tekoälymenetelmä oikeisiin vastauksiin paljastamalla ne suoraan, mistä tulee ohjatun koneoppimisen termi.

Jos opiskelee koneoppimista, on saattanut törmätä termiin MNIST eli Modified National Institute of Standard. MNIST-tietokanta on hyvä esimerkki erittäin yleisesti käytettävästä datajoukosta koneoppimisessa, johon sisältyy esimerkiksi pienoiskuvia. Koneoppimisessa voi syntyä virheitä, joista suurimmat voi välttää jakamalla datajoukon kahteen ryhmään, eli opetusjoukkoon ja testijoukkoon. Oletuksen mukaan opetusjoukolle syötetään tietoa opetusdatasta niin että syntyy malli, joka ennustaa oikean vastauksen mille tahansa syötteelle. Ja jos halutaan todella testata, pystyykö tämä malli ennustamaan oikean vastauksen halutun tavoitteen mukaan, käytetään testidataa.

On syytä muistaa, että vaikka opetusdatasta saadun tiedon voisi säätää niin tarkaksi kuin mahdollista, sillä on sen myötä riski ylisovittua testidatassa, mikä heikentää testituloksen tarkkuutta. Merkittävä osa koneoppimistutkimusta on nimenomaan ylisovituksen välttäminen. Koneoppimismenetelmät ovat erityisen alttiita ylisovittamiseen, koska ne pystyvät kokeilemaan useita erilaisia sääntöjä, kunnes löytyy yksi opetusdataan täydellisesti sopiva sääntö (mts. 38). Ohjelmalla on riski ylirasittua tämän myötä. Siksi on järkevää testattava

erilaisia loogisia sääntöjä tässä opinnäytetyössä, jos aikoo soveltaa koneoppimista raportin puhdistamiseen.

Ohjaamaton koneoppiminen on hyvä mahdollinen ratkaisu tähän työhön, sillä on epäilyttävää, että tämän työn sisältämästä joukosta satunnaisia sanoja löytyisi datajoukkoa. Ohjattuun koneoppimiseen liittyvät oikeat vastaukset tulevat muutenkin entistä subjektiivisemmiksi raporttitekstissä. Jotta ohjaamaton versio toimisi tässä työssä, tietynlaisia rakenteita on löydettävä. Tässä on mahdollista käyttää klusterointia vaikka sen perusteella, sijoittuvatko siivottavat sanat raportin tekstialueen ulkopuolelle. Myös visualisointia voisi käyttää sanojen mahdolliseksi siirtämiseksi raportin ulkopuolelle. (Reaktor & Helsingin yliopisto 2018.) Eräs hyvin tehokas ryppäysalgoritmi on k-means (Shukla 2018, s. 16), jossa etsitään tietty määrä klustereita datajoukossa, jotta voidaan sijoittaa datapisteitä niihin (Garbade 2018; Deshpande & Kumar 2018, 75).

1.6 Tutkimusmenetelmä

Kvalitatiivisessa tutkimuksessa tutkitaan jonkin asian laatua, esimerkiksi sitä, kuinka hyvä jokin asia on jossakin tehtävässä, toiminnossa, asiassa tai vastaavassa. Kvantitatiivisessa tutkimuksessa tutkitaan määrällisiä ominaisuuksia, esimerkiksi että kuinka suuri osuus jokin on jostakin. Tämä työ ei kuitenkaan ole pääasiallisesti laaduntutkimusta, vaikka loppuvaihe kuluukin kehittämismahdollisuuksien tarkasteluun, eikä siinä ole kyse mielipiteiden keräämisestä.

Tapaustutkimuksessa ei pyritä tekemään erityistä muutosta eikä olemaan tutkimuksessa aktiivisesti mukana, vaan tutkimus rajoittuu jonkin asian analyysiin ja uusien ominaisuuksien raportointiin. Tässä työssä kuitenkin pyrittiin enemmänkin kehittämään eikä analysoimaan.

Toimintatutkimuksessa ollaan itse läsnä tutkimuksessa tehden esimerkiksi haastatteluja ja mittauksia tietystä aiheesta, mutta tässä työssä ei tarvita haastatteluja ja mielipiteitä johonkin olosuhteeseen, vaan tietoa siitä, kuinka luoda haluttu ohjelmakokonaisuus.

Sekä etno- että netnografinen tutkimus vaativat kummatkin jossakin yhteisössä olemista ja raportointia yhteisön olosuhteista, mikä on tämän työn aiheen ulkopuolella. Haluamani kokonaisuuden rakenteen selvittämiseksi ei vaadita aktiivista yhteisön jäsenenä oloa. (Kananen 2017.)

Kehittämistutkimus sen sijaan oli lähimpänä tähän työhön soveltuvaa tutkimusmenetelmää. Siinä kehitettiin valmista tuotetta entistä laadukkaammaksi, jos mahdollista. Toimintatutkimus muistuttaa hyvin paljon kehittämistutkimusta, jopa siinä määrin, että suomalaisittain ne eivät tunnu juuri eroavan toisistaan, mutta kansainvälisesti ajateltuna toimintatutkimus keskittyy enemmän niihin asioihin, joilla on sosiaalinen pohja, ja kehittämistutkimus niihin asioihin, joilla on teknologinen pohja tai joka on ihmisen muokattavissa matemaattis-luonnontieteellisiä ominaisuuksia hyödyntäen. (Goldkuhl 2013.)

2 DEMONSTRAATIOITA TEKOÄLYSTÄ

On olemassa useita tapoja toteuttaa tekoälymenetelmä. Niissä hyödynnetään ohjelmointia esimerkiksi todennäköisyyslaskentoihin, erilaisiin vaihtoehtoihin tai opetettaviin asioihin. Jokin tilanne voidaan esimerkiksi purkaa yhä useampiin vaihtoehtoihin, kunnes saavutetaan vaihtoehtojen päätepisteet, ja sitten ohjelmoidaan tekoälymenetelmä siirtymään tiettyyn vaihtoehtoon tai tekemään laskenta jonkin tilanteen perusteella. Tätä lukua varten otetut kuvat ja niiden selitykset perustuvat The Elements of AI -verkkokurssin (2018) demonstrointeihin. Kaikki opinnäytetyön kuvat löytyvät työn lopusta liitteenä.

2.1 Regressio

Regressio on ohjattua koneoppimista siinä missä luokittelukin (Deshpande & Kumar 2018, 57–58), mutta siinä käytetään luokkien sijaan lukuarvoja. Regressiomenetelmät tuottavat vastaukseksi lukuarvon, jonka ei tarvitse olla kokonaisluku. Tällä voi teoriassa mitata esimerkiksi sanojen sijaintiin liittyviä suureita ja todennäköisyyksiä.

Lineaariregressio tarkoittaa laskentatapaa, jossa lopullisen arvon kasvu on riippuvainen toisen arvon kasvusta. Esimerkiksi jos on ostanut omenoita ja ostosten hinta nousee, hinta nousee vielä lisää, jos ostaa banaaneja. Tästä ei

kerrota tässä tekstissä enempää, sillä tämän työn kannalta logistinen regressio on merkittävämpi.

Logistisessa regressiossa otetaan lineaariregression loppuarvo, joka sitten merkitään positiiviseksi tai negatiiviseksi. Tätä varten useampia tekijöitä voidaan ottaa mukaan. Esimerkiksi suodatuksessa voi hyvässä tapauksessa soveltaa sellaista logiikkaa, että merkitään, kuinka todennäköisesti raportin tekstialueen ulkopuolelle jäävä sana poistuu raportista. Demonstraatioksi siitä toimikoon kuva 1 (Liite 1), jossa vasen jana mittaa todennäköisyyttä sanan poistumisesta ja alajana mittaa sanan etäisyyttä raportin varsinaiselta tekstialueelta. Ylös merkityt pisteet ovat raportista poistettuja sanoja ja alas merkityt raporttiin jääneitä sanoja.

Jos argumentit määriteltäisiin hypoteettiselle ohjelmointifunktiolle tämän kaavion mukaan, se olisi merkittävässä osassa työn toivottua lopputulosta. Joka tapauksessa tämä antaa hyvän pohjan sille, kuinka soveltaa tekoälymenetelmän suunnittelussa todennäköisyyksiä.

2.2 Ristinollamenetelmä

Tässä selitetään, kuinka tietokone voi laskea mahdolliset virheelliset siirrot ja oikeat siirrot ja valita niiden valintojen perusteella, minkä vaihtoehdon valita. Tämä käy mahdollisena työn aiheena, sillä kyky valita on yksi tekoälyn ominaisuuksista. Prosessi etenee niin, että otetaan ristinollan tietty tilanne ja lasketaan siitä kaikki mahdolliset seuraavat vaihtoehdot. Sama prosessi tehdään näiden vaihtoehtojen kohdalla, kunnes saavutetaan jokainen mahdollinen lopputulos. Näin syntyy tietynlainen ylösalaisin oleva puu, niin kuin hakemistopuu tietokonejärjestelmissä.

Seuraavaksi merkitään kaikki ne vaihtoehtopolut, joiden lopputuloksena on voitto, erikseen niistä poluista, joista ei synny voittoa. Tällöin syntyy ohjelman tie voittoon ristinollassa. Kuva 2 (Liite 1) havainnollistaa tätä prosessia, jossa arvo 1 on punaisen pelaajan voitto ja arvo 0 edustaa tasapeliä. Voimme nähdä, että mahdollisuutta vihreän voittoon ei ole, joten vihreä joutuu tyytymään tasapeliin parasta mahdollista lopputilannetta varten.

2.3 Bayesin kaava

Jos tämä työ käyttää jotakin menetelmää todennäköisyyksien laskemiseksi tekstin ollessa väärällä tai oikealla paikalla raporttia, Bayesin kaava on yksi vaihtoehto. Tämän kaava punnitsee ristiriitaista informaatiota useilla tieteenaloilla ja tällainen informaatio tulee ilmi myös tässä työssä, jos sitä kehittää tarpeeksi pitkälle.

Bayesin kaavasta on erilaisia versioita, mutta tässä esitetään sen yksinkertainen versio. Olennaiset elementit ovat tässä versiossa priori- ja posteriorikertoimet sekä uskottavuusosamäärä. Priorikerroin mittaa tilannetta ennen muutoksia ja posteriorikerroin mittaa tilanteen todennäköisyyttä muutoksen jälkeen. (BetterExplained s.a.) Sen kaava menee seuraavasti:

posteriorikerroin = uskottavuusosamäärä × priorikerroin

Esimerkkinä siitä, mitä tästä kaavasta voi päätellä, olkoon tilanne, jossa eräässä kylässä nimeltä Kukkainen kulkee Volkswagen- ja Toyota-autoja säännöllisin väliajoin. On arvioitu, että näitä autoja tulee suhteessa x:y. (x = Volkswagen, y = Toyota) Tätä tilannetta päätetään tarkkailla eräänä viikkona ja uuden havainnon mukaan nähtiin 25 Volkswagen-autoa ja 28 Toyota-autoa. Kukkaisessa sijaitsevan Mansikkala-nimisen kaupan vierestä on kulkenut yhteensä viisi Volkswagen-autoa ja seitsemän Toyota-autoa. Illoin Kukkaisessa on kulkenut yhteensä viisi Volkswagen-autoa ja kuusi Toyota-autoa. Sateisella kelillä paikassa on kulkenut yhteensä kolme Volkswagen-autoa ja kaksi Toyota-autoa. Pääsiäisenä paikassa on kulkenut 8 Volkswagen-autoa ja 10 Toyota-autoa. Kumpi auto ilmestyy todennäköisemmin Mansikkalan lähellä sateisella illalla pääsiäispäivänä?

Seuraavaksi lasketaan Mansikkalassa, illalla ja sateisella kelillä syntyvät uskottavuusosamäärät:

Mansikkala: (5 : 7) / (25 : 28) = 4/5 (Toyota-autoja 1,25 kertaa todennäköisemmin.)
Ilta: (5 : 6) / (25 : 28) = 14/15 (Toyota-autoja noin 1,07 kertaa todennäköisemmin.)

Sade: (3 : 2) / (25 : 28) = 42/25 (Volkswagen-autoja 1,68 kertaa todennäköisemmin.)
Pääsiäinen: (8 / 10) / (25 : 28) = 112/125 (Toyota-autoja noin 1,12 kertaa todennäköisemmin.)

Nyt kun meillä on uskottavuusosamäärät, voimme laskea niiden avulla uuden arvioidun suhteen Volkswagen- ja Toyota-autojen ilmaantumiselle:

$$(x / y) \times (4 / 5) \times (14 / 15) \times (42 / 25) = (784 / 625) \times (x / y)$$

Tämän uuden arvion mukaan jokaista 784x esiintyvää Volkswagen-autoa kohden esiintyy myös 625y Toyota-autoa.

2.4 Naiivi Bayesin luokitin

Naiivia Bayesin luokitinta käytetään, kun on olemassa useita eri havaintoja, joiden avulla voi määritellä luokat ja laskea niiden todennäköisyydet. Termi ”naiivi” tulee siitä, että luokitin käyttää selvästi yksinkertaisia keinoja kohteiden luokitteluun, esimerkiksi sitä keinoa, että luokitellaan jokin teksti tietynlaiseksi vain tietyn sanan perusteella sanajärjestyksen sijaan.

Naiivi Bayesin luokitin perustuu oletukseen, että havainnot ovat toisistaan riippumattomia. Klassinen esimerkki tästä luokittimesta on roskapostisuodatus. Otetaan esimerkkinä tapaus, jossa sovelletaan kaksiosaista versiota moniosaisesta naiivista Bayesin luokittimesta (Deshpande & Kumar 2018, 181). Tässä tapauksessa on laskettu tiettyjen sanojen määrä roskaposteista ja asiallisista viesteistä ja josta voidaan päätellä, onko seuraava viesti todennäköisemmin asiallinen viesti vai roskaposti.

sana	**roskaposti**	**asialliset viestit**
omena	37	78
peitto	27	43
raha	108	27
tiili	40	54
yhteensä	53281	203147

Taulukko 1. Naiivi Bayesin luokitin, osa 1.

Nyt voidaan arvioida todennäköisyydet sanoille roskapostiviestissä ja asiallisessa viestissä. Esimerkiksi todennäköisyys sanalle ”omena” roskapostiviestissä on 37 / 53281 = 1 / 1440 ja asiallisessa viestissä 78 / 203147 = 1 / 2604. Nyt lasketaan näiden uskottavuusosamäärä:

(1 / 1440) / (1 / 2604) = 1,8

Lasketaan samalla logiikalla muiden sanojen uskottavuusosamäärät.

sana	**uskottavuusosamäärä**
omena	1.8
peitto	2.4
raha	15.3
tiili	2.8

Taulukko 2. Naiivi Bayesin luokitin, osa 2.

Laskusta 1:1*1.8*2.4*15.3*2.8 = 185.1 voimme päätellä, että on olemassa 185 roskapostia jokaista asiallista viestiä kohden. Toisin sanoen on erittäin todennäköistä, että seuraava viesti on roskapostia.

Asian ydin on se, että tätäkin voi soveltaa työn raportin siivoamiseen tehokkaana tekoälyluokitusmenetelmänä.

3 TENSORFLOW

Tekoälyn kehitys tarkoittaa sitä, että myös tekoälymenetelmät ovat kehittyneet. Yksi suosittu esimerkki on TensorFlow, jolla voi luoda tekoälymenetelmän sovellukselle ominaisten kerrosten avulla. Seuraavat tiedot perustuvat TensorFlow-tutoriaalin (2020) virallisiin kuvauksiin. Tämä luku sisältää myös koodinpätkiä, jotka tullaan esittämään harmaalla taustalla tutoriaalista otettujen koodinpätkien mukaisesti.

3.1 Perusteet

TensorFlow on koneoppimisen sovellus, joka tukee Python- ja C++-kieliä. TensorFlow-kirjaston asennuksen yhteydessä asennettiin myös NumPy-kirjasto, joka sisältää ammattilaistasoisia laskentametodeja (NumPy s.a.). Sieltä löytyy kuitenkin haavoittuvuudeksi määriteltävä bugi, joten kyseisen kirjaston käyttämistä kannattaa turvallisuuden kannalta harkita (Xiao ym. 2018).

TensorFlow on saanut nimensä siitä, että se käyttää tensoreita, joissa on useampi kuin kaksi tietyn aiheen luokitteluelementtiä. Otetaan esimerkiksi hedelmäkori, jossa on joukko omenoita ja päärynöitä (Liite 1 Kuva 3). Puolet omenista ja päärynöistä on mädäntyneitä. Tässä tapauksessa hedelmäkoria edustavassa tensorissa käytetään kahta matriisia, joiden sisällä on syötävät ja pilaantuneet hedelmät. Näiden matriisien sisällä puolestaan on omenista ja päärynöistä koostuvat ominaisuusvektorit. Tensori on siis useasta matriisista koostuva kokonaisuus, joita TensorFlow käyttää neuraaliverkkolaskennoissa. (Shukla 2018, 27–28.)

3.2 Asentaminen

TensorFlow vaati ensiksi asennettavan paketinhallintajärjestelmän nimeltä pip ja samaan aikaan Python-ohjelmointiympäristön. Ohjelmointialusta nimeltä PyCharm ja virtuaaliympäristö asennettiin erikseen.

Erään lähteen mukaan TensorFlow tukisi Python-ympäristön versiota 3.6 (awav 2019), joten ympäristön asennus onnistui versiolla 3.6.8.

TensorFlowin tutoriaaleja varten oli asennettava matemaattisten laskentojen kirjasto nimeltä NumPy ja sen versio 1.16.4.

3.3 Neuraaliverkon kerrosten toiminta

Tekstin yksi edustustapa on kääntää syötedatan sisältämät lauseet upotusvektoreiksi. Upotusvektorit ovat liukuluvuista koostuvia tiheitä vektoreita, jotka oppivat painokertoimet niiden syöttämisen sijaan (Koehrsen 2018). Painokerroin tarkoittaa parametria neuraaliverkon sisällä, joka muuttaa syötedataa piilotettujen kerrosten sisällä (DeepAI s.a.; Halonen 2019). Etukäteen treenattua tekstiä voi käyttää ensimmäisenä kerroksena.

Neuraaliverkko koostuu kerroksista, jotka ovat toisiinsa yhteydessä olevia datarykelmiä eli solmuja (Deshpande & Kumar 2018, 263). Ensimmäinen kerros koostuu annetuista syötteistä, piilotetuissa kerroksissa sijaitsevat solmut suorittavat laskentoja näistä syötteistä ja viimeisessä kerroksessa syntyy toivotut arvot.

Tappiofunktio nimeltä binary_crossentropy on hyvä mahdollisuus binääriluokituksiin ja mahdollisuustilanteiden laskemiseen.

3.4 Kuinka välttää ylisovitusta?

TensorFlow-sivun regressiotutoriaalissa ohjelma kertoi yrityksessä ladata tarvittavat ohjelmat ja käynnistää ohjelma, ettei tensorflow.keras-moduulia ollut olemassa. Keras tarkoittaa tehokasta syvän oppimisen ohjelmointikirjastoa. Tensorflow.keras-kirjasto ei tukenut TensorFlowin versiota 1.5, joten versio 2.0 täytyi ladata. Tutoriaalin jälkeen selvisi jotain tärkeää: ei kannata käyttää liian paljon kerroksia tai pysäyttää aikaisin oppiminen ylisovituksen riskin vuoksi.

Yksi keino estää ylisovitusta on aloittaa pienistä ja yksinkertaisista malleista esimerkiksi käyttämällä layers.Dense-funktiota pohjana. Näitä malleja laajennetaan sitten hiljattain, kunnes validointitappion arvo alkaa palata. Toisaalta jos malli on liian pieni, tarpeeksi informaatiota ei mahdu. Validointitappio on yksi niistä kahdesta arvosta, joiden avulla optimoidaan tekoälymenetelmä huippuunsa; toinen on opetustappio. Laajentamalla malleja hiljattain selvisi, että keski- ja suurikokoisissa malleissa validointitappio palasi takaisin. Myös

TensorFlow-sovelluksen tiedot graafisiksi muuttavasta TensorBoard-työkalusta väitettiin kykenevän katsomaan näitä tietoja, mutta sitä ei koettu tarpeelliseksi ja resursseja siihen oli liian hankalaa saada.

Tutoriaali kertoi myös painokertoimien vakinaistamisprosessista, jossa mallin painokertoimia pienennetään mallin yksinkertaistamiseksi ja sitä myötä ylisovituksen riskin vähentämiseksi. Tämä tehtiin lisäämällä neuraaliverkon tappiofunktioon cost-arvo, joka yhdistettiin suuriin painokertoimiin. Tämä on saatavissa sekä L1- että L2-vakinaistamisena. Edellisessä tapauksessa lisättävä cost-arvo on verrannollinen painokerrointen absoluuttiseen arvoon, jälkimmäisessä se on verrannollinen painokerrointen neliöarvoon. Kirjain L tulee sanasta loss eli tappio.

Kahta tärkeää seikkaa korostettiin vakinaistamisessa: sitä, että jos omaa opetussilmukkaa kirjoittaa, täytyy kysyä mallilta vakinaistamistappioita, ja sitä, että vakinaistamistaktiikka toimii lisäämällä painomaksuja mallin tappioille, jolloin sovelletaan standardia optimointia.

Yksi opetusmekaniikkaan suuntaavista ohjelmointimenetelmistä on, että yksi syöttöarvoista välillä 0.2–0.5 pudotetaan nollaan, jotta kone tietää, että opetuksesta on kyse, eikä ”aidosta” oppimisesta.

3.5 Mallien rakennus

On olemassa kaksi tapaa, jolla tekoälyyn erikoistuvan Keras-ohjelmointikirjaston malleja voi rakentaa: peräkkäinen (sequential) ja toiminnallinen (functional) API. API tulee sanoista Application Programming Interface, joka tarkoittaa ohjelmointirajapintaa. Ohjelmointirajapinnan avulla ohjelmat voivat olla yhteydessä keskenään. Peräkkäinen API sallii mallien luomisen kerros kerrokselta useampiin ongelmiin ja sen rajoitteena on, ettei se salli kerroksia jakavien mallien luomista tai MIMO-tekniikkaa eli Multiple Input and Multiple Output -tekniikkaa, joka tarkoittaa useita antenniyhteyksiä kerralla. Toiminnallinen API taas sallii monipuolisten mallien luomisen, jossa kerrokset voi yhdistää mihin tahansa muuhun samassa mallissa olevaan kerrokseen. Siinä kykenee myös hyödyntämään MIMO-tekniikkaa. (Lin 2017.)

Peräkkäisen mallin olennaiset osat ovat tappiofunktio, optimoija ja metriikat. Tappiofunktio mittaa, kuinka tarkka malli on treenauksen aikana ja malli päivittyy sen sekä optimoijan näkemän datan avulla. Metriikoita käytetään koneoppimisessa treenauksen ja testaamisen tarkkailuun.

Funktionaalisessa API-rajapinnassa luodaan peräkkäistä joustavampia malleja. Mallin rakentamiseksi luodaan syöttösolmu. Toiminnot inputs.shape ja inputs.dtype sisältävät tietoa tämän solmun muodosta ja tyypistä.

Tämän jälkeen seuraa vaihe, jossa jokainen solmu sisältää arvon ja yhden osoittimen, joka osoittaa seuraavaan solmuun, kunnes tulee NULL-arvon eli nolla-arvon sisältävä osoitin. Funktionaalisessa mallissa tallennetaan muuttujaan kerros niin, että samaan muuttujaan tallennetaan uusi kerros, johon on liitetty tämä aikaisempi kerros. Tätä kutsutaan kerrosten rekursiiviseksi kokoamiseksi.

```
from tensorflow.keras import layers

inputs = keras.Input(shape=(784,))
dense = layers.Dense(64, activation='relu')
x = dense(inputs)

x = layers.Dense(64, activation='relu')(x)
outputs = layers.Dense(10)(x)

model = keras.Model(inputs=inputs, outputs=outputs)

inputs = keras.Input(shape=(784,), name='img')
x = layers.Dense(64, activation='relu')(inputs)
x = layers.Dense(64, activation='relu')(x)
outputs = layers.Dense(10)(x)

model = keras.Model(inputs=inputs, outputs=outputs,
name='mnist_model')
```

Koodinpätkässä muuttujaan nimeltä dense tallennetaan ensimmäinen kerros. Muuttujaan x tallennetaan jatkuvasti uusia kerroksia, kunnes luodaan viimeinen kerros outputs. Mallin rakentamiseen kuuluvat datan opetus, sen evaluaatio ja siitä tehtävät päätelmät, ja ne toimivat täysin samalla lailla kuin peräkkäisessä mallissa, joten se jätettiin pois koodinpätkästä.

3.6 Mallien tallennus

Tässä on yksi mahdollinen keino tallentaa malli:

pip install -q pyyaml h5py

Tällä saatiin koodi tallennettua HDF5-formaatissa, joka tulee sanoista Hierarchical Data Format ja joka tukee suurta, monipuolista ja heterogeenistä eli monityyppistä ja -formaattista dataa (Wasser s.a.). Yksi keino tällaiseen tallennukseen oli painokertoimien tallentaminen. Tutoriaalissa näytettiin myös esimerkki koodista, joka tallensi painokertoimet ainoastaan opetuksen aikana. Tällaista tallentamista varten mallilla oli oltava samanlainen arkkitehtuuri kuin alkuperäisellä mallilla.

Seuraava vaihtoehto oli SavedModel-formaatilla tallentaminen. Siinä käsiteltiin kuitenkin Linux-pohjaista tallentamista, joten se päätettiin jättää välistä, koska sovellusta TensorFlow käytettiin tässä tapauksessa Windowsissa.

Checkpoint-ominaisuuksien kuvaus oli, että niillä voi tallettaa edellisen mallin ja jatkaa seuraavasta:

```
def train_and_checkpoint(net, manager):
  ckpt.restore(manager.latest_checkpoint)
  if manager.latest_checkpoint:
    print("Restored from {}".format(manager.latest_checkpoint))
  else:
    print("Initializing from scratch.")

  for example in toy_dataset():
    loss = train_step(net, example, opt)
    ckpt.step.assign_add(1)
    if int(ckpt.step) % 10 == 0:
      save_path = manager.save()
      print("Saved checkpoint for step {}: {}".for-
mat(int(ckpt.step), save_path))
      print("loss {:1.2f}".format(loss.numpy()))

train_and_checkpoint(net, manager)
```

```
opt = tf.keras.optimizers.Adam(0.1)
net = Net()
ckpt = tf.train.Checkpoint(step=tf.Variable(1), optimizer=opt,
```

```
net=net)
manager = tf.train.CheckpointManager(ckpt, './tf_ckpts',
max_to_keep=3)

train_and_checkpoint(net, manager)

print(manager.checkpoints)  # List the three remaining checkpoints
```

Train_and_checkpoint-funktio näyttää, että jos edellistä mallia ei ole, se luodaan. Muussa tapauksessa malli tuodaan checkpoint-paikasta. Kummassakin tapauksessa mallista tallennetaan jälkikäteen useampi checkpoint-paikka.

3.7 UniCoden hyödyntäminen

UniCode on koodausmekaniikka, joka sisältää kaikkia kieliä edustavia merkkejä ja jossa jokainen merkki koodataan koodipisteellä väliltä 0–0x10FFFF. Tätä voi yrittää käyttää keinona luoda ilmoitusviesti projektille.

TensorFlow-dokumentaatio sisältää ominaisuuden nimeltä RaggedTensor, joka tarkoittaa tensorin sisältämiä eripituisia elementtejä, kuten matriiseja. Tämä voidaan tarvittaessa korjata täyttämällä puuttuvien merkkien paikat jollakin merkillä, kuten −1:llä:

```
batch_chars_padded = batch_chars_ragged.to_tensor(default_value=-
1)
print(batch_chars_padded.numpy())
```

Koodinpätkässä oleva batch_chars_padded-muuttuja tasoittaa elementtien pituuserot mainitulla tavalla.

Unicode_split-funktio on mahdollisuuden arvoinen tässä projektissa, koska hypoteettinen viesti sisältää toimintoja, jotka vaativat merkkien eristämistä. Funktio jakaa jokaisen funktiolle annetussa syötteessä olevan merkkijonon UniCode-koodipisteiksi. Jokainen koodipiste UniCode-mekaniikassa kuuluu johonkin skriptiin, joka on yksittäinen kokoelma koodipisteitä. Tämä auttaa koodipisteiden luokittelussa ja sitä myötä koneoppimisessa, jos tämä ominaisuus lisätään neuroniverkkoon. Myös segmentaatio, eli merkkijonon jakaminen, on kokeilemisen arvoinen, jos päädytään testaamaan projektin vaiheita pienissä osissa.

3.8 Valmiin datan lukeminen

Kun tutustuttiin TensorFlow-materiaaliin kuuluvan Record-kirjaston lukufunktioon, oletettiin yhtenä lukufunktion vaihtoehdoista, että raportin pystyy sijoittamaan TensorFlow-sovellukseen luettavaksi niin, että se pystyy analysoimaan, onko varoitusviestille tarvetta.

Lukufunktio on seuraavassa koodinpätkässä:

```
filenames = [filename]
raw_dataset = tf.data.TFRecordDataset(filenames)
raw_dataset
```

Tf.data.TFRecordDataset()-funktio edustaa lukufunktiota, johon tallennetaan tiedostojen nimiä.

Sovelluksen IO-moduulin eli In-Out-moduulin funktioista, joissa käsitellään Record-tiedostoja, on myös olemassa puhtaasti Python-kielen versio, joka on kokeilemisen arvoinen.

3.9 RNN-tekniikat

Tutustuttiin SimpleRNN-, GRU- ja LSTM-kerroksiin, jotka edustivat sisäänrakennettuja RNN-kerroksia Keras-kirjastossa. Ne tulevat sanoista Simple Recurrent Neural Network, Gated Recurrent Unit ja Long Short Term Memory. Ne kaikki ovat verkkoarkkitehtuurin muotoja, jotka prosessoivat kokonaisia eriä syöttösekvenssejä. SimpleRNN-kerros ei sisällä yhtään porttia, joka tehostaisi kerroksen toimintaa matemaattisilla laskelmilla. Sen sijaan GRU-kerroksessa on kaksi porttia ja LSTM-kerroksessa neljä. (Rathor 2018.) RNN on lyhenteensä mukaisesti takaisinkytketty neuroverkko, jonka solmut muodostavat ohjatun kuvaajan ja vievät dataa eteenpäin aika-askelittain (Shukla 2018, 185).

Paremman suorituskyvyn takaamiseksi voi käyttää sisäänrakennettua GRU-kerrosta ja LSTM-kerroksia. On olemassa kolme sisäänrakennettua RNN-kerrosta: SimpleRNNCell, GRUCell ja LSTMCell. Näistä voi kehittää teoriaa tätä opinnäytetyötä varten.

RNN-kerros voi myös palauttaa sen lopulliset sisäiset tilat, joita voi käyttää RNN-toteutuksen palauttamiseksi myöhemmin tai uuden RNN-kerroksen aloittamiseksi. Tähän käytetään käskyn return_state arvoa True. Sillä voi yrittää toteuttaa seuraavan paketinsiivoamisen tarkkailun seuraavaa varoitusviestiä varten. LSTM-kerros sisältää kaksi tilatensoria, kun taas GRU-kerroksella on vain yksi. Sitä käytetään kuitenkin tyypillisesti mallissa encoder-decoder sequence-to-sequence, jossa enkooderin lopullista tilaa käytetään dekooderin alkuperäisenä tilana. Oletustilaa varten käytetään käskyä initial_state.

Tavallisesti RNN-kerroksen sisäinen tila uusitaan jokaisen opetusnäyteryhmän jälkeen. Siltä varalta, jos haluaa prosessoida hyvin pitkiä sekvenssejä, kannattaa pätkittää sekvenssi pienemmiksi, jotta ne mahtuvat yksitellen RNN-kerrokseen ilman, että sen tilaa tarvitsee uusia. Tällä lailla kerros voi säilyttää informaatiota koko sekvenssistä, vaikka se näkeekin vain yhden alisekvenssin kerralla. Tämän voi tehdä asettamalla kooditekstin stateful=True rakentajaan. Otetaan esimerkiksi sekvenssi s = [t0, t1, ... t1546, t1547]:

```
s1 = [t0, t1, ... t100]
s2 = [t101, ... t201]
...
s16 = [t1501, ... t1547]

lstm_layer = layers.LSTM(64, stateful=True)
for s in sub_sequences:
  output = lstm_layer(s)
```

Koodinpätkässä sub_sequences-taulukko sisältää pätkityt sekvenssit, joita edustavat numeroidut s-muuttujat. Tämä tulee hyödyksi, jos siivottavat sanat ovat liian pitkiä.

Bidirektionaalisessa RNN-kerroksessa RNN-malli käy läpi sekvenssin ensiksi alusta loppuun, sitten kopioidaan Bidirectional-kerroksen avulla syötetty RNN-kerros ja käännetään uuden kopioidun kerroksen kenttää go_backwards, jotta se prosessoisi syötteet käänteisjärjestyksessä. Lopputuloksena syntyy etu- ja takakerrossyötteiden summa. (Deshpande & Kumar 2018, 118–119.)

3.10 Pehmustus ja maskeeraus

Pehmustus on keino muokata eripituisista taulukoista samankokoisia lisäämällä nollia lyhyempiin taulukoihin. Jos tulevassa testaamisessa joutuu käyttämään eripituisia vektoreita, tätä voi hyödyntää. Se ei kuitenkaan riitä – on käytettävä myös maskeerausta, jossa ilmoitetaan koneelle, että osa vektoreista on pehmustettu ja että ne tulisi sivuuttaa, sillä muuten myös nollat laskettaisiin mukaan laskentaan.

Pehmustusta ja maskeerausta käytettäessä mallissa maski siirretään neuraaliverkkoon niin, että pehmustus ja maskeeraus käy automaattisesti läpi jokaisen kerroksen, joka kykenee käyttämään niitä.

4 TESTATTAVAT RAPORTTISOVELLUKSET

TensorFlow toimii lähinnä testattavana spekulaationa siitä, voiko sitä soveltaa tähän työhön. On kuitenkin sovelluksia, joihin syntyvää ohjelmaa tullaan kokeilemaan, kuten Owasp, Zenmap ja WireShark.

4.1 Owasp ZAP

Owasp ZAP on Owasp-yrityksen luoma penetraatiotestisovellus, joka erikoistuu verkkosivujen testaamiseen. Nimi tulee sanoista Zed Attack Proxy. ZAP toimii niin, että se sijoittuu testaajan selaimen ja verkkosovelluksen väliin, muokkaa tarvittaessa niiden välisen liikenteen sisältöä ja sijoittaa paketin kohteeseensa. Toisin sanoen, ZAP käyttää man-in-the-middle-penetraatiotestitekniikkaa.

Penetraatiotestiprosessiin kuuluu tutkiminen, hyökkäys ja raportointi sovituissa yhteyksissä. Raportissa listataan, kuinka onnistui hyödyntämään mahdollisia haavoittuvuuksia ja kuinka haastavaa se oli (Zap s.a.). Haavoittuvuuksia voivat olla esimerkiksi portit, jotka kuuluvat tässä tapauksessa poistettaviin tietoihin. Ihminen kykenee poistamaan tällaiset tiedot myös manuaalisesti avaamalla raportin formaatissa docx, mutta se vie liian paljon aikaa, jotta se olisi ollut käytännöllistä. Tarkoituksena oli suunnitella koodi, joka poistaisi kaikki täl-

laiset tiedot kerralla. Tähän vaadittiin hyvin todennäköisesti ehto- ja toistolausekkeita. Ohjelmoinnin soveltaminen sovellukseen Microsoft Word ei kuitenkaan ollut heti yksinkertaista, mutta siinäkin auttoi internet.

4.2 Zenmap

Zenmap on virallinen graafinen käyttöliittymä Nmap-verkkoskannausohjelmalle. Se on vapaa avoimen lähdekoodin ohjelmisto, jonka on tarkoitus yksinkertaistaa Nmap-ohjelman asetukset, hyödyt ja havainnot. Skannaustulokset on mahdollista tallentaa ja katsoa myöhemmin, minkä todistaa muun muassa Zenmap-käyttöliittymän raporttiominaisuus. Raportissa voi nähdä kaikki portit, jolloin syntyvää ohjelmaa on mahdollista soveltaa siihen. (Nmap.org s.a.)

4.3 WireShark

WireShark on erittäin suosittu verkkoprotokolla-analysoija, joka sallii verkon tarkkailun mikroskooppisella tasolla (WireShark s.a.). Se kykenee katsomaan verkkoliikenteen paketteihin ja kehyksiin liittyvät osat lähdeosoitteesta kohdeosoitteeseen ja paketin pituudesta Time-To-Live-arvoon eli siihen, kuinka monessa välittäjälaitteessa paketti voi käydä ennen kuin sen lähettäminen eteenpäin lopetetaan (Lamping & Warnicke 2004, 1.1).

IP-osoitteet eli laitteiden kommunikaatiovälineinä toimivat internetprotokollaosoitteet kuuluivat myös Innocode-yrityksen mukaan ”tarpeettomiin tietoihin” ja tehtävään koodiin kannatti erityisesti tätä sovellusta varten yrittää lisätä ehtolauseke, joka ottaa huomioon juuri tällaiset osoitteet.

5 KOODAAMINEN JA TESTAAMINEN

Tässä luvussa esitetään koodinpätkiä joko itse tehtynä tai muista lähteistä otettuina. Edellisessä tapauksessa koodit esitetään kuvina ja jälkimmäisessä tapauksessa koodit esitetään kirjoituksena erillisellä fontilla verrattuna tämän työn tyypilliseen fonttiin.

Esimerkkimateriaali oli riittävä työongelman ymmärtämiseen ja sitä myötä sopivan ratkaisun kehittämiseen. Jälkimmäinen oli todennäköistä ohjelmoinnin ja oikeiden sovellustoimintojen avulla.

Testiympäristöön kuului kolme kameraa, joista kaksi on toiminnassa. Näiden kameroiden kautta tulviva verkkoliikenne virtaa kameroihin yhdistettyyn kannettavaan. Kannettavaan asennetut verkkoliikennesovellukset analysoivat tätä liikennettä ja tekevät siitä raportin, josta lopuksi poistetaan tarpeeton tieto.

Lisäksi työskentely tapahtui yksityisessä verkossa niin, että koodin onnistumisen jälkeen sitä testattiin toimeksiantajan verkkoon. Verkkoliikennesovelluksiksi ladattiin Owasp, Zenmap ja Wireshark.

Työn valmistelussa tutkittiin useampien ohjelmointialustojen soveltuvuutta. Kali Linux ei sopinut tarkoitukseen, mutta Windows-ympäristön tehokas ohjelmointialusta Visual Studio osoittautui toimivaksi.

Prototyyppiprojektiin nimeltä Koodinsiivous on kehitetty koodi, joka poistaa merkkijonotaulukosta tietyn sanan (Liite 1 Kuva 4). Tämä koodi vaatii ohjelmointikielen C# toistotoimintoja ja funktiota Replace (Kumar 2018). Sana on tämän funktion avulla poistettavissa korvaamalla poistettava sana tyhjällä merkkijonolla.

Visual Studion avulla kykenee avaamaan tiedoston html-formaatissa niin, että sisälle voi sijoittaa koodia. Työn tavoitteiden mukaan suunniteltavan paketinsuodatusohjelman on kuitenkin sovelluttava usean erilaisen raportin tarpeisiin. Tähän tarkoitukseen paras vaihtoehto on antaa ensin koodin lukea koko tiedosto ja vasta sitten käsitellä sitä.

Visual Studio tukee Windows-sovelluksia käsitteleviä ominaisuuksia, joka ilmenee kirjastona nimeltä Windows.Forms. Visual Studio 2019 -alustassa se vaati seuraavien ympäristöjen asennusta: .NET, ASP.NET (ASP tulee sanoista Active Server Pages), Azure ja Node.js (Node.JavaScript) (Young, R. 2019). Tämä on toteutettavissa myös komentokehotteen kautta.

Visual Studio ei testin aikana tukenut Windows.Forms-kirjaston lukufunktioita. Sen sijaan se tuki dynaamisesti linkitettyjä kirjastoja, jotka ovat Microsoftin versio jaetuista kirjastoista Windows-ympäristölle. Jaettu kirjasto on viittauk-

sen sisältävä kirjasto niin, että muut ohjelmat pääsevät käyttämään sitä sen sijaan, että kirjasto olisi osana yksittäistä ohjelmaa (Linux.fi-wiki s.a., What is a DLL? s.a.). Dynaaminen tarkoittaa tässä tapauksessa sitä, että kirjasto ei ole muistissa aina saatavilla, vaan kirjaston ominaisuus tulee käyttöön vain sitä erikseen kutsuttaessa. (EXE Files s.a.) Dynaamisesti linkitettyä kirjastoa on kätevämpää kutsua lyhenteellä DLL eli Dynamic Link Library.

Seuraavassa esimerkissä koodi käyttää Word-luokkaa:

```
public static string GetTextFromWordDocument(object filePath)
    {
        var filePathAsString = filePath as string;
        if (string.IsNullOrEmpty(filePathAsString))
        {
            throw new ArgumentNullException("filePath");
        }

        if (!File.Exists(filePathAsString))
        {
             throw new FileNotFoundException("Could not find file",
             filePathAsString);
        }

        var textFromWordDocument = string.Empty;
        Word.Application wordApp = new Word.Application();
        Word.Document wordDocument = null;
        Word.Range wordContentRange = null;

        try
        {
            wordDocument = wordApp.Documents.Open(ref filePath, Missing.Value,
            true);
            wordContentRange = wordDocument.Content;
            textFromWordDocument = wordContentRange.Text;
        }
        catch
        {
            // handle the COM exception
        }
        finally
        {
            if (wordDocument != null) { wordDocument.Close(false); }
            if (wordApp != null) { wordApp.Quit(false); }

            ReleaseComObject(wordDocument);
            ReleaseComObject(wordContentRange);
            ReleaseComObject(wordDocument);
            ReleaseComObject(wordApp);
        }

        return textFromWordDocument;
    }
```

Esimerkkikoodin alussa luodaan poikkeustilanteet siltä varalta, jos tiedoston nimeä tai itse tiedostoa ei löydy. Koodissa mainittava WordApp on liitännäinen, joka mahdollistaa esimerkiksi verkkosivun muokkaamisen (WordApp

s.a.). Koodin keskiosassa yritetään avata tiedostosijainti. Metodilla Release-ComObject varataan muistia Word-asiakirjaa varten (Marshal.ReleaseCom... s.a.).

On olemassa yksi mahdollinen keino yhdistää koodi Microsoft Wordiin Visual Studion kautta: Visual Studio Tools for Office (mentinet 2016). Microsoft Office -tuotteisiin liittyvä kirjasto on sitä kautta haettavissa ohjelmointialustalta Visual Studio. Sitä ennen kirjaston materiaali vaatii vähintään toisen version kirjastosta nimeltä .NET Framework ja tuen nimeltä .NET Programmibility Support (Microsoft 2019).

Word-tiedoston piti tulostaa lyhyttä tekstiä näytölle, jos polun nimi sijoitettiin funktioon GetTextFromWordDocument. Tämän funktion oli tarkoitus palauttaa Word-sovelluksen sisältämä teksti. Tässä työssä se vaatii C-levyn sijainnin, joten testiasiakirja oli myös siellä. Asiakirjan tekstiä yritettiin tulostaa komentokehotteen kautta ja myös avata asiakirja. Seuraavan komennon oli tarkoitus tulostaa komentokehotteen kautta asiakirjan tekstiä ja avata asiakirja:

```
this.Application.Documents.Open(@"C:\Test\NewDocument.docx");
```

(Microsoft 2017.)

Kummatkaan keinot eivät kelvanneet, koska määritettyä tiedostoa ei löytynyt. Käyttöoikeuksissa oli myös ongelmia. Lisäksi hakutoiminnon apu rajoittui lähinnä Windows 10 -järjestelmän ja Wordin makrojen turvallisuuden parantamisehdotuksiin.

Wordia ei välttämättä tarvitse käyttää Visual Studio -alustaan soveltamiseen, vaan muutkin tiedostomuodot käyvät. Seuraava vaihtoehto oli lisätä PDF eli Portable Document Format -tiedosto tähän alustaan, jota varten on olemassa seuraava koodi:

```
string execPath = GetAssociation(".pdf");
using System.Diagnostics;
...
// Start new process
Process.Start(execPath, "C:\\myfile.pdf").WaitForExit(0);
```

(user3738170 2015.)

Ohjelma ei kuitenkaan löytänyt esimerkkikoodin yläpuolella näkyvää funktiota GetAssociation.

Testaamiseen kuuluu Visual Basic Editor -ominaisuuden kokeilu. Testikoodina toimii Hello World (Keith 2015):

```
Sub Hello_world()
   MsgBox ("Hello world")
End Sub
```

Huolimatta yrityksestä lukea tiedosto sovelluksen Visual Basic Application avulla sovellus ei kuitenkaan hyväksynyt tiettyjä funktioita.

Microsoft.Office.Interop.Excel-kirjasto toimii lisäämällä referenssiksi Microsoft Excel 16.0 Object Library -kirjaston toiminnolla Add Reference.

PDF-asiakirjan kokeilussa polun saamiseksi ei ole kannattavaa ladata kyseistä asiakirjaa Visual Studio -alustan sisältä. Sen sijaan kannattaa hyödyntää PDF-sovelluksia, joista OpenSource PDF on yksi esimerkki. Haluttuja tiedostoja ei kuitenkaan löytynyt yrityksessä ladata kirjasto nimeltä IText 7 ja siirtää se Visual Studio -alustaan.

C#-kielellä toimivaa Iron PDF -kirjastoa kokeiltiin tähän työhön. Iron PDF -pääsivun kautta kykenee lataamaan DLL-tiedostot ja lisäämään ne referensseiksi Visual Studio -projektiin. Seuraava koodinpätkä kuitenkin epäonnistuu tässä työssä muuttaa HTML-teksti PDF-tekstiksi:

```
using IronPdf;
IronPdf.HtmlToPdf Renderer = new IronPdf.HtmlToPdf();
Renderer.RenderHtmlAsPdf("Hello World").SaveAs("html-string.pdf");
```

Tuloksena tulee **System.TypeLoadException:** 'Could not load type 'System.Web.HttpRequest' from assembly 'System.Web, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a'.'

Erään lähteen mukaan tämä tarkoitti sitä, että verkkosivu on avoinna (john-john123123 2014), mutta verkkosivuja oli jo valmiiksi auki, eikä koodissa ole

näkyvillä polkuparametria, jonne voisi sijoittaa URL-osoitteen eli Uniform Resource Locator -osoitteen.

Todennäköisesti tarkoitus oli luoda itse HTML-sivu ja sitten kokeilla. Exception-viesti seuraa, jos HTML-sivun yrittää muuttaa PDF-formaattiin seuraavalla koodilla:

```
var Renderer = new IronPdf.HtmlToPdf();

Renderer.RenderHtmlAsPdf("<h1>Html with CSS and Images</h1>").SaveAs("exam-
ple.pdf");
```

Tämä koodi ei välttämättä hyväksy heikompaa merkkijonoa, eli merkkijonoa ilman erikoismerkkejä. Vahvempaan tarvittaisiin sulkeissa oleva merkkijono, joka hyväksyisi myös erikoismerkit. Seuraavassa listassa näkyy eri merkkijonojen muotoja C#-ohjesivulta (Strings (C# Programming… s.a.):

```
var Renderer = new IronPdf.HtmlToPdf();
string filepath = "<h1>Html with CSS and Images</h1>";
string filepath_2 = String.Format("<h1>Html with CSS and Images</h1>");
string filepath_3 = @"<h1>Html with CSS and Images</h1>";
System.String filepath_4 = @"<h1>Html with CSS and Images</h1>";
var filepath_5 = "<h1>Html with CSS and Images</h1>";
const string filepath_6 = "<h1>Html with CSS and Images</h1>";
string filepath_7 = new string("<h1>Html with CSS and Images</h1>");
```

Vaikka viimeisimmän rivin rakenne viittaa vahvempaan merkkijonoon, Exception-viesti seuraa silti. Ohjelma ei myöskään löydä viittausta System.Web-DLL-tiedostoon (Mamdoohi 2014), vaikka vaihtaisi .NET-standardia.

Testaamiseen kuului Python-ympäristön käyttö. Ympäristöön kuuluvien työkalujen asennusten yhteydessä yksinkertainen Hello world -koodi on mahdollista luoda ja suorittaa. Python-kielellä voi ohjelmoida samankaltaisen prototyypin ongelmanratkaisusta kuin C#-kielellä:

```
greeting = "Hello world"

answer = input("Which word you would like to remove from 'Hello world'? ")

if (answer in greeting):
    greeting = greeting.replace(answer, "")
print(greeting)
```

Työn kannalta haastavaksi osoittautunut tiedostopolun löytäminen vaati lisäselvitystä. Visual Studio -ohjelmointialustan lisäksi testattaviin alustoihin kuuluivat ohjelmointialustat NetBeans ja Eclipse sillä oletuksella, että niiden avulla ohjelma osaa löytää paremmin tiedostopolun.

NetBeans ja Eclipse asentuivat onnistuneesti (TubeMint 2019), mutta edellinen vaati lisäksi Java- ja Python-kieliin liittyviä liitännäisiä, jotta näillä kielillä koodaaminen olisi ollut siinä mahdollista. NetBeans-alustaa pystyi testaamaan kirjoittamalla Hello World -teksti jollakin käytetyllä koodilla.

NetBeans-alustalla pystyy onnistuneesti tulostamaan tekstin ”Hello world”, kunhan versio on ainakin 11.2 ja jota varten on ladattavissa JDK 13 (Alberto 2019). Python-liitännäiset ovat myös ladattavissa kyseiseen alustaan (Umer Softwares 2019). Vaikka kokeiltavana Python-ympäristönä toimii tässä alustassa JPython 3.6, ohjelma ei tunnistanut asennettuja liitännäisiä. Jython-kielellä Hello World -perustekstin tulostus toimii muutamasta varoituksesta huolimatta (Liite 1 Kuva 5).

Videopalvelusta nimeltä YouTube löytyy video, jossa on ohjeet yksinkertaisen tiedoston lukemiseen (thenewboston 2009). Ideana oli testata NetBeans-alustalla toimivan ohjelman lukukykyä näiden ohjeiden avulla. Ohjelma näytti kuitenkin tuntemattomasta syystä NoSuchElementException-viestiä.

NetBeans-alustan lisäksi kokeiltavana alustana toimi Eclipse, jonka asentaminen oli suhteellisen yksinkertaista. Kuitenkin PATH-ympäristömuuttujan muokkaamisesta ja alustan uudelleenkäynnistyksestä huolimatta sen toiminta epäonnistui (Sinner 2013).

Kun koodi käynnistyi NetBeans-alustalla, ohjelma ei todennäköisesti tunnistanut pyydettyä syötettä siinä missä Visual Studio -alustalla se pystyi (Liite 1 Kuva 6). Saattoi olla, että tähän vaikutti useammat käytössä olleet versiot Python-ympäristöstä asennettuna yhtä aikaa, joten kaikki muut paitsi versio 3.6 täytyi poistaa.

Ylimääräisten versioiden poisto ei kuitenkaan ratkaissut ongelmaa. Muu mahdollinen toimenpide oli PDF-tiedoston luominen. Tiettyjen luokkien tuominen

alustalle vaati tässä työssä PDFSharp-kirjaston kokeilua. Huomioitavaa oli, että kirjaston dokumentaatiossa oli kaksi luokkaa: Pdf ja Drawing, joissa oli vaadittavat funktiot. Lisäksi tiedostopolku, joka oli pääongelma, koostui /-merkeistä eikä \-merkeistä, niin kuin resurssienhallinnassa tyypillisesti oli. Näiden tietojen avulla PDF-tiedoston pystyy onnistuneesti luomaan ohjelmoimalla, kuten kuva 7 (Liite 1) osoittaa. Kuvan 8 (Liite 1) koodi taas muuttaa HTML-tiedoston onnistuneesti PDF-tiedostoksi.

PDF-tiedostosta tekstiformaattiin -vaiheen oli tarkoitus seurata HTML-tiedostosta PDF-tiedostoon -vaihetta C#-kielen avulla. Ohjelma ei löytänyt iDiTect-kirjaston (IdiTect s.a.) avulla tiedostopolkua, mutta RasterEdge-kirjaston (Raster Edge s.a.) avulla koodi toimi onnistuneesti niin, että se kykeni luomaan PDF-tiedostosta tekstitiedoston siitäkin varoituksesta huolimatta, että työkannettavan prosessorin arkkitehtuuri ei täsmännyt.

Toisin sanoen, varsinainen koodi kykeni luomaan HTML-tiedostosta tavallisen tekstitiedoston muuttamalla sen ensin PDF-muotoon. Koodi toimii seuraavasti:

```
using System;
using System.IO;
using System.Drawing;
using System.Collections.Generic;
using RasterEdge.XDoc.PDF;

static void Main(string[] args)
{

    try
    {
        // Create a PDF from an existing HTML using C#
        var Renderer = new IronPdf.HtmlToPdf();
        var PDF = Renderer.RenderHTMLFileAsPdf("C:/Users/Markus Frosti/Docu-
        ments/Testihtmltiedosto.html");
        var OutputPath = "C:/Users/Markus Frosti/Documents/Testihtmltiedo-
        sto.pdf";
        PDF.SaveAs(OutputPath);

        String inputFilePath = @"C:/Users/Markus Frosti/Documents/Testiht-
        mltiedosto.pdf";
        String outputFilePath = @"C:/Users/Markus Frosti/Documents/Testiht-
        mltiedosto.txt";
        StreamWriter writer = new StreamWriter(outputFilePath);
        PDFDocument doc = new PDFDocument(inputFilePath);
        PDFTextMgr textMgr = PDFTextHandler.ExportPDFTextManager(doc);
        int pageCount = doc.GetPageCount();
        for (int i = 0; i < pageCount; i++)
        {
            PDFPage page = (PDFPage)doc.GetPage(i);
            List<PDFTextLine> pageTextLines = textMgr.ExtractTextLine(page);
            writeTextLines(pageTextLines, writer);
        }
```

```
        writer.Close();
    }
    catch (IOException e)
    {
        Console.WriteLine("The file could not be read:");
        Console.WriteLine(e.Message);
    }
```

Tarkoituksena oli luoda ohjelma, joka kysyisi käyttäjältä, mitkä sanat poistettaisiin raportista. Tiedoston sisältöä kykeni tuolloin muokkaamaan vain tekstitiedostona ja ajatus siitä, voiko jonkun sanan perusteella poistaa koko rivin, jäi tässä vaiheessa vähemmän tärkeäksi. Seuraavaksi oli vuorossa tekstitiedoston muuttaminen takaisin HTML-muotoon. Siihen riitti esimerkkikoodin inputFilePath- ja outputFilePath-muuttujien arvojen vaihtaminen keskenään.

Valitettavasti kävi ilmi, että raportista poistuivat sen visuaaliset ominaisuudet fontteja myöten niin, että raportti näkyi vain tekstitiedostona. Toisin sanoen tekstitiedoston hyödyntäminen jouduttiin sivuuttamaan.

Tässä työssä oli tarpeen luoda uusia tilejä erilaisissa koodauspalstoissa oikeita kirjastoja varten. Spire.PDF-kirjaston ilmaisversioon soveltuva koodi (Aggarwal 2018) oli kokeilun alla, mutta se ei kyennyt poistamaan erikoisfontilla merkittyä sanaa ja muutenkin se kykeni vain muokkaamaan sanoja taustanväriseksi niin, että ne vain näyttivät poistetulta. Ideaalisesti sanojen tulisi poistua kokonaan PDF-tiedostosta, mielellään niin, että koko ei-toivotun sanan rivi poistuisi.

VintaSoft-kirjasto ja sen esimerkkikoodi, jossa tekstin sisältöä pystyy kuvauksen perusteella poistamaan (VintaSoft s.a.), olivat myös yrityksen kohteena, mutta kirjaston DLL-tiedostot eivät käyneet viittauksiksi.

Aspose-kirjasto oli lupaava vaihtoehto huomioiden sen esimerkkikoodin toimintaesimerkin (PDF redaction s.a.; Replace Text in a PDF Document s.a.). Demonstraatio-ohjelmalle voi antaa PDF-raportin ja pyytää sitä muokkaamaan raportin ei-toivotut sanat johonkin muuhun sanaan. Ajatuksena oli, että ohjelman toimintaa havainnollistava koodi toimisi yhtä hyvin kuin itse ohjelma. Valitettavasti tuli IndexOutOfRangeException-virheilmoitus (Liite 1 Kuva 9), johtuen todennäköisesti käytetyn version rajoittuvuudesta.

ByteScout-kirjaston koodi poistaa halutun tekstin PDF-tiedostosta sen kuvauksen mukaan (Liite 1 Kuva 10). Se ei kuitenkaan tehnyt mitään.

iText-kirjaston yrittäminen oli harkinnassa, mutta palstoilla selvennettiin, ettei PDF-tiedoston muokkaus sellaisenaan yleisesti ottaen ole mahdollista, sillä PDF-dokumentti on oletusarvoisesti renderöity eikä rakenteellinen. (IText 2020, Trong 2019.)

Vaihtoehdoksi jäi siirtyminen PDF-tiedoston muokkauksesta sen muuttamiseen doc- tai docx-formaattiin ja toisinpäin. Olennaisimmista seikoista oli siis jäljellä PDF-tiedoston muuttaminen Word-tiedostoksi, Word-tiedoston sisällön muokkaaminen ja Word-tiedoston muuttaminen HTML-tiedostoksi vaikka PDF-tiedoston kautta.

Spire.PDF toimi onnistuneesti. Siinä oli se ajatus, että se pystyisi muuttamaan PDF-tiedoston Word-tiedostoksi (E-Iceblue s.a.). Tämän lisäksi oli löydettävä, kuinka muokata jälkimmäisen sisältöä C#-kielen avulla. SautinSoft-kirjaston koodi oli lupaava ja sopivan yksinkertainen (SautinSoft s.a.). Ohjelma tulosti kuitenkin, ettei tiedostoa voitu lukea, sillä se sisälsi vioittuneita tietoja. Todennäköisesti formaatti ei kuitenkaan kelvannut (Ramesh 2014).

Suunnitelmissa oli luoda prosessin yksinkertaistamiseksi oma funktio HTML-tiedoston muuttamisesta Word-tiedostoon niin, että se poistaisi myös PDF-tiedoston jälkeenpäin turhana sivutuotoksena. Tässä funktiossa käytetyt kirjastot olivat System.IO, Spire.Pdf ja IronPdf (Liite 1 Kuva 11).

Spire.Doc vaikutti oikeasti kykenevän muokkaamaan Word-tiedoston sisältöä – tosin epävakaasti. Esimerkiksi fontit saattoivat muuttua muokattavissa kohdissa, mutta se ei juuri haitannut, sillä tässä työssä oli kyse tarpeettoman tiedon poistamisesta. Pääongelmana oli se, että prosessi tiivisti koko asiakirjan sisällön yhdelle sivulle niin, että sanat saattoivat peittyä toisiinsa. (Liite 1 Kuva 12; kdr13 2017.)

Lisähuomiona vielä, että Spire.Doc kykeni muuttamaan HTML-tiedoston suoraan Word-tiedostoksi ja toisinpäin ilman, että sitä tarvitsi muuttaa PDF-tiedos-

toksi. Kuitenkin prosessin yhteydessä raportista hävisivät visuaaliset ominaisuudet, jotka tekivät raportista yksilöllisen. Raportin olisi hyvä säilyä koskemattomana tarpeellisia muutoksia lukuun ottamatta. Spire.Doc-kirjaston toiminta jäi siis pois laskuista tämän työn kannalta hyvästä ajatuksesta huolimatta.

Word-tiedoston muuttaminen ehjänä HTML-tiedostoon oli todennäköisesti jotenkin mahdollista, mutta sen lisäksi oli selvitettävä, kuinka estää Word-asiakirjan sivujen yhdistyminen sitä muokattaessa. Erilaisia kokeilemisvaihtoehtoja olivat esimerkiksi erottaa asiakirjan sivut erillisiin tiedostoihin ja muokata sitten niitä yksitellen, mutta yhdessäkään niistä ohjelma ei kyennyt löytämään tiedostopolkua. Yksi vaihtoehto oli myös etsiä ja kokeilla erilaisia koodeja esimerkiksi TechRepublic-yhteisön sivulta (2008), mutta ohjelma ei löytänyt niidenkään kautta tiedostopolkua.

SautinSoft-kirjastosta löytyi keino muuttaa PDF-tiedosto DOCX-formaattiin, jota voi ajatella myös Word-tiedostoksi muuttamisena. Aikaisemmin luotu funktio kykeni kyllä muuttamaan HTML-tiedoston Word-tiedostoksi, mutta vain vanhemmaksi versioksi eikä uudemmaksi. Toisin sanoen se tallensi Word-asiakirjan muodossa doc eikä muodossa docx. Jälkimmäisellä formaatilla avaaminen oli kuitenkin mahdollista ja kun tällaista formaattia käytettiin SautinSoft-kirjaston koodissa, Word-raportin haluttu teksti poistui onnistuneesti.

Aspose ei tässä työssä voinut muokata tekstiä, mutta se kykeni muuttamaan HTML-tiedoston PDF-tiedostoksi, jossa raportti vaikutti suhteellisen eheältä. Se oli toistaiseksi tyydyttävin tulos. Viimeinen osa suunniteltua prosessia oli Word-tiedoston palauttaminen takaisin HTML-muotoon. Spire.Doc-kirjastoa kokeilemalla raportin eheys petti (Liite 1 Kuva 13).

Projektin ehtoja noudattava ratkaisu löytyi oletuksena HTMLAgilityPack-kirjastosta, jonka avulla kykeni muokkaamaan HTML-formaatissa olevia tiedostoja suoraan. Kirjaston kautta on mahdollista poistaa tietty HTML-solmu kokonaan ja tallentaa muokattu tiedosto. Improvisoinnin vuoksi ohjelmaan liitettiin myös kysely.

Kuitenkin projektissa toivottiin mielellään raportin muokkaamista WireSharkille ominaisessa formaatissa. Projekti vaatii myös käänteistoiminnon, eli ohjelma säilyttäisi ainoastaan ne osat raportista, jotka sisältävät tietyn sanan poistaen kaiken muun.

Suositeltavissa olevana vaihtoehtona oli JSON-tiedostoja (Lamping & Warnicke 2004, 5.7.2) käsittelevä kirjasto NewtonSoft. Oletuksena oli, että sen avulla näistä tiedostoista poistuisi paketit käyttäjän antaman syötteen mukaan. Syötteen rakenne on samanlainen kuin Linux-käyttöjärjestelmän komentokehotteessa.

6 TÄRKEIMMÄT TULOKSET

Lopputuloksien hankinnassa kesti oletettua kauemmin, osittain siksi, että yrityksellläni oli kiireitä ja muita työtehtäviä sekä osittain siksi, että oikeiden ratkaisujen etsinnässä oli monia vaihtoehtoja valittavana. Lopulta löysin oikeat ratkaisut niin, että ohjelmaani voi myös parannella myöhemmin.

6.1 Vastaukset tutkimuskysymyksiin

Tutkimukset osoittivat, että on mahdollista ohjelmoida koodi, joka etsii JSON-tiedostosta paketin annettujen kriteerien mukaisesti noudattaen Linux-käyttöjärjestelmän komentorakennetta. Koodiin pystyi myös sijoittamaan käänteistoiminnon, joka etsii eksklusiivisesti eri kriteereillä halutut paketit, esimerkiksi IP-osoitteen ja porttinumeron perusteella. Toisin sanoen, onnistuneesti toteutettu koodi vastaa myönteisesti ensimmäiseen tutkimuskysymykseen, eli pystyykö verkkoanalysoijien raporteista suodattamaan pois turhaa tekstiä tekoälyn avulla.

Toinen kysymys oli se, mitä tekoälyllä tarkoitetaan. Tämä kysymys kuuluu enemmän teorian kuin käytännön alueelle, mutta koodin rakenne pystyi demonstroimaan kapean ja heikon tekoälyn toimintaa. Se pystyi simuloimaan etsintävaistoa tiettyä tarkoitusta varten joko etsittävän kohteen poimimisella tai poistamisella.

6.2 Koodin rakenne

Lopullisen koodin oli tarkoitus poistaa JSON-raportista paketit tiettyjen kriteerien perusteella. Siinä oli myös eksklusiivinen käänteistoiminto, joka toimi niin, että etsittiin paketteja, jotka sisälsivät joitakin annetuista kriteereistä, ja sitten poistettiin muut paketit. Koodissa käytettiin NewtonSoftin ohjelmointikirjastoa. Kaavio koodista näkyy kuvassa 14 (Liite 1).

Raportista tallennettiin kaksi jäsennettyä versiota, joista toista käytettiin normaalissa ja toista käänteisversiossa. Kun ohjelman käyttäjä kirjoittaa syötteen, syöte jaetaan eri sanoihin ja siten ohjelma varmistaa, että syötteessä oli joko jotain tekstiä tai että se alkoi jollakin seuraavista parametreista: framelen, ipsrc, ipdst, ethsrc, ethdst, tcpport, udpport, any. Syötteen lopussa oleva -r-merkintä oli käänteistoiminnon merkki. Kun haluttu syöte annetaan, ohjelma käy läpi jokaisen syötteen sanan ja kun parametri tulee vastaan, ohjelma kerää parametrin jälkeiset arvot kuten IP-osoitteet ja toteuttaisi raportinmuokkausfunktion. Ohjelman muokkausfunktio käy läpi jokaisen lohkon JSON-tiedostosta ja varmistaa, löytyykö parametriin yhdistettyä arvoa. Jos arvo löytyy ja jos käänteistoiminto ei ole voimassa, tämä lohko poistetaan raportista. Käänteistoiminnossa käytetään kahta samaa tallennettua ja jäsennettyä raporttia. Toisesta poistetaan koko sisältö ja toista käytetään parametreihin yhdistettyjen arvojen etsimiseen. Kun arvo on löydetty, se lisätään tyhjennetylle raportille, jolloin saadaan aikaan vaikutelma arvojen mukaisten pakettien säästämisestä. Raportin muokkaus päättyy, kun käyttäjän syötteen sanat on käyty kokonaan läpi. Muokkauksen jälkeen tapahtuu tallentaminen päätteellä ._modified.json.

6.3 Vertaukset aikaisempaan materiaaliin

Tämä työ havainnollisti loogista etsintäominaisuutta aikaisemmin esitetyn ristinollamenetelmän lisäksi. Myös NewtonSoft-kirjaston aikaisemmista hakufunktioista pystyi rakentamaan uudenlaisen hakuohjelman, mikä sopii kehitysmenetelmäksi.

Työn aikana syntynyt lopputulos päätyi yksinkertaisemmaksi ja rajoitteisemmaksi versioksi WireShark-ohjelman suodatusmenetelmästä (Sanders 2018).

Siihen ei ollut aikaa kehittää suurempia tekoälyominaisuuksia, mutta työssä esitetyn materiaalin perusteella se on mahdollista.

Työn tulokset eivät välttämättä olleet ristiriidassa seuraavien TensorFlow-ominaisuuksien kanssa: UniCode, Record-kirjaston lukufunktio ja LSTM-kerrokset sekä toiminnan lisääminen MIMO-tekniikan avulla. TensorFlow-sovelluksella voi yrittää luoda unicode_split-funktion avulla näytön, joka ilmoittaa, mitä parametrejä käytetään paraikaa. Record-kirjaston lukufunktiolla voi mahdollisesti luoda varoitusviestin. LSTM-kerrokset parantavat tekoälymenetelmän suorituskykyä. Ja toiminnallisen mallin MIMO-tekniikkaa käyttämällä jos useampi kuin yksi arvo saadaan kerralla mallin sisään, se nopeuttaa ohjelman toimintaa.

Jollakin TensorFlow-sovelluksen kaltaisella tekoälysovelluksella voisi antaa luotuina ja opetettavina arvoina parametrit ja niiden arvot. Laite päättelisi sitten raportin sisällön perusteella, mitä useista parametreista voisi käyttää opetettavina arvoina. Tuloksena syntyisi verkkoliikenteen mukaan raporttia muokkaama tekoälymenetelmä.

6.4 Oman oppimisen pohdinta

Työni toimivuus on luotettavaa, sillä omien testien lisäksi Innocode-yrityksen henkilökunta testasi koodini toimintaa. Omaan testiraporttiin kuului seitsemän pakettia ja henkilökunta suoritti testin paljon suuremmalla pakettimäärällä. Tosin henkilökunnalla oli kiireinen aikataulu ja aikaa kului tehokkaan testiympäristön valmistelussa. Lisäksi koska suoritin työni valtaosin etätyönä, aikaa kului sopivien ajanhetkien löytämiselle työni arvioimiseksi.

Odotuksella kuitenkin on hyvät puolensa työn kannalta. Esimerkiksi työntekijä voi olla vapaampi kehittämään erilaisia teorioita ja suunnitelmia varsinaista työhetkeä varten. Lisäksi hän voi yrittää tehdä keskeytyneitä töitä.

Kun keskittyy kunnolla tekoälyn luonteen pohtimiseen, voi oppia monia asioita tekoälyyn liittyen. Esimerkiksi hallitsen aikaisempaa paremmin tekoälyyn miellettävät ominaisuudet, erilaisia demonstraatioita tekoälystä ja Bayesin kaavan

merkityksen. TensorFlow-sovelluksen perusteet näyttivät myös tekoälymenetelmän rakenteen ja että mikä on tärkeää muistaa sitä suunnitellessa, kuten ylisovituksen välttäminen. Näin olennaiset aiheet tekoälyssä tarjoavat monipuolisia mahdollisuuksia innovaatiolle ja on innostavaa yrittää kehittää niistä jotain uutta.

Opinnäytetyötä luodessa Visual Studio -alustan käyttö oli tarpeellista, mikä johti väistämättä siihen, että tajusin kunnolla kirjastojen tarpeellisuuden ohjelmoinnissa ja että kuinka hyödyntää kirjastoja erilaisia ohjelmointivaihtoehtoja etsiessä. Oikeaa ratkaisua etsiessä opin myös, kuinka muokata ohjelmoinnin kautta perustekstitiedostoa, PDF-tiedostoa ja HTML-tiedostoa. Ennen kaikkea työ opetti, kuinka hyödyntää erilaisia ohjelmointikirjastoja, erityisesti Newton-Soft-kirjastoa, ja kuinka muokata JSON-tiedostoja. Toisin sanoen, jos jatkossa työtehtävänä on vaikka muokata koodin avulla tekstitiedostoa, minulta kuluu siinä vähemmän aikaa kuin alussa. On todennäköistä, että tällainen informaatio tulee hyödyksi jatkossakin. Kuitenkin työn aikana paljastui myös, että sille on rajansa, kuinka suoraan ja helposti joitakin ratkaisuja saa esille sopivia kirjastoja etsimällä. Voi olla, että joskus joutuu soveltamaan useampia kirjastoja.

6.5 Jatkokehitysideoita

Yrityksen sisäisten tilanteiden vuoksi aika ei riittänyt täysin jatkokehitysten toteuttamiseen. Ideoita voisi kuitenkin olla mahdollista kehittää hankittua tietoa käyttämällä. Nämä ehdotukset ovat tällä hetkellä hypoteettisia. Ohjelmalle voisi yrittää keksiä inklusiivisen etsintätoiminnon, eli ohjelma etsii vain ne paketit, joissa on kaikki annetut arvot. Tällä tavalla ohjelma saadaan entistä tarkemmaksi ja monipuolisemmaksi. Ohjelmaa voisi yrittää myös automatisoida, esimerkiksi niin, että ohjelma poistaa raportteja viiden minuutin välein. Poikkeuksellisten pakettien automaattinen poisto lisää ohjelman eheyttä, mutta liian tiheä poisto lisää tarpeettomasti ohjelman kuormitusta. Joitakin näistä varten luotavia virheilmoituksia varten try/catch-ominaisuus voisi tulla hyödylliseksi. Se on poikkeustilanteiden nappaamista varten tarkoitettu toiminto ohjelmoinnissa. Kumoamistoiminnolla pystyisi peruuttamaan edellisen toiminnon, jos se paljastuu vasta jälkikäteen ei-toivotuksi. Tämä ominaisuus on tullut tutuksi erityisesti Microsoft-tuotteista ja se on loistava keino satunnaisia virheitä varten. Lopuksi jos ohjelmaa kehitetään tarpeeksi adaptiiviseksi, muitakin kuin

vain JSON-tiedostoja voisi soveltaa ohjelmaan. Näitä kehitysideoita hyödyntämällä saisi koodista tekoälyä enemmän noudattavan kokonaisuuden, ja vaikkei niitä kaikkia pystyisikään toteuttamaan, työ tarjoaa lisää tietoa aiheesta.

LÄHTEET

Aggarwal, K. 2018. Edit PDF Text using C#. StackOverflow-viesti. Saatavissa: https://stackoverflow.com/questions/50365323/edit-pdf-text-using-c-sharp [viitattu 17.3.2020].

Alberto. 2019. Run a Java application with JDK 13 on NetBeans. StackOverflow-viesti. Saatavissa: https://stackoverflow.com/questions/58623701/run-a-java-application-with-jdk-13-on-netbeans [viitattu 8.3.2020].

awav. 2019. TensorFlow nightly build for Python 3.7. GitHub-viesti. Saatavissa: https://github.com/tensorflow/tensorflow/issues/32718 [viitattu 14.8.2020].

BetterExplained. s.a. WWW-dokumentti. Saatavissa: https://betterexplained.com/articles/understanding-bayes-theorem-with-ratios/ [viitattu 25.2.2020].

ByteScout. s.a. ByteScout PDF Extractor SDK – C# – Remove Text. WWW-dokumentti. Viitattu 18.3.2020. Saatavissa: https://bytescout.com/articles/pdf-extractor-sdk-c-remove-text [viitattu 18.3.2020].

DeepAI. s.a. Weight (Artificial Neural Network). WWW-dokumentti. Saatavissa: https://deepai.org/machine-learning-glossary-and-terms/weight-artificial-neural-network [viitattu 5.4.2020].

Deshpande, A. & Kumar, M. 2018. Artificial Intelligence for Big Data: Complete guide to automating Big Data solutions using Artificial Intelligence techniques. Birmingham: Packt Publishing Ltd.

E-Iceblue. s.a. How to convert PDF to Doc in C#, VB.NET. WWW-dokumentti. Saatavissa: https://www.e-iceblue.com/Tutorials/Spire.PDF/Spire.PDF-Program-Guide/Conversion/How-to-convert-PDF-to-Doc-in-C-VB.NET.html [viitattu 18.3.2020].

EXE Files. s.a. Mitä ovat Dynaamisesti linkitetty kirjasto -tiedostot? WWW-dokumentti. Saatavissa: https://www.exefiles.com/fi/extensions/file-types/dynamic-link-library/ [viitattu 2.4.2020].

Garbade, M. 2018. Understanding K-means Clustering in Machine Learning. Medium-julkaisu. Saatavissa: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1 [viitattu 17.2.2020].

Goldkuhl, G. Action Research vs. Design Research: Using Practice Research As a Lens For Comparison and Integration. 2013. Opinnäytetyö?. Saatavissa: https://www.researchgate.net/publication/283316882_ACTION_RESEARCH_VS_DESIGN_RESEARCH_USING_PRACTICE_RESEARCH_AS_A_LENS_FOR_COMPARISON_AND_INTEGRATION [viitattu 6.8.2020].

Grandi passi in avanti… 2020. Boston Dynamics. Videoleike. Saatavissa: https://www.facebook.com/watch/?v=206009477262537 [viitattu 8.8.2020].

Gulli, A. & Pal, S. 2017. Deep Learning with Keras. Packt Publishing Ltd. [viitattu 5.8.2020].

Halonen, K. 2019. Tekoälyn hyödyntäminen urheiluvammojen ennaltaehkäisyssä. Kandidaatintyö. Saatavissa: https://lutpub.lut.fi/bitstream/handle/10024/160046/Kandidaatinty%C3%B6_%20Halonen.pdf?sequence=1&isAllowed=y [viitattu 5.4.2020].

IdiTect. s.a. How to Convert PDF to Text in C#.NET Code. WWW-dokumentti. Saatavissa: https://www.iditect.com/tutorial/pdf-to-text/ [viitattu 15.3.2020].

Iron PDF. s.a. HTML Files to PDF. WWW-dokumentti. Saatavissa: https://ironpdf.com/#file-to-pdf [viitattu 13.3.2020].

IText. 2020. How to remove text from a PDF? WWW-dokumentti. Saatavissa: https://itextpdf.com/en/resources/faq/technical-support/itext-7/how-remove-text-pdf [viitattu 18.3.2020].

johnjohn123123. 2014. Could not load file or assembly 'System.Web, Version=4.0.30319.17929, Culture=neutral, PublicKeyToken=b77a5c561934e089' or one of its dependencies. The system cannot find the file specified. Microsoft-viesti. Saatavissa: https://forums.asp.net/t/1962893.aspx?Could+not+load+file+or+assembly+System+Web+Version+4+0+30319+17929+Culture+neutral+PublicKeyToken+b77a5c561934e089+or+one+of+its+dependencies+The+system+cannot+find+the+file+specified+ [viitattu 20.2.2020].

Kananen, J. 2017. Kehittämistutkimus interventiotutkimuksen muotona. Opas opinnäytetyön ja pro gradun kirjoittajalle. Jyväskylän ammattikorkeakoulun julkaisuja. 60-61.

kdr13. 2017. Save and close file after text replace/remove. E-IceBlue-viesti. Saatavissa: https://www.e-iceblue.com/forum/save-and-close-file-after-text-replace-remove-t7041.html [viitattu 19.3.2020].

Keith, M. 2015. VBA basics - Hello World. YouTube. Videoleike. Saatavissa: https://www.youtube.com/watch?v=iCZMHrIrtOI [viitattu 17.2.2020].

Koehrsen, W. Neural Network Embeddings Explained: How deep learning can represent War and Peace as a vector. 2018. Medium-julkaisu. Saatavissa: https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526 [viitattu 5.8.2020].

Kumar, M. 2018. C# | Replace() Method. WWW-dokumentti. Saatavissa: https://www.geeksforgeeks.org/c-sharp-replace-method/ [viitattu 11.2.2020].
Lamping, U. & Warnicke, E. 2004. Wireshark user's guide. WWW-dokumentti. Saatavissa: https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs [viitattu 5.8.2020].

Lin, J. 2017. Keras Models: Sequential vs. Functional. WWW-dokumentti. Saatavissa: https://jovianlin.io/keras-models-sequential-vs-functional/ [viitattu 7.8.2020].

Linux.fi-wiki. s.a. Jaetut kirjastot. Wiki. Saatavissa: https://www.linux.fi/wiki/Jaetut_kirjastot [viitattu 2.4.2020].

Mamdoohi, H. 2014. Could not find file or assembly System.Web.Services Version=2.0.0.0. CodeProject-viesti. Saatavissa: https://www.codeproject.com/Questions/828117/Could-not-find-file-or-assembly-System-Web-Service [viitattu 8.8.2020].

Marr, B. 2019. Artificial Intelligence in Practice: How 50 Successful Companies Used AI and Machine Learning to Solve Problems. Padstow: TJ International Ltd.

Marshal.ReleaseComObject(Object) Method. s.a. Microsoft. Dokumentti. Saatavissa: https://docs.microsoft.com/en-us/dotnet/api/system.runtime.interopservices.marshal.releasecomobject?view=netframework-4.8 [viitattu 2.4.2020].

mentinet. 2016. Where can I find Microsoft.Office.Interop.Word.dll (2010)? StackOverflow-viesti. Saatavissa: https://stackoverflow.com/questions/28944379/where-can-i-find-microsoft-office-interop-word-dll-2010 [viitattu 13.2.2020].

Microsoft. 2019. How to: Install Office primary interop assemblies. Saatavissa: https://docs.microsoft.com/fi-fi/visualstudio/vsto/how-to-install-office-primary-interop-assemblies?redirectedfrom=MSDN&view=vs-2019 [viitattu 13.2.2020].

Microsoft. 2017. How to: Programmatically open existing documents. 2019. Saatavissa: https://docs.microsoft.com/en-us/visualstudio/vsto/how-to-programmatically-open-existing-documents?view=vs-2019 [viitattu 14.2.2020].

Nmap.org s.a. Chapter 12. Zenmap GUI User's Guide. WWW-dokumentti. Saatavissa: https://nmap.org/book/zenmap.html [viitattu 18.2.2020].

NumPy s.a. 2020. WWW-dokumentti. Saatavissa: https://numpy.org/ [viitattu 14.8.2020].

PDF redaction. s.a. Aspose. Editointiohjelma. Saatavissa: https://products.aspose.app/pdf/redaction [viitattu 17.3.2020].

Ramesh, A. 2014. Read document file in C#. CodeProject-viesti. Saatavissa: https://www.codeproject.com/Questions/742874/Read-document-file-in-Csharp [viitattu 19.3.2020].

Raster Edge s.a. How to Convert, make Adobe PDF document to text file (notepad .txt) using XDoc.PDF for .NET in C#, asp.net, aspx, Winforms, Azure. WWW-dokumentti. Saatavissa: http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-text/ [viitattu 15.3.2020].

Rathor, S. 2018. Simple RNN vs GRU vs LSTM :- Difference lies in More Flexible control. Medium-julkaisu. Saatavissa: https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57 [viitattu 4.9.2020]

Reaktor & Helsingin yliopisto. 2018. Tekoälyn perusteet: The Elements of AI. Verkkokurssi. Saatavissa: https://www.elementsofai.com/fi/ [viitattu 13.2.2020].

Replace Text in a PDF Document. s.a. Aspose. WWW-dokumentti. Saatavissa: https://docs.aspose.com/display/pdfnet/Replace+Text+in+a+PDF+Document [viitattu 17.3.2020].

Sanders, C. 2018. Analyzing Large Capture Files 4: Whittling with Filters. WWW-dokumentti. Saatavissa: https://chrissanders.org/2018/06/large-captures4-filter-whittling/ [viitattu 14.9.2020]

SautinSoft s.a. How to delete a specific text from DOCX document using C# and VB.Net. WWW-dokumentti. Saatavissa: https://www.sautinsoft.com/products/document/examples/delete-text-from-docx-document-net-csharp-vb.php [viitattu 18.3.2020].

Shukla, N. 2018. Machine learning with TensorFlow. Manning Publications Co.

Sinner. 2013. Symbol 'cout' could not be resolved. Eclipse Community Forums -viesti. Saatavissa: https://www.eclipse.org/forums/index.php/t/500293/ [viitattu 10.3.2020].

Strings (C# Programming Guide). s.a. Microsoft. WWW-dokumentti. Saatavissa: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/strings/ [viitattu 20.2.2020].

TechRepublic. 2008. How do I… Modify Word documents using C#? WWW-dokumentti. Saatavissa: https://www.techrepublic.com/blog/how-do-i/how-do-i-modify-word-documents-using-c/ [viitattu 19.3.2020].

TensorFlow. 2020. WWW-dokumentti. Saatavissa: https://www.tensorflow.org/ [viitattu 9.2.2020].

thenewboston. 2009. Java Programming Tutorial - 81 - Reading from Files. YouTube. Videoleike. Saatavissa: https://www.youtube.com/watch?v=3RNYUKxAgmw [viitattu 9.3.2020].

Trong, D. 2019. How to remove text in file pdf with iTextSharp. StackOverflow-viesti. Saatavissa: https://stackoverflow.com/questions/57622053/how-to-remove-text-in-file-pdf-with-itextsharp [viitattu 18.3.2020].

TubeMint. 2019. How to Install NetBeans 11 IDE & Java JDK on Windows 10/8/7. YouTube. Videoleike. Saatavissa: https://www.youtube.com/watch?v=yX-qG2Ooqzk [viitattu 8.3.2020].

Umer Softwares. 2019. How to use python in NetBeans IDE? YouTube. Videoleike. Saatavissa: https://www.youtube.com/watch?v=qmlAykxqQ8M [viitattu 9.3.2020].

user3738170. 2015. Adding a pdf file into Visual Studio. StackOverflow-viesti. Saatavissa: https://stackoverflow.com/questions/24272336/adding-a-pdf-file-into-visual-studio [viitattu 17.2.2020].

VintaSoft s.a. PDF: Remove content from PDF page. Redaction marks. WWW-dokumentti. Saatavissa: https://www.vintasoft.com/docs/vsimaging-dotnet/Programming-Pdf-Delete_Existing_Content_From_Pdf_Page.html [viitattu 17.3.2020].

Wasser, L. s.a. Hierarchical Data Formats – What is HDF5? WWW-dokumentti. Saatavissa: https://www.neonscience.org/about-hdf5 [viitattu 6.8.2020].

What is a DLL? s.a. Microsoft. WWW-dokumentti. Saatavissa: https://support.microsoft.com/en-us/help/815065/what-is-a-dll [viitattu 13.2.2020].

WireShark s.a. WWW-dokumentti. Saatavissa: https://www.wireshark.org/ [viitattu 8.8.2020].

WordApp s.a. Benefits for freelance writers. WWW-dokumentti. Saatavissa: https://www.wordapp.com/benefits-freelance-writers/ [viitattu 2.4.2020].

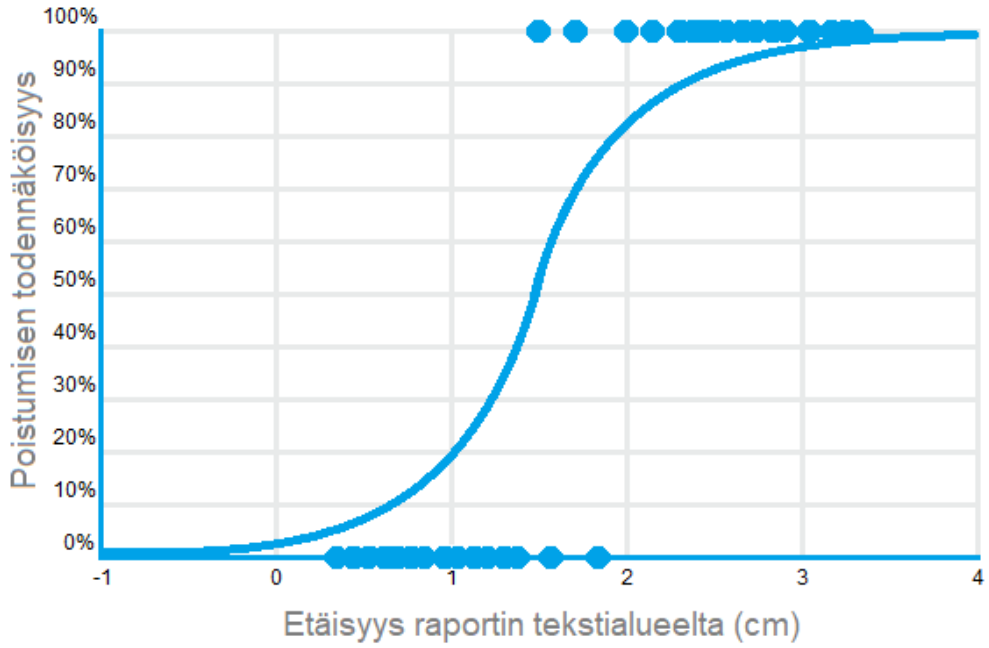
Xiao, Q., Li, K., Zhang, D. & Xu, W. 2018. Security risks in deep learning implementations. PDF-dokumentti. Saatavissa: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8424643 [viitattu 6.8.2020].

Young, R. 2019. Windows Forms application option seems to be missing? StackOverflow-viesti. Saatavissa: https://stackoverflow.com/questions/53212956/windows-forms-application-option-seems-to-be-missing [viitattu 12.2.2020].

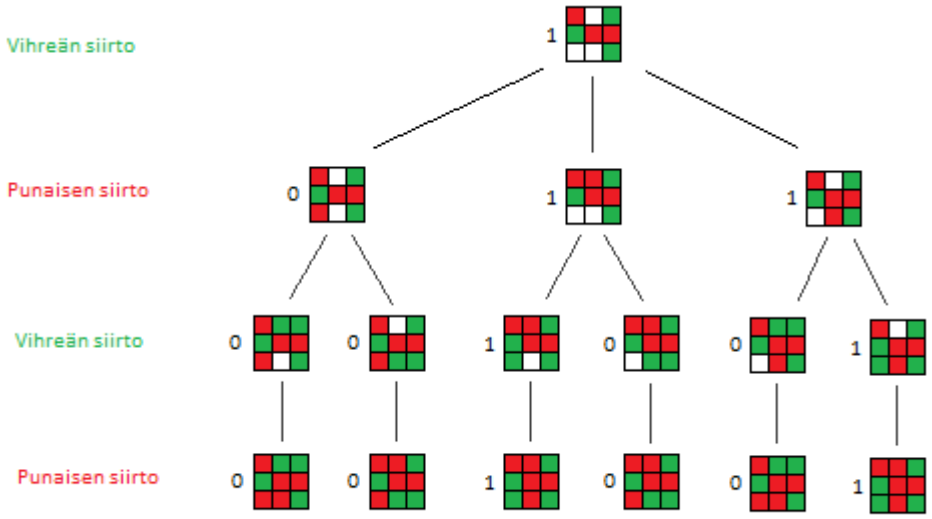
Zap s.a. Getting Started. WWW-dokumentti. Saatavissa: https://www.zaproxy.org/getting-started/ [viitattu 18.2.2020].

KUVALUETTELO

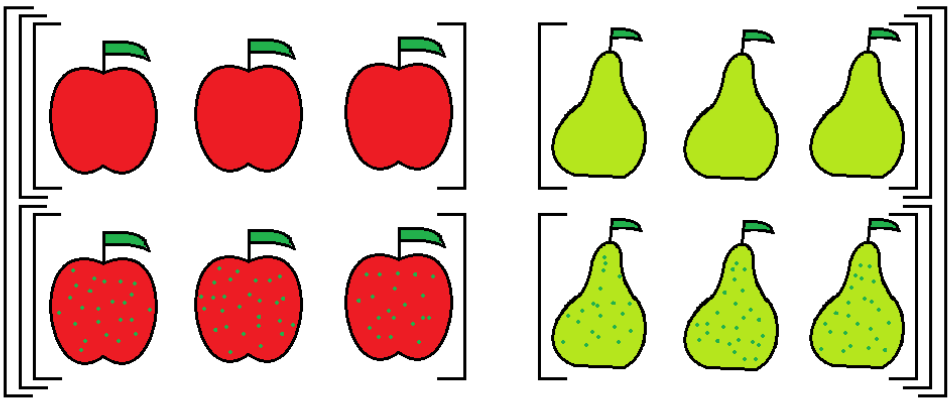
TYÖSSÄ KÄYTETYT KUVAT



Kuva 1. Poistuuko sana raportista?



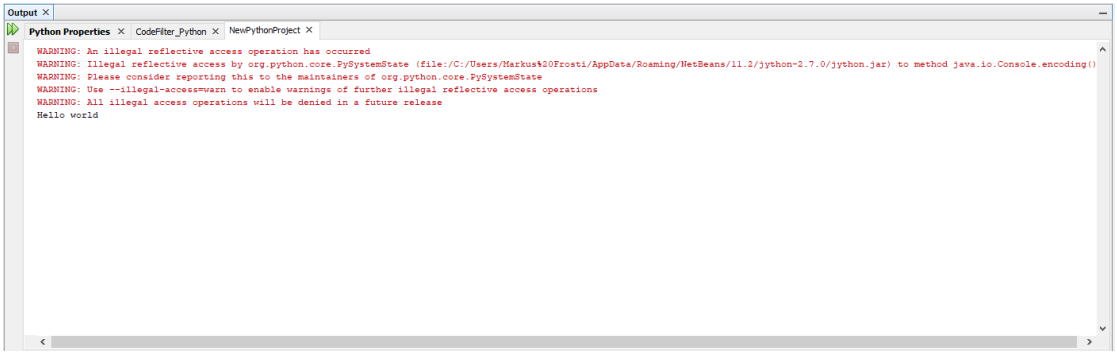
Kuva 2. Kaikki mahdolliset ristinollasiirrot



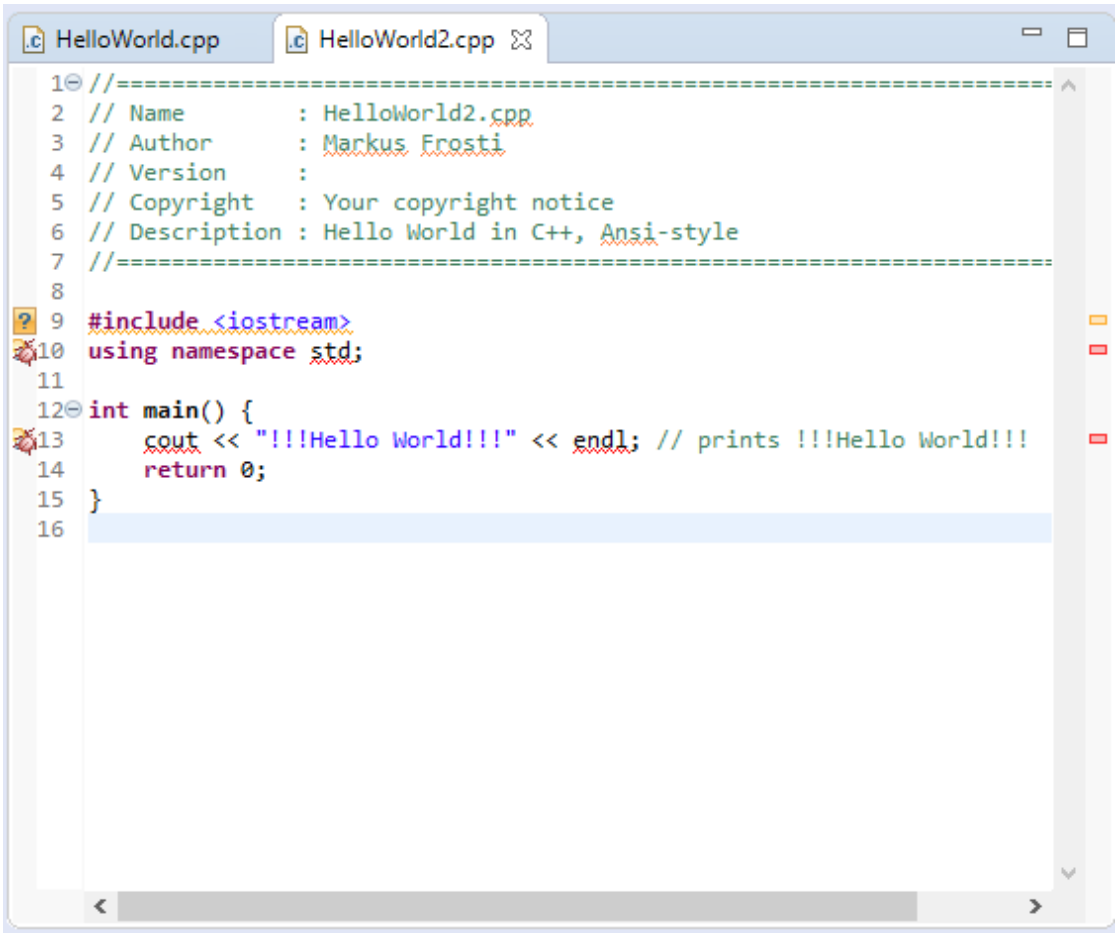
Kuva 3. Hedelmätensorit

```
using System;

namespace Koodinsiivousprototyyppi
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Which word would you like to remove? [Hello] [everybody] [all]");
            string answer = Console.ReadLine();

            string[] sentences = { "Hello everybody", "Hello all", "Hello everybody", "Hello all" };
            foreach (string sentence in sentences)
            {
                Console.WriteLine(sentence);
            }
            int x = 0;
            while (x < sentences.Length)
            {
                if (sentences[x].Contains(answer))
                {
                    sentences[x] = sentences[x].Replace(answer, " ");
                }
                x++;
            }

            Console.WriteLine("\t");

            foreach (string sentence in sentences)
            {
                Console.WriteLine(sentence);
            }
        }
    }
}
```

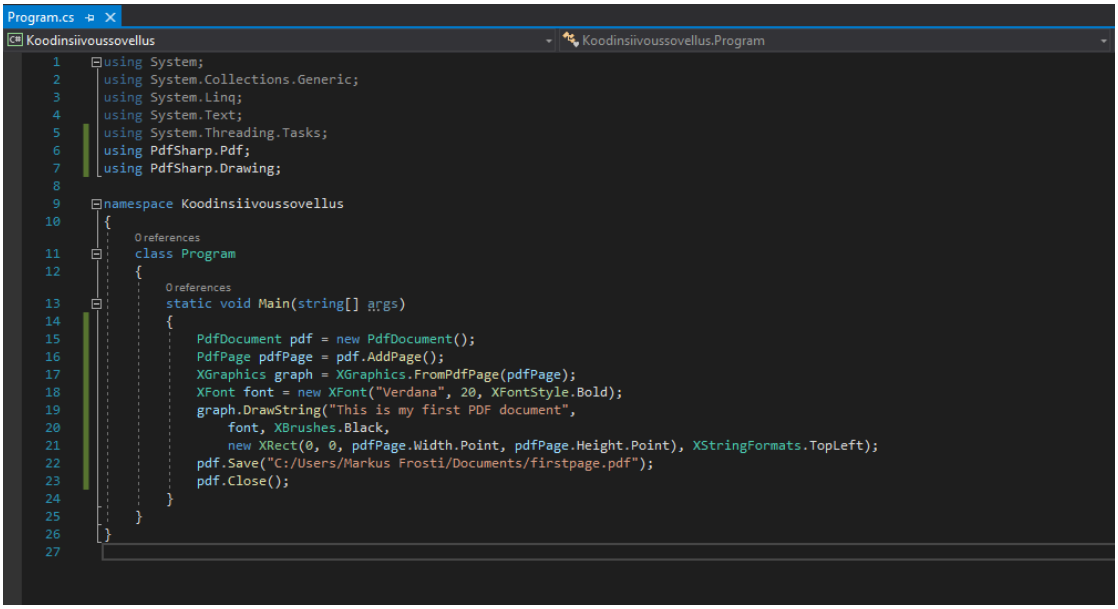
Kuva 4. Sananpoisto lauseesta



Kuva 5. Tuloste Jythonilla

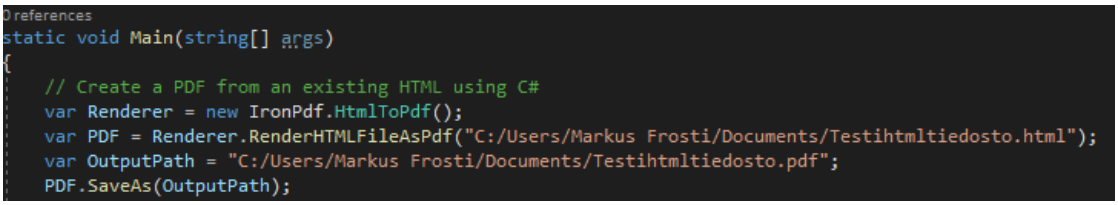


Kuva 6. Virheet koodissa



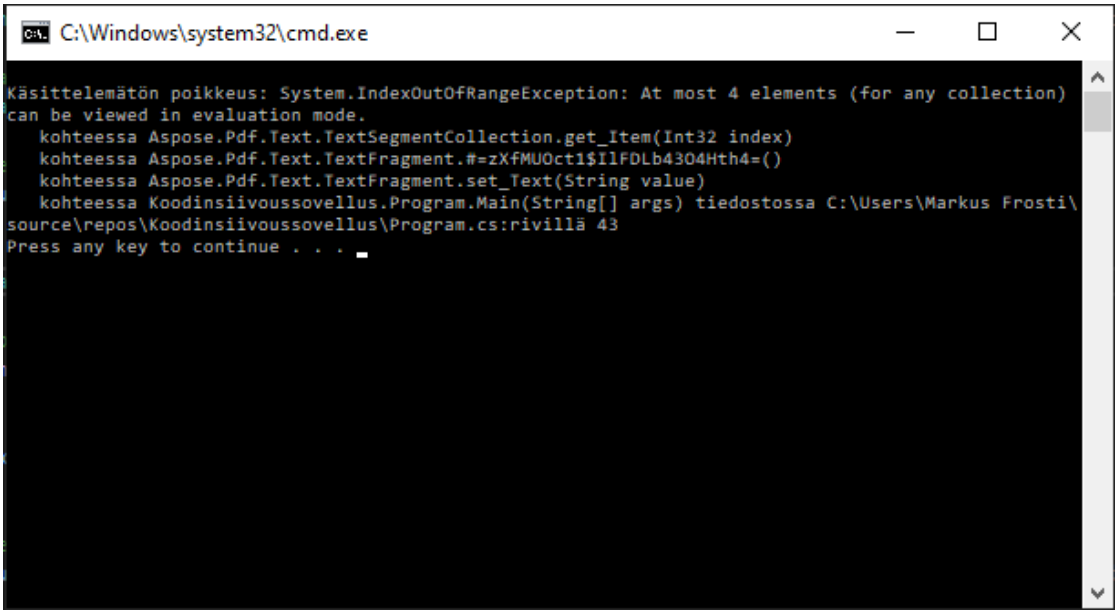
```
Program.cs
Koodinsiivoussovellus                          Koodinsiivoussovellus.Program
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using PdfSharp.Pdf;
using PdfSharp.Drawing;

namespace Koodinsiivoussovellus
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            PdfDocument pdf = new PdfDocument();
            PdfPage pdfPage = pdf.AddPage();
            XGraphics graph = XGraphics.FromPdfPage(pdfPage);
            XFont font = new XFont("Verdana", 20, XFontStyle.Bold);
            graph.DrawString("This is my first PDF document",
                font, XBrushes.Black,
                new XRect(0, 0, pdfPage.Width.Point, pdfPage.Height.Point), XStringFormats.TopLeft);
            pdf.Save("C:/Users/Markus Frosti/Documents/firstpage.pdf");
            pdf.Close();
        }
    }
}
```

Kuva 7. PDF-tiedoston luominen



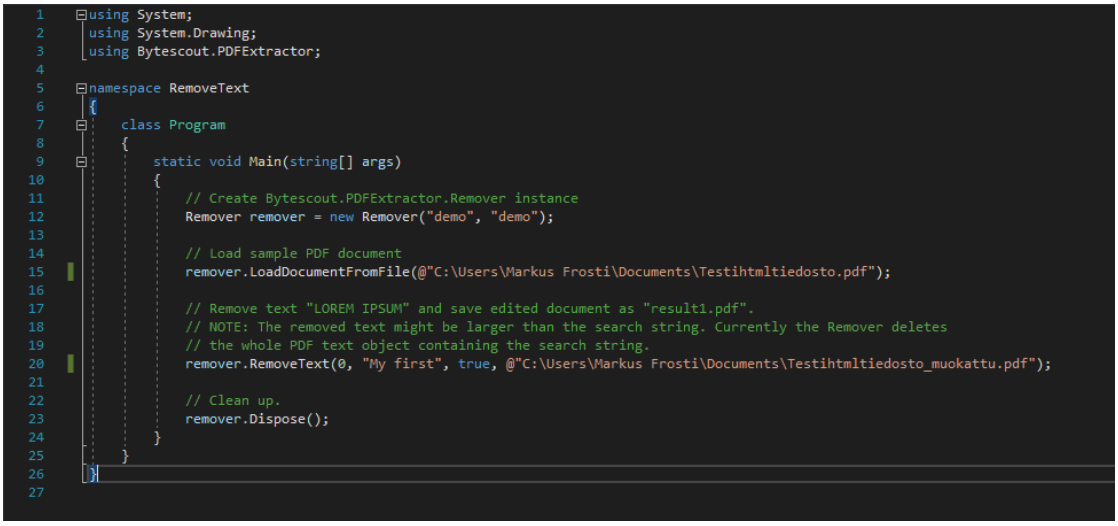
```
0 references
static void Main(string[] args)
{
    // Create a PDF from an existing HTML using C#
    var Renderer = new IronPdf.HtmlToPdf();
    var PDF = Renderer.RenderHTMLFileAsPdf("C:/Users/Markus Frosti/Documents/Testihtmltiedosto.html");
    var OutputPath = "C:/Users/Markus Frosti/Documents/Testihtmltiedosto.pdf";
    PDF.SaveAs(OutputPath);
```

Kuva 8. Sananpoisto lauseesta (Iron PDF)



```
C:\Windows\system32\cmd.exe

Käsittelemätön poikkeus: System.IndexOutOfRangeException: At most 4 elements (for any collection)
can be viewed in evaluation mode.
   kohteessa Aspose.Pdf.Text.TextSegmentCollection.get_Item(Int32 index)
   kohteessa Aspose.Pdf.Text.TextFragment.#=zXfMUOct1$IlFDLb43O4Hth4=()
   kohteessa Aspose.Pdf.Text.TextFragment.set_Text(String value)
   kohteessa Koodinsiivoussovellus.Program.Main(String[] args) tiedostossa C:\Users\Markus Frosti\
source\repos\Koodinsiivoussovellus\Program.cs:rivillä 43
Press any key to continue . . .
```

Kuva 9. Virheilmoitus taulukon ulkopuolelle siirtymisestä

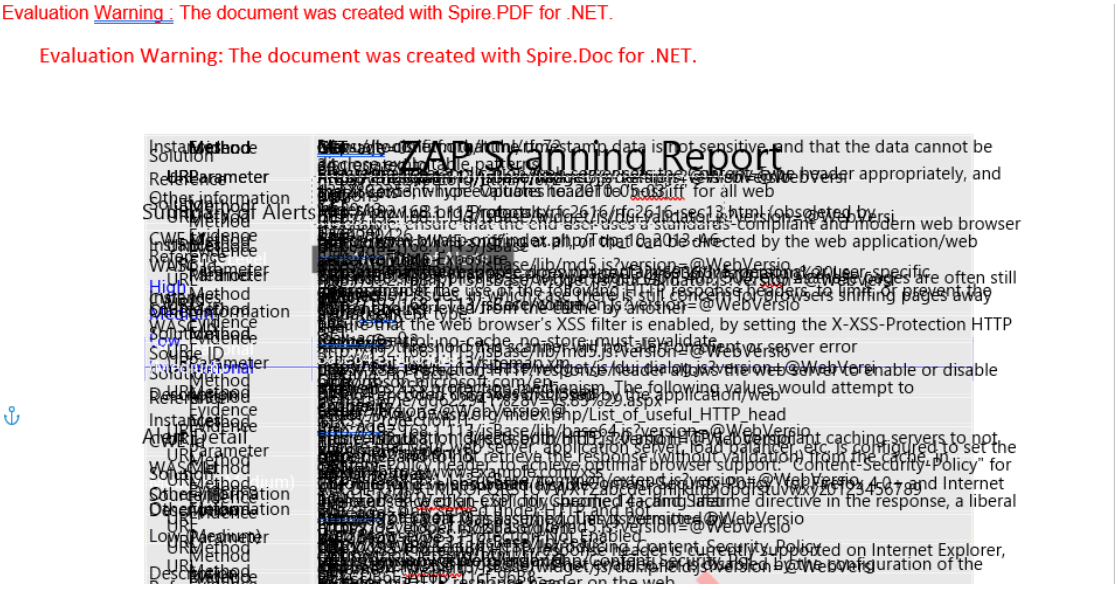


```
using System;
using System.Drawing;
using Bytescout.PDFExtractor;

namespace RemoveText
{
    class Program
    {
        static void Main(string[] args)
        {
            // Create Bytescout.PDFExtractor.Remover instance
            Remover remover = new Remover("demo", "demo");

            // Load sample PDF document
            remover.LoadDocumentFromFile(@"C:\Users\Markus Frosti\Documents\Testihtmltiedosto.pdf");

            // Remove text "LOREM IPSUM" and save edited document as "result1.pdf".
            // NOTE: The removed text might be larger than the search string. Currently the Remover deletes
            // the whole PDF text object containing the search string.
            remover.RemoveText(0, "My first", true, @"C:\Users\Markus Frosti\Documents\Testihtmltiedosto_muokattu.pdf");

            // Clean up.
            remover.Dispose();
        }
    }
}
```

Kuva 10. Tekstinpoisto (ByteScout s.a.)

```
1 reference
static void HtmlToDoc(string filepath)
{
    var Renderer = new IronPdf.HtmlToPdf();
    var PDF = Renderer.RenderHTMLFileAsPdf(filepath + ".html");
    var OutputPath = filepath + ".pdf";
    PDF.SaveAs(OutputPath);

    PdfDocument doc = new PdfDocument();
    doc.LoadFromFile(OutputPath);
    doc.SaveToFile(filepath + ".doc", FileFormat.DOC);
    File.Delete(OutputPath);
}
```

Kuva 11. Muutos HTML-formaatista Word-formaattiin



Kuva 12. Sanojen peittyminen asiakirjassa

ZAP Scanning Report

Summary of Alerts

High

0

1

3

4

Medium

Low

Informational

Alert Detail

Description

The site is only served under HTTP and not HTTPS.

URL

Instances

Solution

Trial information

Reference

CWE Id

WASC Id

Source ID

Low (Medium)

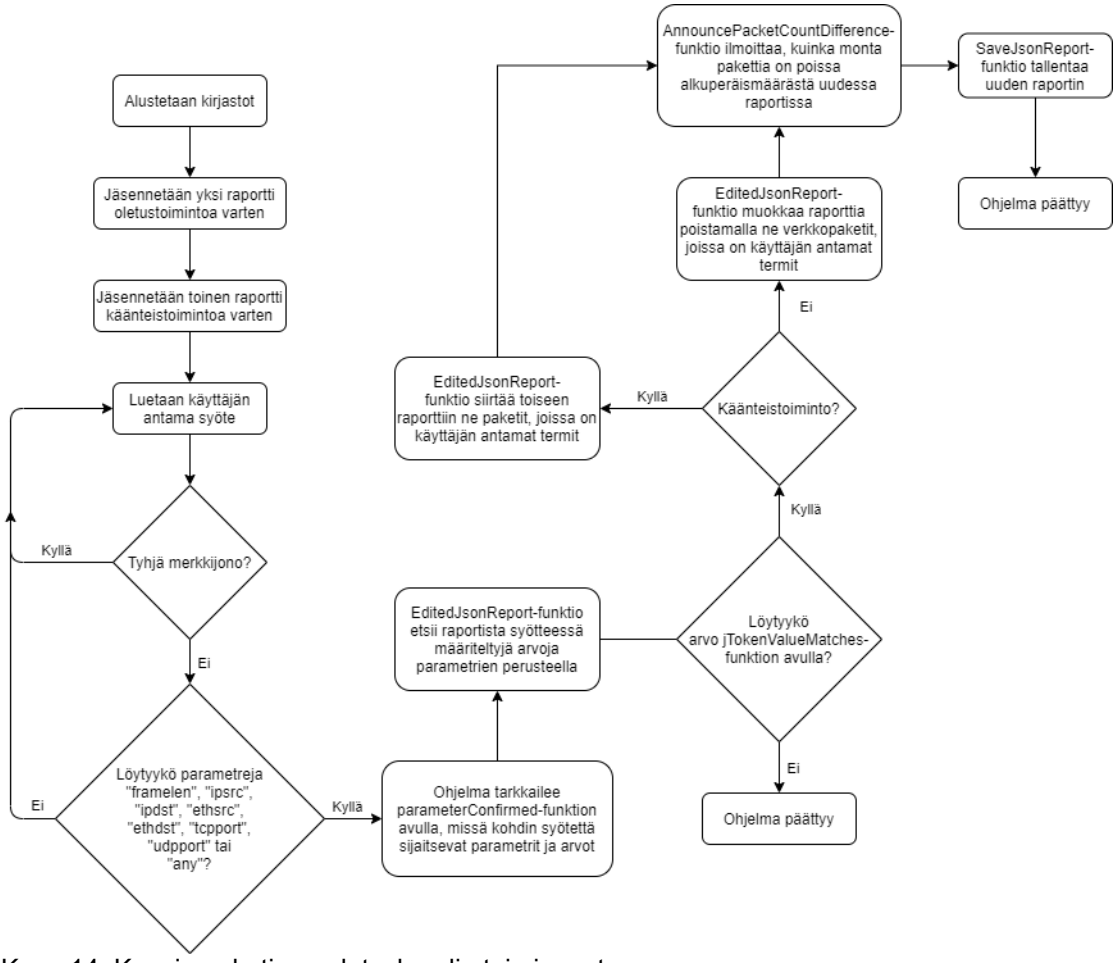
ct and mitigate certain

ks. These attacks are

e. CSP provides a set

ces of content that

Kuva 13. Heikkolaatuinen HTML-sivu



Kuva 14. Kaavio paketinsuodatuskoodin toiminnasta