Kaapo Immonen

Avainsanapohjaisen luonnollisen kielen käsittelyn (NLP) soveltuvuus rikosilmoitusten luokitteluun

Tekniikan ja innovaatiojohtamisen akateeminen yksikkö
Tekniikan Kandidaatintutkielma
Data-arkkitehtuurin koulutusohjelma

Vaasa 2024

VAASAN YLIOPISTO
Tekniikan ja innovaatiojohtamisen akateeminen yksikkö
Tekijä: Kaapo Immonen
Tutkielman nimi: Avainsanapohjaisen luonnollisen kielen käsittelyn (NLP) soveltuvuus rikosilmoitusten luokitteluun
Tutkinto: Tekniikan kandidaatti
Oppiaine: Data-arkkitehtuuri
Työn ohjaaja: Maarit Välisuo
Valmistumisvuosi: 2024 **Sivumäärä:** 62

TIIVISTELMÄ:
Suomen poliisin käsiteltäväksi tulee päivittäin yli tuhat rikosilmoitusta. Verkkorikollisuuden kasvu ja rikoskokonaisuuksien jatkuva monimutkaistuminen kuormittavat resurssillisesti rajallista rikostutkintaa, ja siksi myös tarve kehittää uusia työkaluja menetelmiä kasvaa. Yksi potentiaalinen ratkaisu on käsitellä ihmisen tuottamaa tekstiä tietokoneella hyödyntäen luonnollisen kielen käsittelyä (NLP). Tässä työssä tutkitaan, miten NLP soveltuu luokittelemaan ilmoituksia rikosnimikkeisiin niissä olevan selostuksen perusteella.

Työssä perehdytään luonnollisen kielen käsittelyyn dokumenttien luokittelussa ja sen edellytyksiin. Työssä käsitellään kohinan poistamista hyödyntäen esikäsittelymenetelmiä, kuten stemmausta ja lemmatisointia. Esitetään, miten esikäsitellyt syötteet muunnetaan numeeriseksi ominaisavaruudeksi käyttämällä menetelmiä, kuten TF-IDF ja word embedding. Lisäksi tutkielmassa tarkastellaan, miten luokittelualgoritmi optimoidaan ja koulutetaan suorittamaan luokittelua näiden ominaisuuksien perusteella.

Työssä toteutettavien mallien suorituskykyä arvioidaan vertaamalla niiden luokittelun tarkkuutta suomalaisten lainvalvontaviranomaisten manuaaliseen luokitteluun. Perinteisten NLP-menetelmien lisäksi työssä käyttöönotetaan ja hienosäädetään esikoulutettu suomenkielinen FinBERT-malli, jotta voidaan tutkia NLP:n mullistaneen transformer-arkkitehtuurin suorituskykyä osana vertailuasetelmaa.

Mallien vertailussa esikoulutettu FinBERT-malli osoittautui yleistyskyvyltään suorituskykyisimmäksi malliksi. Myös yksittäisten sanojen merkitystä korostavat TF-IDF-menetelmää hyödyntävät mallit suoriutuivat huomattavan hyvin, erityisesti käytettäessä stemmausta esikäsittelyvaiheessa. Lemmatisointiin perustuvat mallit puolestaan heikkenivät liiallisen yksinkertaistamisen vuoksi, ja word embedding -menetelmät tuottivat heikoimmat tarkkuustulokset pienen sanaston ja kohinan takia.

Tulokset tukevat käsitystä siitä, että NLP-mallit kykenevät tunnistamaan rikosilmoitusten keskeisiä ominaisuuksia, mutta tarkkuutta voidaan parantaa etenkin esikäsittelyn ja ominaisuuksienpoiminnan kehittämisellä. FinBERT-mallin menestys osoittaa, että transformer-arkkitehtuuriin perustuvat mallit ovat erityisen suorituskykyisiä luokittelussa, joissa koulutusdata on rajallista.

AVAINSANAT: Avainsanapohjainen AI, Luonnollisen kielen käsittely, NLP, Word Embedding, TF-IDF, Dokumenttien luokittelu

Sisällys

Kuviot

Taulukot

Lyhenteet

NLP	Natural Language Processing
TF-IDF	Term Frequency-Inverse Document Frequence
FinBERT	Finnish bidirectional encoder representations from transformers
LogReg	Logistic Regression
SVM	Support Vector Machine

1 Johdanto

Poliisille kirjattujen rikosilmoitusten määrä vaihtelee vuosittain. Poliisin tietoon rikoksia ja rikkomuksia tuli vuoden 2023 aikana yhteensä 520 300, joista poliisi sai selvitettyä 67,55 % (Tilastokeskus, 2023). Valtakunnallistesti poliisin käsiteltäväksi tulee noin 1 425 ilmoitusta päivittäin. Samalla yhteiskunnan ja globaalin ympäristön muutokset monimuotoistavat rikoksia, ja hankaloittavat rikostutkinnan toimintaympäristöä. Etenkin digitalisaation myötä kasvanut verkkorikollisuus kuormittaa poliisia, sillä tutkintakokonaisuudet muuttuvat ja monimutkaistuvat. Rikosilmoituksen voi tehdä kuka tahansa, ja poliisilla on velvollisuus kirjata sille rikoksena tutkittavaksi ilmoitettavat asiat (Esitutkintalaki 805/2011, 3:1 §) sekä toimittaa päätös esitutkinnasta ilman aiheetonta viivytystä (Esitutkintalaki 805/2011, 3:11 §). Poliisin toimintaan vaikuttavat muuttuvan toimintaympäristön ja lainsäädännön lisäksi myös henkilöstö- ja taloudelliset resurssit. Henkilöstöresurssien määrän lisääminen ei yksin ratkaise rikostutkinnan haasteita, vaan on tarve kehittää uusia työkaluja ja menetelmiä, jotka vastaavat paremmin modernin rikostutkinnan toimintaympäristön vaatimuksia.

Luonnollisen kielen käsittely (engl. *natural language processing*, NLP) on kasvattanut lähivuosina suosiotaan tekoälypohjaisten teknologioiden, erityisesti laajojen kielimallien räjähdysmäisen kasvun myötä. NLP on tekoälyn osa-alue, jonka keskityksenä on ihmisen tuottaman kielen ymmärtäminen ja käsittely tietokoneiden avulla (Eisenstein, 2018, s. 1). Yksi luonnollisen kielen käsittelyn keskeisistä sovelluksista on dokumenttien luokittelu, jossa asiakirjat luokitellaan yhteen tai useampaan luokkaan niiden asiasisällön perusteella.

Rikosilmoitus sisältää selostusosion johon ilmoittaja kirjaa kertomuksen rikoksesta ja sen tapahtumasarjasta. Tämä ihmisen tuottama selostus toimii perustana poliisin päätökselle siitä, epäilläänkö rikoksen tapahtuneen (Poliisi, n.d). Luonnollisen kielen käsittelyn avulla rikosilmoitusten selostusosioita voidaan analysoida automaattisesti. Selostuksesta voidaan automaattisesti tunnistaa oleellisia tietoja tai luokitella ilmoitus tiettyyn

rikosnimikkeeseen sen piirteiden perusteella, mikä nopeuttaa ilmoitusten käsittelyä ja tehostaa rikostutkintaa.

1.1 Tutkielman tavoite ja rakenne

Tässä kandidaatintutkielmassa arvioidaan, kuinka luonnollisen kielen käsittelyä voidaan soveltaa rikosilmoitusten käsittelyn tehostamiseksi. Erityisesti tutkielmassa tarkastellaan, miten tarkasti avainsanapohjaisia TF-IDF- ja word embedding -menetelmiä hyödyntävät NLP-mallit luokittelevat rikosilmoituksen asianmukaiseen rikosnimikkeeseen sen selostuksen perusteella. Avainsanapohjainen lähestymistapa on valittu, sillä se on yksinkertainen mutta tehokas tapa tunnistaa rikosnimikkeille tyypillisiä ominaispiirteitä, joiden perusteella luokittelu voidaan tehdä. Avainsanapohjaiset luokittelijat vaativat vähemmän koulutusdataa verrattuna syväoppimismalleihin. Tämä tekee niistä sopivia tutkielmaan, jossa dataa ja resursseja on rajallisesti. Tutkielmassa luokiteltaviksi rikosnimikkeiksi on valittu petos, varkaus ja näpistys. Kokeellisessa osassa toteutettujen NLP-mallien luokittelun tuloksia verrataan suomalaisten lainvalvontaviranomaisten tekemään manuaaliseen luokitteluun.

Lisäksi tutkielmassa verrataan miten syväoppimisen transformer-arkkitehtuuriin perustuva Virtasen ja muiden (2019) kehittämä suomen kielelle esikouluttettu FinBERT-malli (engl. *Finnish bidirectional encoder representations from transformers*) suoriutuu luokittelusta verrattuna muihin tutkielmassa toteutettuihin luokittelijoihin.

Kandidaatintutkielmassa pyritään vastaamaan seuraaviin kysymyksiin:

1. Soveltuuko avainsanapohjainen luonnollisen kielen käsittely rikosilmoitusten luokitteluun?
2. Miten eri menetelmät ja luokittelijat vaikuttavat luokittelun tarkkuuteen?
3. Suoriutuuko esikoulutettu syväoppimismalli paremmin kuin käyttökohteeseen rajallisella datalla koulutettu luokittelija?

Kysymykset 2 ja 3 toimivat kysymyksen 1 alikysymyksinä. Alikysymysten tarkoitus on tukea tutkielman menetelmien valintaa, sekä ohjata tulosten perusteella tehtäviä

johtopäätöksiä. Tutkielmassa arvioidaan vain NLP-teknologian teknistä soveltuvuutta, eikä esimerkiksi soveltuvuutta käsitellä lainsäädännön tai eettisyyden näkökulmista.

Tämä tutkielma jakautuu kolmeen pääosaan: taustaosaan, teoriaosaan ja kokeelliseen osuuteen. Seuraavassa osassa esitellään tutkielman taustaa, jossa tarkastellaan valittujen rikosnimikkeiden valintaa ja luonnollisen kielen käsittelyn historiaa sekä aiempia aiheen tutkimuksia. Teoreettinen viitekehys esitetään pääluvussa 3, jossa käsitellään kokeellisessa osassa käytettyjen konseptien ja menetelmien teoriaa sekä toimintaa. Luvussa tarkastellaan muun muassa datan esikäsittelyä, ominaisuuksien poimintamenetelmiä ja käytettyjä luokittelualgoritmeja. Lisäksi perehdytään transformer-arkkitehtuuriin ja siihen perustuvan FinBERT-mallin toimintaan.

Pääluvut 4 ja 5 muodostavat tutkielman kokeellisen osuuden. Luvussa 4 esitellään kokeellisessa osassa käytetty aineisto ja kuvataan teoreettiseen viitekehykseen perustuvien mallien käytännön toteutusta, niiden konfiguraatioita sekä koulutusvaiheen jälkeistä suorituskykyä. Luvussa 5 analysoidaan mallien lopullista toimivuutta ja tarkastellaan niiden tekemää luokittelua tarkemmin. Viimeisessä pääluvussa 6 pyritään vastaamaan tutkielman tutkimuskysymyksiin ja esittämään tulosten perusteella tunnistettuja kehityskohteista jatkotutkimuksia varten.

2 Tutkielman tausta

Tässä luvussa pohjustetaan tarkemmin kandidaatintutkielmaan valittuja rajauksia, sekä esitetään tutkielmalle keskeisiä taustamateriaaleja. Ensimmäisessä alaluvussa esitetään tutkielmaan valitut rikosnimikkeet ja niiden merkittävyys tutkielmalle. Toisessa alaluvussa esitellään NLP-teknologian keskeiset kehitykset sekä tutustutaan muihin tutkielmalle relevantteihin luonnollisen kielen käsittelyn tutkimuksiin.

2.1 Rikosnimikkeet

Rajallisten resurssien, etenkin käytettävissä olevan datan takia kandidaatintutkielmaan on valittu kolme rikosnimikettä, joiden perusteella NLP-mallien koulutus ja luokittelu suoritetaan. Kolme rikosnimikettä luo moniluokkaisen asetelman, joka on tutkielmalle riittävä arvioidessa avainsanapohjaisen luonnollisen kielen käsittelyn soveltuvuutta pitäen samalla työmäärän tutkielmalle sopivana. Seuraavissa alaluvuissa esitetään tutkielmaan valitut rikosnimikkeet sekä perustellaan niiden sopivuutta.

2.1.1 Varkaus

Suomen rikoslaissa varkaus määritetään seuraavasti:

> Joka anastaa toisen hallusta irtainta omaisuutta, on tuomittava varkaudesta sakkoon tai vankeuteen enintään yhdeksi vuodeksi kuudeksi kuukaudeksi. Yritys on rangaistava. (Rikoslaki 769/1990, 28:1 §)

Nimikkeenä varkaus on varkausrikosten perusmuotoinen teko. Muut varkausrikokset kuten näpistys tai törkeä varkaus sisältävät varkauden tunnusmerkkien lisäksi erikseen määritettyjä ominaispiirteitä tai seikkoja.

Tilastokeskuksen (2023) mukaan vuonna 2023 kaikista viranomaisten tietoon tulleista rikoksista noin neljäsosa oli varkausrikoksia, ja rikoksista n. 45 % sarjoittuivat

nimikkeeseen varkaus. Vuonna 2023 varkausrikosten selvitysprosentti oli 37,4 %, ja varkaus-nimikkeen selvitysprosentti oli huomattavasti matalampi: vain 16,1 %. Beukerin (2023, s. 117) mukaan matalaan selvitysprosenttiin vaikuttaa muun muassa yksinkertaiset, mutta tutkinnallisesti vaikeasti selvitettävät rikokset kuten polkupyörävarkaudet. Beuker perustelee, että selvitysaste voi nousta merkittävästi poliisin onnistuessa selvittämään joitakin laajoja rikossarjoja kyseisenä vuonna.

Varkaus on valittu tutkielmaan rikosnimikkeeksi, koska se on yhteiskunnallisesti näkyvä sekä tilastollisesti merkittävä osa Suomessa tapahtuvaa rikollisuutta. Luonnollisen kielen käsittely voi tehostaa ilmoitusten käsittelyä automatisoimalla rikosten luokittelua, nopeuttaen käsittelyaikoja. Laajemmin kehitettynä NLP-malli voisi esimerkiksi tunnistaa ilmoituksia, jotka ovat selvitettävissä tai potentiaalisesti toisen rikoksen kanssa sarjoittuvia. Automaattinen sarjoitus nopeuttaisi rikosten välisien yhteyksien selvittämistä, ja auttaa viranomaisia tilannekuvan muodostamisessa sekä operatiivisen toiminnan ohjaamisessa.

2.1.2 Näpistys

Näpistys on toinen tutkielmaan valittu varkausrikos, ja on Suomen rikoslaissa määritetty seuraavasti:

> Jos varkaus, huomioon ottaen anastetun omaisuuden arvo tai muut rikokseen liittyvät seikat, on kokonaisuutena arvostellen vähäinen, rikoksentekijä on tuomittava näpistyksestä sakkoon. Yritys on rangaistava. (Rikoslaki 769/1990, 28:3 §)

Näpistys on lievempi muoto varkaudesta. Tilastokeskuksen (2023) mukaan n. 53 % kaikista vuonna 2023 kirjatuista varkausrikoksista oli näpistyksiä. Tilastoista ilmenee, että näpistysten selvitysprosentti varkausrikoksista korkein: 55,6 %. Beuker (2023, s. 117) perustelee korkeaa selvitysastetta rikostyypin ilmitulotavasta. Hänen mukaansa näpistys tulee yleensä ilmi tekijän jäädessä paikan päällä kiinni, jolloin se voidaan suoraan kirjata selvitetyksi.

Rikoslain määritelmän mukaan anastetun omaisuuden arvo on keskeisessä asemassa määrittäessä tekoa varkauden ja näpistyksen välillä (RL 28:3 §). Laissa ei kuitenkaan määritellä tarkkaa rajaa omaisuuden arvolle, jonka perusteella teko on näpistys. Kihlakunnan syyttäjä Pörsti sekä oikeustieteen tohtori Paukku on esittänyt Savolaisen (2024, s. 9) opinnäytetyössä näpistyksen rahamääräiseksi rajaksi 500 euroa. Savolaisen mukaan samaa määritystä käytetään Poliisiammattikorkeakoulussa sekä Sisä-Suomen poliisilaitoksella. Lakisäädännön tulkinnanvaraisuus vaikeuttaa mahdollisten automaatioratkaisuiden tai -teknologioiden käyttöönottamista, sillä järjestelmät vaativat tarkkoja määritelmiä suorittaakseen esimerkiksi luokittelua tarkasti.

Näpistys on valittu tutkielmaan, koska se on yhteiskunnallisen merkityksen lisäksi vahvasti liitännäinen nimikkeeseen varkaus. Valitsemalla kaksi varkausrikosta, voidaan paremmin arvioida NLP-mallin kykyä hahmottamaan kahden samankaltaisen nimikkeen väliset erot.

2.1.3 Petos

Kolmas tutkielmaan valittu rikosnimike on petos. Petoksen määritelmä rikoslaissa on seuraava:

> Joka, hankkiakseen itselleen tai toiselle oikeudetonta taloudellista hyötyä taikka toista vahingoittaakseen, erehdyttämällä tai erehdystä hyväksi käyttämällä saa toisen tekemään tai jättämään tekemättä jotakin ja siten aiheuttaa taloudellista vahinkoa erehtyneelle tai sille, jonka eduista tällä on ollut mahdollisuus määrätä, on tuomittava petoksesta sakkoon tai vankeuteen enintään kahdeksi vuodeksi.
>
> Petoksesta tuomitaan myös se, joka 1 momentissa mainitussa tarkoituksessa dataa syöttämällä, muuttamalla, tuhoamalla tai poistamalla taikka tietojärjestelmän toimintaan muuten puuttumalla saa aikaan tietojenkäsittelyn lopputuloksen vääristymisen ja siten aiheuttaa toiselle taloudellista vahinkoa. Yritys on rangaistava. (Rikoslaki 769/1990, 36:1 §)

Petos on nimikkeenä petosrikoksen perusmuotoinen teko. Viimeisen vuosikymmenen aikana petosrikosten määrä on kaksinkertaistunut (Beuker, 2023, s. 104). Beuker perustelee kasvua tietotekniikan kehittymisellä, joka on mahdollistanut tekotavaltaan täysin

uudenlaisia petosrikoksia. Tilastokeskuksen (2023) mukaan vuotena 2023 petosrikoksia kirjattiin yhtensä 41 889.

Petosrikosten rikoshyöty on mittava, vaikka petosrikokset muodostavat vain noin 8 % kaikista poliisin tietoon tulleista rikoksista. Finanssiala ry:n (2024) pankeilta kerättyjen tietojen mukaan vuonna 2023 petoshuijauksien yritetty kokonaisrikoshyöty Suomessa oli 76,9 miljoonaa euroa. Suomalaisilta huijattu rikoshyöty oli 44,2 miljoonaa euroa, sillä pankit onnistuivat torjumaan tai palauttamaan noin 43 % huijareille menevistä varoista. Huijausmäärät jatkavat kasvuaan, ja vuoden 2024 ensimmäisellä puoliskolla huijausten yritetty rikoshyöty oli jo 45,7 miljoonaa euroa (Finanssiala, 2024). Lisäksi Beuker (2023, s. 115) kertoo, että omaisuusrikoksista yhä suurempi osa on siirtynyt verkkoon.

Petos on valittu tutkielmaan, koska se on merkittävyyden lisäksi tekotavaltaan eriävä varkausrikoksista. Kolmannella rikosnimikkeellä tutkielmaan saadaan luotua moniluokkaisen luokittelun asetelma, joka on kaksiluokkaista luokittelua sopivampi, sillä realistisesti rikosnimikkeitä on enemmän kuin kaksi. Kaksiluokkainen luokittelu olisi sopiva, mikäli haluttaisiin rikosnimikkeeseen luokittelun sijaan esimerkiksi tunnistaa selostuksesta, onko rikosta ylipäätänsä tapahtunut. Petosrikokset ovat yhä enemmän esillä julkisuudessa, koska ne aiheuttavat valtakunnallisesti mittavia taloudellisia menetyksiä suomalaisille. Tehostamalla ilmoitusten käsittelyä saataisiin nopeammin tietoa esimerkiksi uusista tekotavoista, joiden avulla petoksia voidaan tutkia ja ennaltaehkäistä entistä tehokkaammin.

2.2 Luonnollisen kielen käsittely

2.2.1 NLP:n kehitysvaiheet

Luonnollisen kielen käsittelystä, menetelmistä ja niiden soveltuvuudesta eri käyttökohteisiin on laajasti tutkimuksia läpi teknologian kehitysvaiheiden. Varhaiset 1980- ja 1990-luvun NLP-menetelmät perustuivat sääntöpohjaisiin, ennalta määritettyihin sanastoihin

ja kielioppisääntöihin. Marcus (1993) esittelee Penn Treebank -nimisen esikäsitellyn laajan englanninkielisen korpuksen, jossa jokaiselle lauseelle on määritelty erikseen syntaktinen jäsennys. Jokaiselle Penn Treebank -korpuksen sanalle on määritetty sen osasynonyymi sekä syntaktinen rooli lauseessa. Penn Treebankin annotaatio on ollut perustana monessa syntaktisen jäsennyksen mallissa sekä koneoppimisalgoritmien kehityksessä.

1990-luvun loppupuolella etenkin dokumenttien luokittelussa alkoi siirtymä tilastollisiin menetelmiin perustuviin NLP-malleihin. Joachims (1998) demonstroi julkaisussaan, kuinka Vapnikin ja Cortesin kehittämää tukivektorikonetta (engl. *Support vector machine*, SVM) voidaan hyödyntää tekstien luokittelussa. Tilastollisiin menetelmiin perustuvien mallien etuna on niiden kyky oppia piirteitä ja todennäköisyyksiä korpuksesta, ilman erikseen määriteltyjä sääntöjä.

2010-luvulla Mikolov ja muut (2013) esittelevät julkaisussaan, kuinka sanoja sekä niiden muodostamien lauseiden välisiä semanttisia suhteita voidaan mallintaa ja käsitellä matemaattisina sanavektoreina. Mikolov ja muut esittelevät kehittämänsä neuroverkkopohjaisen Word2Vec-algoritmin, joka muuntaa syötetyn korpuksen vektoriavaruudeksi. Termi word embedding tarkoittaa asiayhteyden mukaan sanan vektoriesitysmuotoa tai prosessia, jossa sanat muunnetaan vektorimuotoon. Word embedding on NLP-menetelmänä laajasti käytetty etenkin kone- ja syväoppimisessa.

Yksi NLP:n ja syväoppimisen uusimmista kulmakivistä on transformer-arkkitehtuuri, jonka itsehuomio-mekanismi (engl. *self-attention*) mahdollistaa syötteiden rinnakkaisen prosessoinnin, parantaen mallin koulutusnopeutta huomattavasti (Vaswani ja muut, 2017, s. 6–7). Julkaisun transformer-arkkitehtuuriin perustuva malli suoriutui paremmin kuin perinteisimpiin RNN tai LSTM neuroverkkorakenteisiin pohjautuvat mallit. Transformer-arkkitehtuuri toimii perustana suosioon nousseille esikoulutetuille laajoille kielimalleille kuten GPT, Claude, Gemini ja BERT.

2.2.2 NLP ja tutkielman konteksti

Luonnollisen kielen käsittelyä ja sen soveltuvuutta tukemaan turvallisuusviranomaisten toimintaa on tutkittu jonkin verran. Vysotka ja muut (2022) demonstroivat konferenssijulkaisussa, kuinka NLP-menetelmiä käyttämällä voidaan analysoida rikosilmoituksia tai tunnistaa disinformaatiota ja propagandaa. Etenkin rikollisuuden osa-alueella tutkimukset keskittyvät NLP-menetelmien hyödyntämiseen luokittaessa rikollisuuteen liittyviä uutisartikkeleita asianmukaisiin rikoslaissa säädettyihin nimikkeisiin tai oikeudenkäynnin prosessin tunnistamiseen rikosasian ja siviiliasian välillä (Bonisoli ja muut, 2021; Shilaskar ja muut, 2024; Tomar ja Gupta, 2023). Rikoksesta kertovan uutisartikkelin sekä itse rikoksen rikosilmoituksessa voi kuitenkin olla paljon eroavaisuuksia. Esimerkiksi käytetty sanasto, esitetyt yksityiskohdat, tai muun rikostutkinnallisesti tärkeän tiedon muoto voi olla hyvin eriävä uutisen ja rikosilmoituksen välillä. Uutisessa kohderyhmänä on laaja lukijayleisö, ja rikosilmoituksessa sitä tutkiva viranomainen.

Suomen kielellä aihetta ei ole juurikaan julkisesti tutkittu rikosten luokittelun kontekstissa. Kuitenkin suomenkielisiä tutkimuksia, joissa käytetään NLP-menetelmiä sovellettaviin käyttökohteisiin, on saatavilla. Diplomityössään Mähönen (2013) käyttää NLP-menetelmiä ja luokittelualgoritmeja tekstin luokittelussa. Muita esimerkkejä NLP:n sovelluksista suomenkielisen tekstin luokitteluun on vahingon syyn ennustaminen vahingonkuvauksesta sekä uutisten luokittelu sen otsikon perusteella (Jokela, 2022; Mansikka, 2020). Näissä tutkielmissa käytetyt NLP-menetelmät eivät ole riippuvaisia käsiteltävän datan kontekstista, ja ovat sovellettavissa tutkielman tekniseen toteutukseen. Aikaisemmissa tutkielmissa keskitys on ollut lähinnä eri luokittelualgoritmien vertailussa, eikä esimerkiksi tutkimuksissa olla mitattu miten sama luokittelija suoriutuu käytettäessä eriäviä esikäsittelyn ja ominaisuudenpoiminnan menetelmiä.

3 Teoreettinen viitekehys

Tässä luvussa esitetään tutkielmalle keskeiset käsitteet ja teoriat. Luvussa perehdytään etenkin kokeellisessa osassa käytettyihin menetelmiin, jotta voidaan paremmin ymmärtää niiden toiminta ja toiminnalliset vaatimukset. Kahdessa ensimmäisessä alaluvussa tunnistetaan tutkielmassa käsiteltävän datan rakenne, ja pohjustetaan sille suoritettavan esikäsittelyn merkitys. Kolmannessa alaluvussa käsitellään valittuja NLP-menetelmiä, joita hyödynnetään luokittelualgoritmien koulutuksessa. Neljäs alaluku käsittelee valittuja SVM- ja logistisen regression luokittelijoita. Viimeisessä alaluvussa esitetään tarkemmin transformer-arkkitehtuuria, johon tutkielmassa käytetty esikoulutettu FinBERT-malli perustuu.

3.1 Rakenteeton data

Tietojenkäsittelyssä data voidaan jakaa kolmeen luokkaan sen rakenteen perusteella: rakenteellinen, semirakenteellinen ja rakenteeton data. Datan rakenne määrittää sen käyttömahdollisuudet sekä luo perustan tarvittaville toimenpiteille, joiden avulla dataa voidaan hyödyntää haluttuun käyttötarkoitukseen. Taulukossa 1 esitetään Mishran ja Misran (2017) käyttämät määritelmät ja käytännön esimerkit datan rakenneluokille.

Taulukko 1. Datan rakenteet (Mishra & Misra, 2017, s. 742–745).

Datan rakenne	***Ominaisuudet***	***Esimerkki***
Rakenteellinen	*Organisoitua dataa, jolla on selkeä rakenne. Suoraan käytettävää.*	*Relaatiotietokannan tietueet*
Semirakenteellinen	*Ei kiinteää rakennetta, mutta sisältää rakennetta jäsentäviä elementtejä.*	*XML, JSON, sähköposti*
Rakenteeton	*Ei määritettyä rakennetta. Vaatii tiedonlouhintaa.*	*Vapaamuotoinen teksti, kuvat, videot*

Taulukon 1 määritysten nojalla rikosilmoitus voidaan luokitella semirakenteelliseksi dataksi, sillä ilmoitus sisältää jäsenneltyä tietoa ilmoittajasta ja rikoksen ajankohdasta, mutta myös vapaamuotoista ihmisen tuottamaa tekstiä ilmoituksen selostusosiossa. Tutkielman tavoitteena on tutkia luonnollisen kielen käsittelyn soveltuvuutta luokittelemaan ilmoitus rikosnimikkeeseen sen selostuksen perusteella, joten tutkielmassa käsitellään pelkästään rakenteetonta dataa. Seuraavassa alaluvussa käsitellään rakenteettomalle datalle tehtävää esikäsittelyä.

3.2 Esikäsittely

Luonnollisen kielen käsittelyn yksi keskeisistä haasteista on syötteiden käsittely ennen kielellistä prosessointia. Koska NLP pyrkii käsittelemään luonnollista eli ihmisen tuottamaa kieltä, pitää syötteiden käsittelyssä ottaa huomioon käsiteltävän kielen kieliopilliset säännöt, jotta lauseiden semantiikka säilyy. Suomenkieliseen luonnollisen kielen käsittelyyn on saatavilla kattavasti resursseja. Erityisen hyödyllinen on suomalaisten yliopistojen, CSC:n ja Kotimaisten kielten keskuksen muodostaman FIN-CLARIN-konsortion ylläpitämä Kielipankki. Kielipankki tarjoaa runsaasti hyödyllistä aineistoa ja valmiita NLP työkaluja, kuten Turun yliopiston ylläpitämä suomen kielellä esikoulutetun FinBERT-mallin (Kielipankki, n.d).

Rajallisten resurssien takia tutkielmassa keskitytään vain NLP-menetelmien toimivuuden kannalta tärkeisiin, etenkin ominaisuuksien valintaan liittyviin esikäsittelyvaiheisiin. Tutkielmassa oletetaan, että käsiteltävä teksti on suomenkielistä, eikä sisällä kirjoitusvirheitä tai kieleen kuulumattomia merkintöjä. Rajauksien tarkoitus on keskittää tutkielma käsittelemään NLP-menetelmiä. Mähönen (2013, s. 40–44) perustee, että tietokone ei ymmärrä tekstiä samalla tavalla kuin ihminen, ja pelkästään mahdollisten kirjoitusvirheiden käsittely vaatisi huomattavasti lisätoimenpiteitä syötteiden esikäsittelyssä.

Datan esiprosessoinnilla pyritään yhtenäistämään syötteet poistamalla luokituksen kannalta epäolennaiset osat datasta, ja siten vähentämään datassa olevaa kohinaa (Mansikka, 2020, s. 8). Kohinan poistaminen parantaa tarkkuutta vähentämällä käsiteltävien alkioiden määrää. Kohinan poistaminen on merkittävässä asemassa tutkielmalle, sillä käytettävän datasetin rajallisuuden myötä on välttämätöntä, että malli keskittyy käsittelemään vain aidosti merkittäviä piirteitä.

3.2.1 Merkkimuunnokset ja tokenisaatio

Yksinkertaisin tapa yhtenäistää käsiteltävää dataa lienee merkkimuunnokset. Esikäsittelynä merkistöä voidaan yksinkertaistaa esimerkiksi muuttamalla kaikki kirjaimet pieniksi kirjaimiksi tai poistamalla välimerkit ja numerot. Mansikka (2020, s. 8) muistuttaa, että esikäsittelyssä tehdyt toimenpiteet voivat johtaa datassa olevien semanttisten merkityksen menettämiseen. Esikäsittelyä toteuttaessa on tärkeä tunnistaa datasta tarvittava tieto, sekä pohtia miten eri toimenpiteet vaikuttavat halutun tiedon eheyteen. Tässä tutkielmassa tutkitaan avainsanapohjaisen toteutuksen soveltuvuutta luokitteluun, joten esimerkiksi numerot ja välimerkit poistetaan osana esikäsittelyä. On kuitenkin tiedostettava, että esikäsitellyistä selostuksista ei voida myöhemmin enää poimia esimerkiksi kellonaikoja, tili- tai puhelinnumeroita.

Merkkimuunnosten lisäksi toinen esikäsittelyssä suoritettava toimenpide on tokenisointi (engl. *tokenization*). Mähösen (2013, s. 36) mukaan tokenisoinnissa käsiteltävästä tekstistä paikannetaan sanarajat, joiden avulla teksti voidaan jakaa pienempiin osiin. Tokenisaatio on esivaatimus monessa NLP-menetelmässä. Lisäksi se parantaa datan käsiteltävyyttä muissa esikäsittelyn vaiheissa. Jokelan (2022, s 11) mukaan tokenisaatio voi olla yksinkertainen toimenpide, jossa sanat erotellaan toisistaan tekstissä olevien välilyöntien perusteella. Sofistikoituneemmat menetelmät voivat ottaa huomioon myös muita seikkoja sanarajojen paikannuksessa. Myös tokenisaatiota tehdessä on muistettava, että valittu menetelmä voi vaikuttaa käsiteltävän tiedon eheyteen tai sisällön merkitykseen. Esimerkiksi kaksiosaiset käsitteet voivat menettää tai muuttaa merkitystä tokenisoidessa

välilyöntien perusteella, ja esimerkiksi *musta aukko* muuttuisi erikseen käsiteltäviksi tokeneiksi *musta* ja *aukko*.

3.2.2 Ominaisuuksien karsinta

Ominaisuuksien karsinnalla tai valinnalla tarkoitetaan menetelmiä, joilla käsiteltävää tekstiä yhtenäistetään entisestään kohinan vähentämiseksi (Mansikka, 2020, s.8). Karsinnalla vähennetään myös käsiteltävien merkkien määrää, johtaen parempaan suorituskykyyn. Merkkimuunnosten lisäksi esikäsittelyn aikana voidaan poistaa yleisesti esiintyvät sanat, joilla ei tilastollista merkittävyyttä. Näitä sulkusanoiksi (engl. *stop words*) kutsuttuja sanoja esiintyy usein jokaisessa dokumentissa sen luokasta riippumatta, tehden sanoista merkityksettömiä luokittelun kannalta. Mähösen (2013. s 44) mukaan sulkusanat voidaan poistaa hyödyntäen tilastollisia menetelmiä tai käyttämällä valmiita sulkusanalistoja, jotka sisältävät kielessä yleisimmin esiintyvät sanat. Suomen kielessä sulkusanojen esimerkkinä Mähönen käyttää Kotimaisten kielten keskuksen listaamia useimmin esiintyviä sanoja kuten: *on, ja, se, siksi ja tämä*.

Suomen kielessä esiintyy paljon eri sana- ja taivutusmuotoja, jotka aiheuttavat kohinaa käsiteltävässä datassa. Tietokone ei kykene esimerkiksi hahmottamaan, että sana *pankin* ja *pankkien* ovat taivutusmuotoja sanasta *pankki*. Manning (Mähönen, 2013, s. 14) esittää, että kirjoitusasuun liittyviä haasteita voidaan ratkaista hyödyntäen stemmausta tai lemmatisointia. Hyödyntäessä perusmuotoistamista eli lemmatisointia muutetaan jokainen käsiteltävässä datassa esiintyvä sana sen perusmuotoon. Esimerkiksi sanat *pankin* ja *pankkien* muutettaisiin perusmuotoon *pankki*. Stemmaus on suoraviivaisempi menetelmä, jossa sanasta poistetaan sen loppuosaan liitetty taivutusmuoto. Stemmausta käyttäessä karsittaisiin sana *pankin* muotoon *pan* sekä sana *pankkien* muotoon *pank*. Tässä tapauksessa lemmanisointi yhtenäistää ominaisuuksia enemmän verrattuna stemmaukseen. Menetelmien välillä valitessa kannattaa pohtia, halutaanko tiettyjen sanamuotojen merkitys säilyttää.

3.3 NLP-menetelmät

Tässä osiossa syvennytään tutkielmassa käytettyihin luonnollisen kielen käsittelyn menetelmiin ja niiden taustalla olevaan teoriaan. On huomioitava, että esikäsittelyosiossa esitettyjä toimenpiteitä kuten tokenisaatio, sulkusanojen poisto, stemmaus ja lemmatisointi voidaan myös mieltää NLP-menetelmiksi. Tässä osiossa käsitellään niitä prosesseja, joilla käsiteltävä luonnollinen kieli muunnetaan numeraaliseen muotoon, jota tietokoneet pystyvät käsittelemään. Menetelmien tavoitteena on säilyttää käsiteltävän luonnollisen kielen ominaispiirteitä vielä muunnoksen jälkeen.

3.3.1 TF-IDF

TF-IDF (engl. *Term Frequency-Inverse Document Frequence*) on menetelmä, joka korostaa sanojen tärkeyttä suhteessa niiden esiintymistiheyteen, muuntaen samalla luonnollisen kielen numeeriseen vektorimuotoon. TF-IDF koostuu kahdesta osasta: termifrekvenssistä (engl. *Term Frequency*, TF) eli esiintymistiheydestä, sekä käänteisestä dokumenttitiheydestä (engl. *Inverse Document Frequence*, IDF). Kaavassa 1 on esitetty, miten yksittäisen termin esiintymistiheys voidaan laskea dokumentista. Termi t_i on laskettavan termin esiintymiskertojen lukumäärä, ja termi d vuorostaan kaikkien termien lukumäärä dokumentissa.

$$TF(t_i, d) = \frac{t_i}{d}. \tag{1}$$

Kaavalla 1 voidaan esimerkiksi selvittää, että aiemmassa kappaleessa sanan *termi* esiintymistiheys on noin 0.034. Samalla korostuu myös esikäsittelyn rooli, sillä käytettäessä lemmatisointia sanan *termi* esiintymistiheys olisi noin 0.068.

Käänteisellä dokumenttiyhteydellä voidaan laskea jokaiselle termille IDF-arvo, joka kuvaa termin harvinaisuutta koko dokumenttikokoelmassa. Mitä harvemmin termi esiintyy käsiteltävissä dokumenteissa, sitä suurempi sen IDF-arvo. Kaavassa 2 on esitetty

matemaattinen yhtälö IDF-arvon laskemiseen, missä termi N on dokumenttien lukumäärä ja termi n_i on niiden dokumenttien määrä, joissa termi t_i esiintyy.

$$IDF(t_i) = \log\frac{N}{n_i}. \quad (2)$$

Robertson (2004) käsittelee julkaisussaan käänteisen dokumenttiyhteyden tehokkuutta tilastollisena menetelmänä. Hänen mukaansa IDF:n tehokkuus perustuu sen onnistuneeseen implementaatioon intuitiosta, jonka mukaan useassa dokumentissa monesti esiintyvä termi ei ole hyvä tilastollinen diskriminaattori. Mikäli dokumenteista ei esikäsittelyssä karsittaisi sulkusanoja, saisivat nämä sanat matalan IDF-arvon. Sulkusanojan karsimisen seurauksena ei tarvitse laskea IDF-arvoja termeille, jotka on tunnistettu etukäteen merkitsemättömiksi.

Yhdistämällä termifrekvenssi ja käänteinen dokumenttiyhteys saadaan menetelmä, jolla voidaan laskea termin tärkeys suhteessa sen esiintymistiheyteen. Yhdistetty TF-IDF laskukaava voidaan esittää yhtälöllä

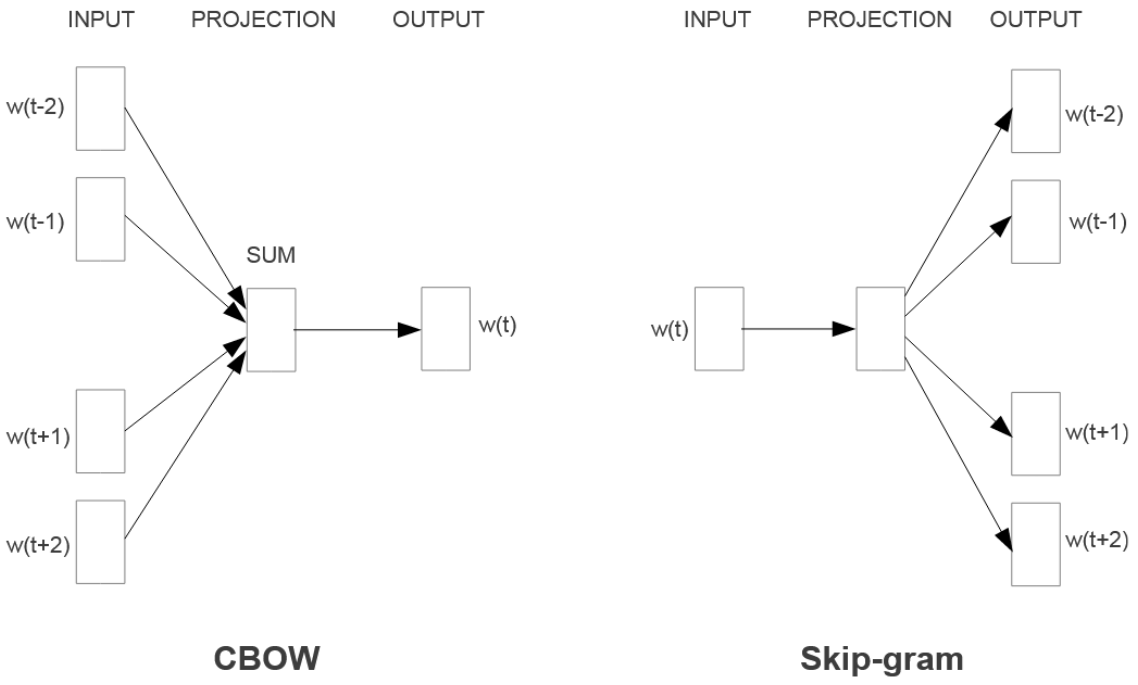
$$TFIDF(t_i, d, D) = \frac{t_i}{d} \times \log\frac{N}{n_i}. \quad (3)$$

Robertsonin (2004, s. 503) mukaan yhdistetty TF-IDF menetelmä on osoittautunut varsin tehokkaaksi ja vaikeasti voitettavaksi, jonka takia se on laajasti käytetty tiedonhaun ja luonnollisen kielen käsittelyn tekniikoissa. Menetelmä on valittu tutkielmaan sen avainsanapohjaisuuden takia. TF-IDF kykenee tunnistamaan dokumenttiluokille eli rikosnimikkeille tiettyjä merkityksellisiä sanoja, jonka perusteella luokittelu voidaan suorittaa.

3.3.2 Word embedding

Word embedding perustuu sanojen esittämiseen vektoreina moniulotteisessa vektoriavaruudessa siten, että syntaksisesti ja semanttisesti samankaltaiset sanat sijoittuvat

lähelle toisiaan (Mikolov ja muut, 2013). Menetelmän on havaittu kykenevän esittämään tehokkaasti semanttisesti toisilleen merkittäviä sanoja. Esimerkiksi kun koulutetusta mallista sanan *mies* vektorimuoto vähennettiin sanojen *kuningas* ja *nainen* vektoreiden summasta, saatiin tulokseksi vektoriesitys sanalle *kuningatar* (Mikolov ja muut, 2013, s. 2). Tutkielmassa käytetään Mikolovin ja muiden (2013) kehittämää Word2Vec-menetelmää. Word2Vec algoritmin toteutus perustuu yksinkertaiseen, yhden piilokerroksen sisältävään neuroverkkomalliin. Mallin toiminta riippuu käytettävästä arkkitehtuurimallista. Kuviossa 1 on esitetty Word2Vec:n kaksi arkkitehtuurimallia: Continuous Bag-of-Words (CBOW) ja Skip-gram.



Kuvio 1. Word2Vec-menetelmän arkkitehtuurimallit (Mikolov ja muut, 2013, s. 5).

Kuvion 1 vasemmalla esitetty CBOW-arkkitehtuuri perustuu kohdesanan ennustamiseen sille annetun kontekstin perusteella. CBOW-arkkitehtuuria käytettäessä kontekstisanat w(t±n) asetetaan vektoreinaan samaan projektioavaruuteen, jonka jälkeen niistä lasketaan keskiarvo (Mikolov ja muut, 2013, s. 4). Ennustettavaksi kohdesanaksi w(t) valitaan se mallin sanastosta löytyvä sana, jonka vektoriesitys vastaa parhaiten laskettua keskiarvovektoria. Mikolov ja muut (2013, s. 4) kutsuu esittelemäänsä mallia nimellä CBOW, sillä

toisin kuin perineinten bag-of-words-malli, se käyttää jatkuvaa, jakautunutta esitysmuotoa kontekstin esittämiseen. Käytännössä esitysmuodolla tarkoitetaan sanojen moniulotteisten vektorimuotojen esittämistä samassa projektioavaruudessa.

Kuvion 1 oikealla esitetyn Skip-gram-arkkitehtuurin toimita on käänteinen: malli pyrkii ennustamaan kontekstisanat annetun kohdesanan perusteella. Malli ottaa syötteeksi kohdesanan w(t) ja ennustaa sanoja w(t±n), joiden vektoriesitykset ovat projektioavaruudessa lähellä käsiteltävää kohdesanaa (Mikolov ja muut, 2013, s. 4–5).

Molemmissa arkkitehtuurimallien toiminnassa on omat vahvuutensa. CBOW-malli on toimintamallinsa takia nopeampi kouluttaa, sillä malli ennustaa yhden sanan lasketun keskiarvon perusteella, kun taas skip-gram laskee puolestaan usean ennustettavan kontekstisanan. Toisaalta skip-gram kykenee ennustamaan harvinaisempiakin sanoja, koska se oppii yksittäisten sanojen suhteet kontekstiin. Tutkielman kokeellisessa osassa Word2Vec ajetaan käyttäen skip-gram-arkkitehtuuria, koska se tarjoaa erilaisen lähestymistavan käsiteltävän luonnollisen kielen ymmärtämiseen. Yhdistämällä skip-gram-mallin vektoriesitykset TF-IDF arvoilla, voidaan huomioida termien tärkeys suhteessa kontekstiin. Täten Skip-gram mahdollistaa tärkeiden sanojen painottamisen TF-IDF-arvojen avulla paremmin kuin CBOW-arkkitehtuuri.

3.4 Luokittelualgoritmit

Aiemmassa alaluvussa esitetyt NLP-menetelmät eivät itsessään suorita luokittelua, vaan tuottavat käsiteltävästä tekstistä numeraalisia ominaispiirteitä. Jotta ominaispiirteiden perusteella voidaan suorittaa varsinainen luokittelu, tarvitaan erillinen luokittelualgoritmi. Tässä alaluvussa esitetään tutkielmassa käytetyt luokittelualgoritmit. Tutkielmaan valitut luokittelualgoritmit ovat diskriminatiivisia malleja, joiden toiminta perustuu luokkien erottamiseen toisistaan tiettyjen piirteiden perusteella (Jurafsky ja Martin, 2024, s. 77–78). Mallit koulutetaan valvotusti erillisen opetusaineiston avulla. Näiden mallien

toiminta tukee tutkielman tavoitetta luokitella selostuksia rajattuihin rikosnimikkeisiin niiden sisällön perusteella.

3.4.1 Logistinen regressio (LogReg)

Logistinen regressio on valvotun koneoppimisen perusmalli luonnollisen kielen luokittelutehtävissä (Jurafsky ja Martin, 2024, s. 77). Logistista regressiota käytetään pääosin havainnon luokitteluun kahden luokan välillä, mutta se soveltuu myös moniluokkaiseen luokitteluun. Moniluokkaisessa luokittelussa logistinen regressio kuitenkin olettaa, että havainto voi kuulua vain yhteen luokkaan kerrallaan.

Logistinen regressio oppii määrittämään sille annetun koulutusdatan perusteella kullekin syötepiirteelle x sen tärkeyttä merkitsevän painovektorin w, sekä vakiotermiksi kutsutun bias-termin b. Kaavassa 4 on esitetty logistisen regression toimintaa kuvaava pistetulomerkintä, missä z on syötteiden painotettu summa.

$$z = w \times x + b. \tag{4}$$

Painovektori w voi olla positiivinen tai negatiivinen riippuen siitä, onko syötepiirre x luokittelupäätöksen kannalta merkittävä negatiiviselle tai positiiviselle luokalle. Kaavalla 4 laskettu painotettu summa ei ole todennäköisyysteorian mukainen, sillä tulos voi olla välin [0, 1] ulkopuolella oleva negatiivinen tai positiivinen reaaliluku. Painotetun arvon muuttamiseksi todennäköisyydeksi $P(y_k = 1|x)$ tarvitaan sigmoid- tai softmax-funktio (Jokela, 2022, s. 14). Sigmoid soveltuu käytettäväksi vain kaksiluokkaisessa luokittelussa, joten tutkielmassa käytetään softmax-funktiota

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}},\ 1 \leq i \leq K, \tag{5}$$

missä z_i on käsiteltävän luokan painotettu arvo ja z_j kaikkien muiden luokkien painotettu arvo. Softmax-funktio ottaa syötteeksi vektorin z = $[z_1, z_2, \dots, z_K]$, jossa K on luokkien määrä, ja muuntaa jokaisen syötteen arvon välille [0, 1] siten, että niiden kokonaissumma on 1 (Jurafsky ja Martin, 2024, s. 85).

Lisäksi tarvitaan luokittelijan entropiaa eli epävarmuutta tai -järjestystä ilmaisevaa häviöfunktiota, jonka avulla luokittelija optimoidaan suosimaan tarkkoja ja oikeita ennusteita. Epävarmuus lasketaan suhteuttamalla luokkien välistä todennäköisyysjakaumaa, eli ristientropiaa. Tutkielmassa yksittäisen näytteen ristientropia $L_{log}(y, p)$ saadaan laskettua yhtälöllä

$$L_{log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p)), \quad y \in \{0, 1\}, \tag{6}$$

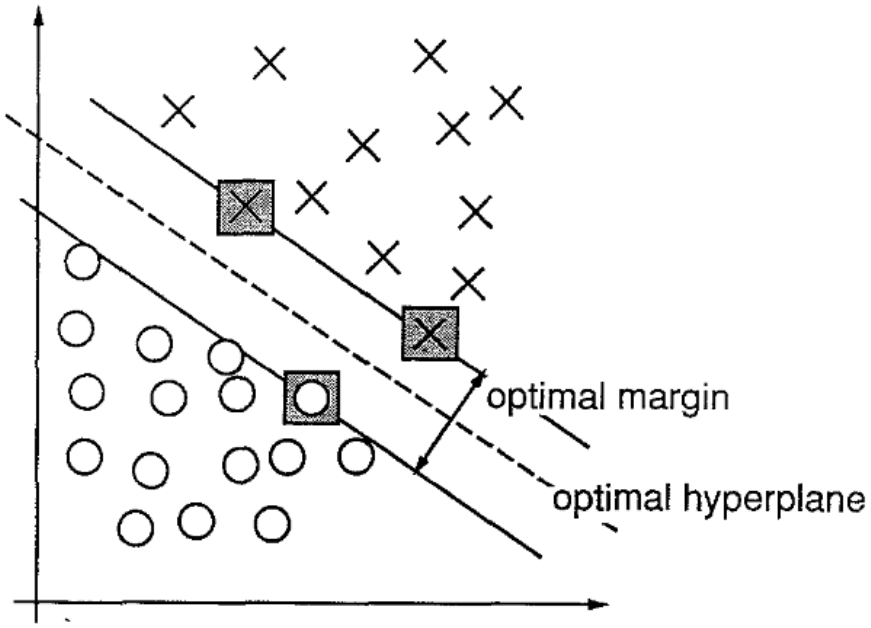
missä y on näytteen todellinen luokka ja p on softmax-funktiolla ennustettu todennäköisyys (Scikit-learn, 2024, Luku 3.4.2.12).

Logistinen regressio on valittu tutkielmaan koska se on NLP-luokittelun perusmalli, ja siksi hyvä lähtökohta luokittelulle. Jurafskyn ja Martinin (2024, s. 84) mukaan logistinen regressio suoriutuu hyvin tilanteissa, jossa luokkien välillä on paljon samankaltaisuuksia vrt. varkaus ja näpistys.

3.4.2 Tukivektorikone (SVM)

Vuonna 1995 Vapnik ja Cortes (1995, s. 274–275) esittelivät uudenlaisen tukivektoriverkostoihin perustuvan luokitinmallin, joka projisoi syötteet epälineaarisen kuvauksen avulla vektoriavaruuteen, ja pyrkii sitten muodostamaan lineaarisen päätöspinnan syötteiden jakamiseksi luokkiin. Vapnikin ja Cortesin (1995, s. 274) mukaan toimintaperiaate varmistaa mallin tehokkaan yleistyskyvyn eli kyvyn luokitella ennalta tuntemattomia näytteitä. Alla olevassa kuviossa 2 hahmotellaan, kuinka tukivektorikone muodostaa syötteiden erottavan lineaarisen päätöspinnan kaksiulotteisessa avaruudessa.

Päätöspintaa muodostaessa SVM pyrkii maksimoimaan luokkien välisen marginaalin käyttäen marginaalitasoja. Marginaalitasot kulkevat päätöspintaa lähimpinä olevien syötteiden, eli tukivektorien kautta.



Kuvio 2. Esimerkki SVM:n päätöspinnasta (Vapnik & Cortes, 2024, s. 275).

SVM on logistisen regression lailla suunniteltu alun perin kaksiluokkaiseen luokitteluun, mutta malli kykenee moniluokkaiseen luokitteluun käyttäessä one-versus-one (OvO) tai one-vs-rest (OvR) implimentaatiota (Scikit-learn, 2024, Luku 1.4.1.1). OvO-toteutuksessa jokaiselle luokkien parille luodaan oma luokitin, ja lopullinen luokka määritetään sen perusteella, mikä luokka saa eniten luokituksia. Jos esimerkiksi käytössä on kolme luokkaa, koulutettaisiin kolme luokitinta pareille AB, AC ja BC. OvR-toteutuksessa jokaiselle luokalle koulutetaan luokitin, joka erottaa kaikki kyseisen luokan näytteet muista luokista. Näyte luokitellaan siihen luokkaan, joka saa korkeimman todennäköisyyden. Tutkielmassa käytetään OvR-toteutusta, sillä se on suorituskyvyltään ja skaalautuvuudeltaan parempi etenkin luokkien määrän kasvaessa.

Tukivektorikoneen primaaliongelmaa eli päätöstason optimointia voidaan esittää rajoitukset ja määritelmät huomioon ottaen yhtälöllä

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i, \tag{7}$$
$$y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,\ \zeta_i \geq 0,\ i = 1, \ldots, n,\ x_i \in \mathbb{R}^p, y \in \{1, -1\}^n,$$

missä w on päätöstason suuntaa määrittävä painovektori ja w^T painovektorin transpoosi. Muuttuja ζ_i kuvaa, kuinka paljon näytteet voivat olla marginaalin sisäpuolella. Parametri C säätelee marginaalin maksimoinnin ja luokitteluvirheen minimoinnin välistä tasapainoa. y_i on näytteen luokkamerkintä, ja $\phi(x_i)$ ominaisuusmuutosfunktio, joka muuntaa syöteavaruuden korkeaulotteiseksi ominaisuusavaruudeksi, jossa lineaarinen erotus voidaan tehdä (Scikit-learn, 2024, Luku 1.4.7). Tämä optimointifunktio on suoraviivainen ja laskennallisesti tehokas.

Primaariongelmasta johdettu duaaliongelma on esitetty kaavassa 8, missä α on Lagrangen kertoimien vektori, jossa jokainen α_i vastaa näytettä (x_i, y_i), ja missä e on ykkösmatriisi (Scikit-learn, 2024, Luku 1.4.7).

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha,\ \ y^T \alpha = 0, 0 \leq \alpha_i \leq C, i = 1, \ldots, n. \tag{8}$$

Lisäksi kaavassa 8 käytetty matriisi Q voidaan määrittää seuraavasti:

$$Q_{ij} = y_i y_j K(x_i x_j),\ K(x_i x_j) = \phi(x_i)^T \phi(x_j), \tag{9}$$

missä K on kerneli (Scikit-learn, 2024, Luku 1.4.7). Kerneli mahdollistaa päätöspinnan löytämisen myös epälineaarisilla muunnoksilla ilman, että tarvitaan korkeampiulotteisempaan ominaisuusavaruuteen siirtymistä. Mikäli käytetyn tukivektorikoneen valittu kerneli on epälineaarinen, käyttää malli duaalifunktiota.

Päätöstason optimoinnin jälkeen tukivektorikoneella voidaan suorittaa näytteen x luokitus yhtälöllä

$$f(x) = \sum_{i \in SV} y_i \alpha_i K(x_i, x) + b, \quad (10)$$

missä SV tarkoittaa tukivektoreita ja kerneli K laskee samankaltaisuuden tukivektorin x_i ja näytteen x välillä. Termi b on poikkeama, joka siirtää päätöspintaa sen alkuperäisestä pisteestä.

Kokonaisuudessaan tukivektorikoneiden luokittelun tarkkuutta ja soveltuvuutta on tutkittu paljon viimeisen kolmen vuosikymmenen aikana (Mansikka, 2020; Tomar ja Gupta, 2023; Shilaskar ja muut, 2024; Joachims, 1998). Tukivektorikoneet suoriutuvat monipuolisista luokittelutehtävistä säilyttäen tarkkuuden, ja on siksi valittu tutkielmaan.

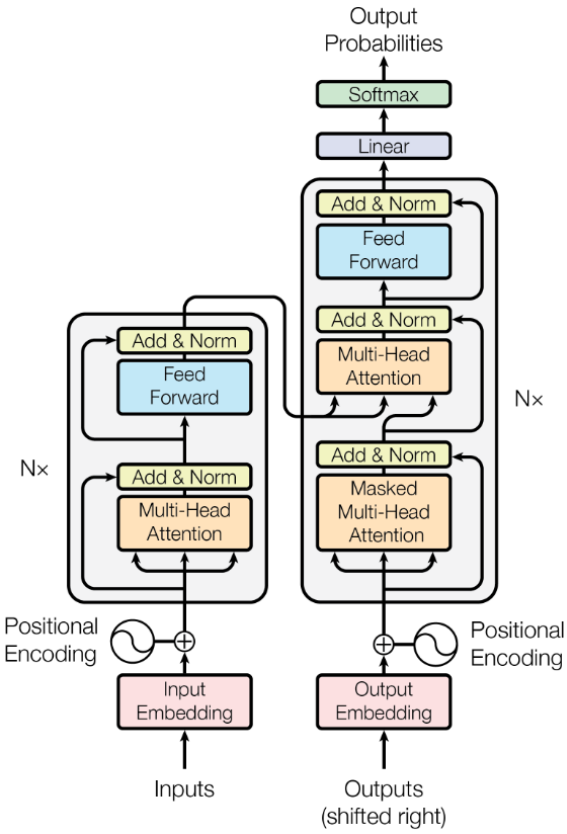
3.5 Transformer-arkkitehtuuri

Kuten aiemmin tutkielmassa on mainittu, yksi luonnollisen kielen käsittelyn kehityksen viimeisimmistä kehityksistä on syväoppimisen transformer-arkkitehtuuri. Tässä kappaleessa esitellään transformer-arkkitehtuurin peruskonsepti, jotta voidaan ymmärtää tutkielmassa käytetyn FinBERT-mallin toimintaa. Koska tutkielmassa käsitellään luonnollista kieltä, oletetaan, että transformer-mallin syöte on vapaamuotoista tekstiä.

Transformer-arkkitehtuurin toiminta perustuu itsehuomiomekanismiin, jonka avulla malli kykenee huomioimaan jokaisen sanan merkityksen suhteessa kaikkiin muihin sekvenssissä esiintyviin sanoihin (Vaswani ja muut, 2017, s. 5). Toimintaperiaatteen ansiosta malli kykenee oppimaan kontekstin ja semantiikan merkitykset tehokkaasti. Transformer-arkkitehtuurissa itsehuomiomekanismin toimintaa on tehostettu implementoimalla monipäinen itsehuomiointirakenne, joka mahdollistaa monen itsehuomiomekanismin samanaikaisen ajon rinnakkaisesti.

Alla olevassa kuviossa 3 on esitetty transformer-arkkitehtuuri. Vaswanin ja muiden (2017) esittämä malli voidaan jakaa kahteen osaan: N kerrosta sisältävään enkooderiin ja dekooderiin. Enkooderien ja dekooderien määrä riippuu käyttökohteesta. Yksittäinen

enkooderin kerros koostuu kahdesta osakerroksesta, joista ensimmäinen sisältää monipäisen itsehuomiomekanismin. Ensimmäisen osakerroksen tuottamat vektorit käsittelee toisen osakerroksen syöttökerros, joka pyrkii oppimaan piirteitä syötteestä. Molempien vaiheiden jälkeen ulostulot yhdistetään ja normalisoidaan alkuperäiseen syötteeseen.



Kuvio 3. Transformer-arkkitehtuuri (Vaswani ja muut, 2017, s. 3).

Kuvion 3 oikealla puolella esitetty dekooderi sisältää kolmannen osakerroksen, jossa käytetään maskattua itsehuomiomekanismia. Kerroksen alussa oleva maskattu osakerros estää dekooderia ottamasta huomioon sekvenssissä käsiteltävän sanan jälkeen esiintyviä sanoja, jolloin ulostulo on riippuvainen vain ennen käsiteltävää sanaa esiintyvistä sanoista (Vaswani ja muut, 2017, s. 3). Enkooderit vastaavat syötteen analysoinnista, ja dekooderit muuntavat analysoituja piirteitä uudeksi sekvenssiksi.

Koska transformer-arkkitehtuurissa syötteitä ei käsitellä järjestyksessä vaan rinnan, tulee syötteeseen lisätä paikkakoodaukset ennen enkoodausta tai dekoodausta, jotta malli hahmottaa ja oppii sanojen järjestyksen merkityksen. Kuvion 3 mukaisesti lopussa suoritetaan enkooderin tai dekooderin tuottamat piirteet ennustettavaksi tulokseksi (Linear) sekä muutetaan tulos todennäköisyysjakaumaksi (Softmax).

3.5.1 FinBERT

Virtanen ja muut (2019) esittelevät julkaisussaan suomen kielellä esikoulutettua versiota BERT-mallista nimeltään FinBERT. Luokitellessa suomenkielisiä uutisia kymmeneen eri luokkaan, FinBERT suoriutui paremmin kuin monikielinen M-BERT riippumatta koulutusdatasetin koosta (Virtanen ja muut, 2019, s.9). FinBERT ei teknisesti eroa aikaisemmin vuonna 2019 Devlinin ja muiden (2019) esittelemästä transformer-arkkitehtuuriin perustuvasta BERT-mallista, vaan erona on mallin esikoulutukseen käytetyn materiaalin kieli. Tässä tutkielmassa ei perehdytä tarkemmin Virtasen ja muiden (2019) toteuttamaan FinBERT-mallin esikoulutukseen.

BERT on diskriminatiivinen malli, jota voidaan käyttää luonnollisen kielen käsittelyn tekstin syvällistä ymmärtämistä vaativissa tehtävissä. Malli keskittyy sille syötetyn sisällön mallintamiseen ja analysointiin, eikä siksi tarvitse transformer-arkkitehtuurissa esitettyjä sisältöä tuottavia dekoodereita. FinBERT perustuu BERT-mallin perusarkkitehtuuriin, joka koostuu 12 enkooderikerroksesta (Virtanen ja muut, 2019, s.7). Devlinin ja muiden (2019, s. 1) esittelemässä BERT-mallissa käytetään kaksisuuntaista itsehuomiomekanismia, joka ottaa huomioon sekvenssissä käsiteltävän sanan ennen ja jälkeen esiintyvät sanat.

FinBERT on valittu tutkielmaan, sillä toisin kun tutkielman muut mallit, se ei ole esikoulutuksen takia sanaston muodostamisessa riippuvainen tutkielman koulutusdatasta. FinBERT tuo tutkielman vertailuasetelmaan myös uudemman transformer-pohjaisen toteutuksen.

4 Kokeellinen osa

Tutkielman kokeellisessa osassa on esitelty kandidaatintutkielman osana toteutetut NLP-luokittelijamallit. Ensimmäisessä alaluvussa on esitelty tarkemmin mallien kouluttamisessa, validoinnissa ja vertailussa käytetty aineisto. Aineiston esittelyn yhteydessä jäsennetään kokeellisen osan rakenne. Toisessa alaluvussa käsitellään käytettyjen mallien toteutuksia sekä niiden välisiä eroavaisuuksia. Viimeisessä alaluvussa esitetään kunkin toteutetun mallin validointivaiheen jälkeiset tarkkuusarvot, jolloin varsinaisia vertailuasetelman tuloksia analysoidessa voidaan vertailla mallien tarkkuuksien muutosta, ja siten arvioida yleistyskykyä.

4.1 Käytetty aineisto

Tässä alaluvussa käsitellään kokeellisessa osassa käytettyä aineistoa, joka jakautuu käyttötarkoituksen mukaan koulutus-, validointi tai vertailudataksi. Kunkin datasetin sisältö, käyttötarkoitus ja merkitys tutkielmalle on esitetty sille varatussa alaluvussa.

Tutkielman aineisto muodostuu näennäisistä selostuksista, joissa kuvataan kuvitteellisia varkaus-, näpistys- tai petosrikoksen tapahtumasarjoja. Jokaiselle selostukselle on määritetty, mihin rikosnimikkeeseen selostus kuuluu. Näiden selostus-nimike-tietueiden luonti- ja keräysprosessi on selostettu seuraavassa osiossa.

4.1.1 Koulutusdata

Nimensä mukaisesti koulutusdataa käytetään NLP-mallien kouluttamisvaiheessa, joka on esivaatimuksena luokittelun suorittamiselle. Kouluttamisvaiheessa malli muuntaa koulutusdatan numeraalisiksi ominaisuuksiksi, joiden perusteella se oppii luokille ominaiset ja luokittelupäätöksen kannalta tärkeät piirteet. Syötteen muuntaminen ominaisuuksiksi tapahtuu aiemmin esiteltyjen NLP-menetelmien avulla, ja itse luokittelulogiikan ja -

päätöksen oppii tekemään valittu luokittelualgoritmi. Varsinaisesti koulutusvaiheessa keskitys on luokittelijan kouluttamisessa valittujen NLP-menetelmien avulla.

Tutkielmassa käytetty koulutusdata on tuotettu hyödyntäen seuraavia menetelmiä:

1. Itsenäisesti tuotetut näennäisselostukset.
2. Laajan kielimallin avulla tuotetut näennäisselostukset, joita on tarvittaessa muokattu.
3. Anonyymilla kyselytutkimuksella kerätyt näennäisselostukset.

Menetelmät 2 ja 3 valittiin tutkielmaan nopeuttamaan ja monipuolistamaan tuotettavaa koulutuskorpusta. Laajan kielimallin avulla tuotettuja selostuksia on tarvittaessa muokattu, mikäli niissä esiintyy loogisia epäjohdonmukaisuuksia tai muuta rikosilmoituksen selostuksen kontekstista poikkeavaa. Esimerkiksi, jos kielimallin tuottama rikosselostus sisälsi lopussa maininnan siitä, että rikollinen on saatu kiinni ja tuomittu tehdystä rikoksesta, on tämä osa jätetty pois.

Menetelmä 3 on valittu tutkielmaan vahvistamaan koulutusdatan kieliasullista monipuolisuutta ja -äänisyyttä. Kyselytutkimuksessa vastaajaa pyydettiin kirjoittamaan näennäinen selostus valitsemastaan rikoksesta. Toteutetun kyselytutkimuksen ohjeistus on esitetty liitteessä 1. Kyselyn avulla saatiin kerättyä ihmisten käyttämiä erilaisia kirjoitustyylejä ja sanastoa. Monipuolisella koulutusdatalla saadaan laajennettua mallin sanastoa, jolloin mallin kyky löytää luokittelupäätökselle merkittäviä ominaisuuksia on parempi.

Selostuksien tuottamiseen on menetelmästä riippumatta käytetty samaa ohjeistusta. Kukin koulutuskorpukseen kuuluva tietue on tuotettu seuraavien ohjeiden ja määritysten puitteissa:

1. Valitse, mihin tutkielmassa käytettyyn rikosnimikkeeseen selostus kuuluu.
2. Kuvittele tai keksi rikosnimikkeeseen sopiva tapahtumasarja, ja kirjoita siitä selostus ikään kuin tekisit rikosilmoitusta.

3. Päätä, mitä tietoa pystyt ilmoituksessa antamaan. Selostusta kirjoittaessa voi ”vahingossa” unohtaa jotain oleellisia asioita, joita ei itselle tulisi mieleen tai joita on noloa paljastaa. Rikos on kuitenkin varmasti tapahtunut.

Muodostettavan sanaston monipuolisuuden varmistamiseksi anonyymissa kyselytutkimuksessa vastaajia on pyydetty kirjoittamaan laatimansa selostus toisen kerran käyttäen eri ilmaisumuotoja ja sanastoa. Myös kielimallin avulla, että itse tuotettujen selostusten osalta on säännöllisesti vaihdettu kirjoitustyyliä, sanastoa ja selostuksessa käytettävää asiajärjestystä. Taulukossa 2 on esitetty tutkielmassa käytetyn koulutusdatasetin koostumus.

Taulukko 2. Koulutusdatasetin kokoonpano.

Menetelmä	***Petos***	***Näpistys***	***Varkaus***
Itse luotu	22	20	19
Kielimalli	21	20	19
Kysely	7	10	12
Yhteensä	**50**	**50**	**50**

Koulutusdatasetissä on pyritty tasapainottamaan itse luotujen ja kielimallin luomien selostusten määrää, jotta korpus on mahdollisimman monipuolinen, looginen ja realistinen. Settiä on vahvistettu kyselytutkimuksella kerätyillä selostuksilla. Settiä ei voitu tasata kaikkien kolmen menetelmän kesken, johtuen kyselytutkimuksen pienestä jakelusta sekä rajallisesta kuukauden pituisesta vastausikkunasta. Datasetti on tasapainotettu nimikkeiden mukaan, sillä epätasainen jakauma koulutusdatassa voi johtaa enemmän edustetun luokan ylisovittamiseen ja toisen luokan alisovittamiseen. Tasapainotuksella pyritään varmistamaan, että malli oppii erottamaan eri luokat toisistaan tasapuolisesti. Tasapainotuksen takia tutkielmaan ei ole myöskään otettu esimerkiksi neljättä *muu*-luokkaa, joka vaatisi kattavan määrän dataa kaikista muista rikosnimikkeistä.

4.1.2 Validointidata

Validointidata on osa alkuperäistä koulutuskorpusta, ja tuotettu samoin menetelmin kuin koulutusdata. Validointidatasetti muodostetaan erottamalla erillinen otos koulutuskorpuksesta. Validointidatan tarkoitus on arvioida mallin suoritus- ja yleistämiskykyä uusille syötteille, jolloin varmistetaan, ettei malli mukaudu liiallisesti sille annettuun koulutusdataan eli ylisovitu (engl. *overfitting*). Ylisovittumisen seurauksena malli olisi hyvin tarkka luokittelemaan juuri koulutusdatan piirteitä, mutta sen yleistyskyky luokitella uutta aineistoa olisi huono.

Tutkielmassa koulutusdatasta on eritelty 30 % otos, joka muodostaa validointidatasetin. Validointidataksi eriteltyä dataa ei käytetä mallin koulutuksessa, vaan se toimii erillisenä testinä arvioidakseen koulutusvaiheen aikana saavutettua suorituskykyä. Mallin validoinnilla varmistetaan, että malli on aidosti toimintakykyinen ennen lopulliseen vertailuvaiheeseen siirtymistä.

4.1.3 Vertailudata

Vertailudatalla tarkoitetaan niitä tuotettuja selostuksia, joita käytetään arvioidessa toteutettujen NLP-mallien lopullista suorituskykyä. Datasettiin kuuluvia selostuksia ei käytetä koulutusvaiheessa, koska tarkoituksena on mitata toteutettujen mallien yleistämiskykyä täysin uusilla syötteillä.

Kun arvioidaan soveltuvuutta rikosilmoitusten luokitteluun, on oleellista varmistaa vertailudatan oikeellisuus ja yhdenmukaisuus viranomaisten käytänteiden kanssa. Vertailudatan luokitteli manuaalisesti kolme Suomen poliisissa työskentelevää viranomaista. Lainvalvontaviranomaisen asiantuntemuksen hyödyntämiseksi tutkielmalle haettiin erillinen tutkimuslupa keskusrikospoliisilta. Manuaalinen luokittelu toteutettiin sähköisen kyselytutkimuksen avulla. Lainvalvontaviranomaisilta kerätyt luokitukset toimivat

vertailukohtana lopullisessa luokittelussa, jossa arvioidaan toteutettujen NLP-mallien soveltuvuutta rikosilmoitusten luokitteluun.

Vertailudatasetin luomisessa käytettiin tarkempaa liitteessä 2 esitettyihin määrityksiin perustuvaa kuratointia. Kuratoimalla vertailudataa ennalta määritettyihin asetelmiin voidaan luokittelua arvioida monipuolisesti pitäen samalla datamassa hallittavana. Lisäksi kuratointia tarvittiin, jotta vertailudata pysyisi sekä tutkielman että lainvalvontaviranomaisten rajallisten resurssien puitteissa. Alla olevassa taulukossa 3 on esitetty kuratoidun vertailudatasetin koostumus, joka koostuu 44 selostuksesta. Taulukossa 3 esitetyt, ja tutkielmassa käytetyt vertailudatan laatumääritelmät ja selosteet ovat esitetty tarkemmin liitteessä 2.

Taulukko 3. Vertailudatasetin koostumus.

Laatu	***Synonyymi***	***Erilaiset***	***Samanlaiset***
Suppea	6 kpl	6 kpl	3 kpl
Normaali	6 kpl	5 kpl	3 kpl
Kattava	6 kpl	6 kpl	3 kpl
Yhteensä	**18 kpl**	**17 kpl**	**9 kpl**

Viranomaisille luokiteltavaksi lähetetty setti sisälsi 45 ilmoitusta, joista 44 oli käyttökelpoisia tutkielmaan valittujen nimikkeiden perusteella. Valmis datasetti sisältää 15 petos- ja näpistysrikoksen selostusta sekä 14 varkausnimikkeen selostusta. Yksi 15:stä lähetetystä varkausselostuksista sarjoittui tutkielman ulkopuoliseen moottorikulkuneuvon käyttövarkausnimikkeeseen, jonka takia se jätettiin pois. Vertailudatasetin minimaalinen epätasapainoisuus ei ole merkittävä, koska settiä ei käytetä mallin kouluttamiseen vaan sen yleistyskyvyn testaamiseen.

4.2 Esikäsittelyn toteutus

Tutkielmassa käytetty data tuotiin ohjelmointialustalle käsiteltäväksi kahtena listana: selostus- ja nimikelistana. Selostuksen ja rikosnimikkeen välinen yhteys säilytettiin listan indekseillä siten, että esimerkiksi indeksissä yksi olevan selostuksen nimike sijaitsi vastaavasti nimikelistan indeksissä yksi. Kokeellisen osan esikäsittelyvaiheen merkkimuunnokset toteutettiin Python-ohjelmointikielen perusominaisuuksin.

Syötteiden tokenisoinnissa, sulkusanojen poistamisessa, sekä stemmauksessa käytettiin Natural Language Toolkit (NLTK) -moduulin valmiita kirjastoja nltk.tokenize, nltk.corpus ja nltk.stem (NLTK, 2024). NLTK-moduuli soveltuu käytettäväksi tutkielmaan koska se on aktiivisesti ylläpidetty, helppokäyttöinen, ja tukee suomen kieltä. Kirjastot tarjoavat valmiin sulkusanalistan ja suomen kielelle konfiguroidun käyttövalmiin stemmerin. Toteutettujen merkkimuunnosten, sekä NLTK-kirjastojen tokenisoinnin ja stemmauksen toiminta on havainnollistettu taulukossa 4.

Taulukko 4. Esikäsittelyn vaikutus satunnaisesti valittuun syötteeseen.

Tehty toimenpide	**Teksti**
Alkuperäinen syöte	*Huomasin tilitapahtumissani useita luvattomia korttiveloituksia, yhteensä 800 euroa. En ole antanut tietojani kenellekään enkä tehnyt kyseisiä ostoksia*.
Merkkimuunnokset Tokenisointi	*['huomasin', 'tilitapahtumissani', 'useita', 'luvattomia', 'korttiveloituksia', 'yhteensä', 'euroa', 'antanut', 'tietojani', 'kenellekään', 'enkä', 'tehnyt', 'kyseisiä', 'ostoksia']*
Merkkimuunnokset Tokenisointi Stemmaus	*['huomas', 'tilitapahtum', 'use', 'luvattom', 'korttiveloituks', 'yht', 'euro', 'antanu', 'tieto', 'kene', 'enk', 'tehny', 'kyseis', 'ostoks']*
Merkkimuunnokset Tokenisointi Lemmatisointi	*['huomata', 'tili#tapahtuminen', 'usea', 'luvaton', 'kortti#veloitus', 'yhteensä', 'euro', 'antaa', 'tieto', 'kukaan', 'ei', 'tehdä', 'kyseinen', 'ostos']*

Suomen kielen lemmatisointiin käytettiin trankit-kirjastoa, jota TurkuNLP-ryhmä suosittelee oman vanhemman projektinsa sijaan. Yhä ylläpidetyn Trankit-kirjaston suomenkielinen malli on koulutettu samalla korpuksella kuin elinkaarensa päätyyn tullut TurkuNLP:n malli (TurkuNLP, 2024). Lemmatisoinnin toiminta on kuvattu taulukossa 4.

On tärkeä muistaa, että syötteiden esikäsittelyssä käytetään joko stemmausta tai lemmatisointia. Tutkielmaa varten on luotu erilliset stemmausta ja lemmatisointia käyttävät mallit, jotta valitun menetelmän tehokkuutta voidaan arvioida.

4.3 NLP-menetelmien toteutus

Tässä alaluvussa esitetään tutkielman koeosassa käytettyjen TF-IDF- ja word embedding -menetelmien toteutus. Nämä teoreettisessa viitekehyksessä tarkemmin esitetyt NLP-menetelmät muuttavat syötteet numeraalisiksi ominaisuuksiksi, joiden perusteella koulutettavat NLP-mallit oppivat tekemään luokittelun.

4.3.1 TF-IDF

Toteutuksessa esikäsiteltyjen selostusten TF-IDF-arvon laskentaan käytettiin valmista *TfidfVectorizer*-toteutusta, joka kuuluu osaksi laajasti käytettyä Scikit-learn-kirjastoa. *TfidfVectorizer* ajettiin oletusparametrein, pois lukien toteutuksen mukana tulleita esikäsittelytoimintoja, joille ei ollut jo tehdyn esikäsittelyn takia tarvetta.

Alapuolella olevassa taulukossa 5 on esitetty lemmatisoidun satunnaisotoksen TF-IDF-arvojen perusteella kullekin käytetylle luokalle viisi merkittävintä sanaa. Taulukosta 5 nähdään, että missään luokassa ei esiinny keskenään samoja sanoja. Tulos viittaa siihen, että luokkien välillä on löydettävissä eroavia ominaisuuksia, mikä on otollista luokittelun kannalta.

Taulukko 5. TF-IDF: lemmatisoidun otoksen viisi merkittävintä sanaa luokittain.

Petos		**Näpistys**		**Varkaus**	
Sana	**TF-IDF**	**Sana**	**TF-IDF**	**Sana**	**TF-IDF**
pankki	0.1397	tasku	0.0869	pyörä	0.1229
Verkko	0.1137	huomata	0.0842	työ	0.0982
tili	0.0935	lompakko	0.0765	auto	0.0892
tehdä	0.0893	kadota	0.0746	talli	0.0839
saada	0.0805	matka	0.0740	kalu	0.0729

Taulukosta 5 nähdään, että nimikkeen varkaus merkittävin sana on *pyörä*, mikä on todennäköisesti anastettu esine selostuksessa. Luokittelupäätöstä tehdessä ilmoituksessa esiintyessä sana *pyörä*, on sanan korkean merkittävyyden takia todennäköistä, että selostus luokitellaan varkaudeksi. Toisaalta, jos varkausrikoksen selostuksessa ei esiinny sanaa pyörä, täytyy selostuksesta löytyä muita luokalle merkittäviä sanoja.

4.3.2 Word2Vec

Word embedding -menetelmänä kokeellisessa osassa käytettiin Mikolovin ja muiden (2013) kehittämää Word2Vec-mallia. Word2Vec-malli on ajettu taulukon 6 mukaisilla parametreillä. Arvot parametreihin *vector_size, window* ja *min_count* on valikoitu suhteessa käytettävän datasetin pieneen kokoon. Kokonaisfrekvenssin minimiarvoksi on valittu yksi, koska esikäsittelyssä on poistettu riittävästi kohinaa, ja halutaan ottaa huomioon jokainen esiintyvä sana. Parametri *workers* on valittu sopivaksi suhteessa kokeellisessa osassa käytetyn tietokoneen suorituskykyyn. Skip-gram-arkkitehtuurin valinta ja sopivuus on perusteltu tutkielman teoreettisessa viitekehyksessä. Lemmatisoidulle ja stemmatulle syötteille luodaan erilliset mallit, jotta mallin sanasto on yhteensopiva käytettävien sanamuotojen kanssa.

Taulukko 6. Käytetyn Word2Vec-mallin konfiguraatio.

Parametri	***Arvo***	***Kuvaus***
vector_size	100	Sanavektoreiden ulottuvuuksien määrä.
window	5	Maksimietäisyys nykyisen ja ennustetun sanan välillä lauseessa.
min_count	1	Kokonaisfrekvenssin minimiarvo, jotta sana otetaan huomioon.
workers	4	Mallin kouluttamiseen käytettyjen säikeiden määrä.
sg	1	Käytettävä arkkitehtuuri: 0 = CBOW, 1 = Skip-gram

Word2Vec-mallin luomien word embedding -vektorien yhdistämiseen käytettiin keskiarvopoolausta (engl. *average pooling*). Keskiarvopoolaus aggregoi sanat yhdeksi dokumenttitason ominaisuusvektoriksi, säilyttäen kuitenkin sanojen merkitystä sen kontekstissa. Aggregoinnilla saadaan muutettua tekstidokumentit kiinteän pituisiksi vektoreiksi, jotka ovat yhteensopivia valittujen luokittelualgoritmien kanssa.

Keskiarvopoolauksessa syöte jaetaan sanajoukoittain ryhmiin tai ikkunoihin W, jonka jälkeen jokaisesta ikkunasta lasketaan keskiarvo. Viimeisessä vaiheessa vielä ikkunoiden keskiarvoista lasketaan yhteinen keskiarvo. Toimenpiteen voi esittää matemaattisesti yhtälöllä

$$u = \frac{1}{m}\sum_{j=1}^{m} \left(\frac{1}{|W_j|}\sum_{v \in W_j} v\right), \tag{11}$$

missä m on ikkunoiden lukumäärä, $|W_j|$ on ikkunassa olevien word embeddingien lukumäärä, ja v edustaa yksittäistä word embedding -vektoria. Ikkunoista on otettu keskiarvo, koska se vähentää ominaisuusavaruuden kompleksisuutta ja pienentää ylisovituksen riskiä.

4.3.3 Yhdistetyt ominaisuudet

Tutkielmassa kolmantena ominaisuuksien poimintamenetelmänä käytetään TF-IDF- ja word embedding -ominaisuuksien yhdistettyä muotoa. Jotta ominaisuudet voidaan yhdistää, täytyy TF-IDF matriisi ensin muuntaa tiheäksi matriisiksi, joka on laskennallisesti yhteensopiva keskiarvopoolattujen word embeddingien kanssa. Molemmat ominaisuudet tulee myös normalisoida, jotta niiden mittakaavat ovat keskenään yhtenäisiä. Normalisointi on totutettu tutkielmassa *StandardScaler*-menetelmällä, joka skaalaa kaikki muuttujat siten, että niiden keskiarvo on 0 ja keskihajonta 1. Normalisoinnin jälkeen ominaisuudet voidaan yhdistää toisiinsa, jolloin ominaisuusmatriisi sisältää tietoa molempien menetelmien ominaispiirteistä. Normalisoidut ominaisuudet on tutkielmassa muodostettu alla olevalla ohjelmakoodilla.

```
def combine_tfidf_embeddings(tfidf, embeddings):
    # Painota ominaisuudet
    tfidf_weight=0.7
    tfidf_weighted = tfidf * tfidf_weight
    embeddings_weighted = embeddings * (1 - tfidf_weight)

    # Yhdistä painotetut ominaisuudet
    combined_features      =      np.hstack((tfidf_weighted,
embeddings_weighted))

    return combined_features
```

Toteutuksessa on päädytty painottamaan TF-IDF-arvoja, sillä koulutusvaiheen testauksessa menetelmä suoriutui paremmin kuin word embedding. TF-IDF keskittyy yksittäisten sanojen korostamiseen ja on toimintakykyinen, vaikka koulutusdatasetti olisi pieni. Luokittelupäätös voi usein perustua vain tiettyjen sanojen tunnistamiseen, ilman tarvetta ymmärtää monimutkaisia kontekstuaalisia suhteita, joita word embedding -menetelmät tarjoavat. Yhdistämällä sanan esiintymistiheyden ja semanttisen merkityksen, malli voi hyödyntää sekä sanan tärkeyttä että sen kontekstuaalista merkitystä, mikä voi parantaa luokittelun tarkkuutta.

4.4 Luokittelijoiden toteutus

Molemmat teoreettisessa viitekehyksessä esitetyt luokittelualgoritmit on toteutettu käyttäen Scikit-learn-kirjaston valmiita luokittelijoita. Tukivektorikone käyttää kirjaston *SVC* (engl. *Support Vector Classification*) toteutusta epälineaarisella kernelillä, joka valitaan myöhemmin esitellyssä hyperparametrien optimoinnissa. Logistisen regression toteutukseen käytetään valmista *LogisticRegression*-menetelmää, ja iteraatioiden maksimimääräksi on asetettu 1000. Molempien luokittimien toteutuksessa parametri *random_state* on asetettu arvoksi 42, varmistaen että mallien toteutus on jokaisella kerralla toistettavissa. Siten voidaan varmistaa, että mahdolliset erot johtuvat muutetuista asetuksista tai itse datasta, eikä satunnaisuudesta. Tarkemmat mallikohtaiset hyperparametrikonfiguraatiot on esitetty kokeellisen osan koulutusvaiheosiossa. Valitsemalla sekä lineaarisen logistisen regression että epälineaarisen tukivektorikoneen voidaan arvioida, riittääkö yksinkertainen lineaarinen malli rikosilmoitusten luokitteluun vai parantaako monimutkaisempi, epälineaarinen malli luokittelun tarkkuutta.

4.5 FinBERT

Esikoulutetusta FinBERT-mallista käytettiin uusinta *cased-v1* versiota, ja luokkien määräksi alustettiin tutkielman rajausten mukaisesti kolme. Esikoulutetun mallin hienosäätöön käytettiin koulutusdatasettiä. FinBERT suoritti syötteen esikäsittelyn itse, joten erillistä esikäsittelyä ei tarvittu. Esikoulutetun mallin hienosäädössä malli mukautetaan tiettyyn luokittelutehtävään. FinBERT-mallin sanasto ei ole riippuvainen tutkielman koulutusdatasta, vaan se käyttää dataa hienosäätääkseen itseään käyttökohteeseen sopivaksi. Taulukossa 7 on esitetty FinBERT mallin hienosäätämiseen käytetyt parametrit. Hienosäädön parametreiksi valittiin ne arvot, joilla mallin validoinnin tarkkuus saatiin maksimoitua.

Taulukko 7. Käytetyt FinBERT hienosäätöparametrit.

Parametri	***Arvo***	***Kuvaus***
train_epochs	*5*	*Kuinka monta kertaa koko koulutusdata käydään läpi koulutuksen aikana.*
train_batch_size	*4*	*Näytteiden eräkoko parametreja päivitettäessä.*
eval_batch_size	*4*	*Näytteiden eräkoko arvioinnin aikana.*
warmup_steps	*200*	*Oppimisnopeuden vakauttamisen askelmäärä koulutuksen alussa.*
weight_decay	*0.01*	*Painojen säännöllistämisen skaalauskerroin.*

Viisi koulutussykliä valikoitui sopivaksi pienelle koulutusdatasetille, ja malli pystyi oppimaan hyvin ilman ylisovittumista. Syötteiden eräkoko valittiin sopivaksi suhteutettuna datasetin kokoon. Malli päivittää painoarvonsa jokaisen käsitellyn erän jälkeen, jolloin liian suuri eräkoko pienellä datasetillä johtaisi alisovittamiseen. Parametri *warmup_steps* estää mallia tekemästä liian suuria päivityksiä koulutuksen alussa. Arvo 200 valikoitui sopivaksi, koska se on käytettävällä setillä riittävän matala koulutusvaiheen vakauttamiseksi ilman hienosäädön liiallista rajoittamista. Käytetty skaalauskerroin osoittautui testatuista arvoista tasapainoisimmaksi ja tuotti parhaat validointitulokset. Muita testattuja kertoimia olivat muun muassa 0.0, 0.001, 0.1 ja 0.0001. Käytetyllä konfiguraatiolla saavutettiin 100 % validointitarkkuus.

4.6 Koulutusvaihe ja mallien tarkkuus

Kokeellisen osuuden viimeisessä alaluvussa on esitetty käytettyjen luokittelualgoritmien parametrioptimoinnin toteutus sekä validointivaiheessa mallien tarkkuuden mittaamiseen käytetty menetelmä. Viimeisenä esitetään koulutettujen mallien luokittelutarkkuudet ennen lopulliseen, yleistyskykyä mittaavaan vertailudatasetin luokitteluun siirtymistä.

4.6.1 Luokittimien hyperparametrien optimointi

Luokittimien hyperparametrien optimoinnilla voidaan parantaa mallin suorityskykyä etsimällä asetukset, jotka maksimoivat luokittelualgoritmin suorituskyvyn. Tutkielmassa optimaaliparametrien löytämiseen on käytetty *GridSearchCV*-toteutusta, jossa käydään läpi kaikki mahdolliset parametrien yhdistelmät ja valitaan se, joka tuottaa parhaan suorituskyvyn (Scikit-learn, 2024, Luku 3.2). Suorituskyvyn mittaamiseen on käytetty ristivalidointia, jonka toiminta on esitetty tarkemmin seuraavassa alaluvussa. Kaikkien mahdollisten parametrien läpikäynti on laskennallisesti raskasta, mutta varmistaa optimaaliparametrien löytämisen. Tutkielman pienen datasetin takia menetelmä on sopiva, ja eniten aikaa vienyt parametrioptimointi kesti tukivektorikoneella 7.46 sekuntia ja logistisella regressiolla 22.08 sekuntia. *GridSearchCV*-toteutuksella valitut, oletusasetuksista eroavat logistisen regression sekä tukivektorikoneen hyperparametrit on esitetty taulukoissa 8 ja 9.

Taulukko 8. LogReg: optimaaliset hyperparametrit.

Luokitin	***Esikäsittely***	***Ominaisuus***	***Parametrit***

LogReg	*Stem*	*TF-IDF*	`C: 10`
			`penalty: 'l2'`
			`solver: 'saga'`

LogReg	*Stem*	*Embedding*	`C: 0.1`
			`penalty: 'l2'`
			`solver: 'saga'`

LogReg	*Stem*	*Yhdistetty*	`C: 100`
			`penalty: 'l1'`
			`solver: 'saga'`

LogReg	*Lemma*	*TF-IDF*	`C: 0.1`
			`penalty: 'l2'`
			`solver: 'saga'`

LogReg	*Lemma*	*Embedding*	`C: 0.1`
			`penalty: 'l2'`
			`solver: 'saga'`

LogReg	*Lemma*	*Yhdistetty*	`C: 100`
			`penalty: 'l1'`
			`solver: 'saga'`

Taulukko 9. SVM: optimaaliset hyperparametrit.

Luokitin	*Esikäsittely*	*Ominaisuus*	*Parametrit*

SVM	*Stem*	*TF-IDF*	`C: 0.1`
			`coef0: 1.0`
			`kernel: 'poly'`
			`degree: 5`
			`gamma: 'scale'`

SVM	*Stem*	*Embedding*	`C: 1`
			`coef0: 0.5`
			`kernel: 'poly'`
			`degree: 2`
			`gamma: 'scale'`

SVM	*Stem*	*Yhdistetty*	`C: 0.01`
			`coef0: 0.0`
			`kernel: 'sigmoid'`
			`gamma: 'scale'`

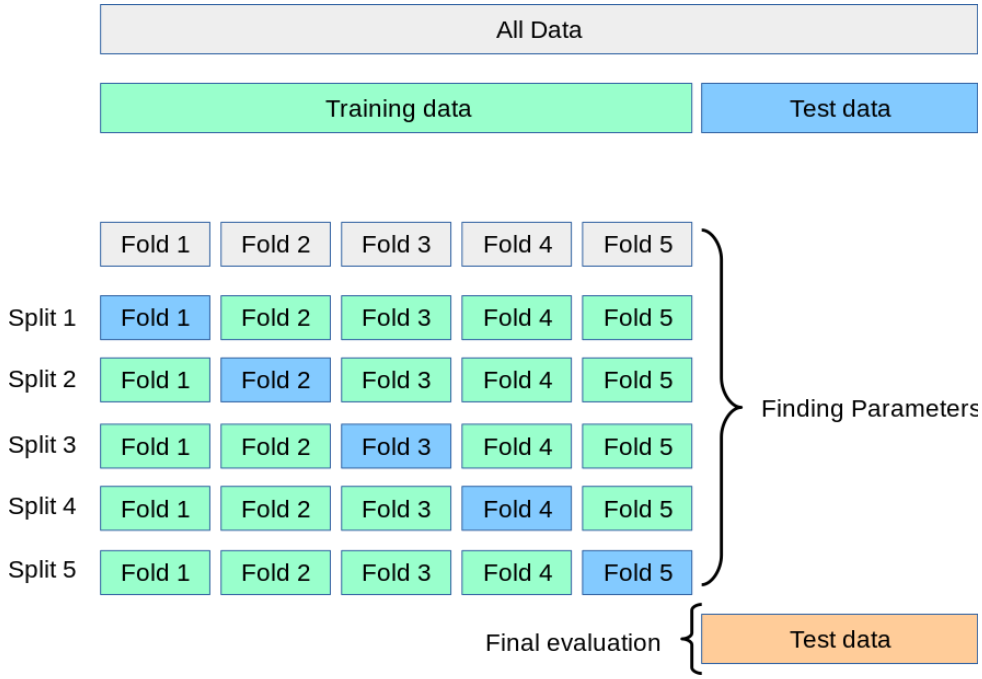
SVM	*Lemma*	*TF-IDF*	`C: 10`
			`coef0: 0.0`
			`kernel: 'sigmoid'`
			`gamma: 'scale'`

SVM	*Lemma*	*Embedding*	`C: 10`
			`coef0: 0.5`
			`kernel: 'poly'`
			`degree: 2`
			`gamma: 'scale'`

SVM	*Lemma*	*Yhdistetty*	`C: 1`
			`coef0: 0.0`
			`kernel: 'sigmoid'`
			`gamma: 0.001`

4.6.2 Ristivalidointi (CV)

Luokittelijoiden parametrioptimointi ja koulutuksen jälkeinen, tarkkuutta mittaava validointi on toteutettu hyödyntäen ristivalidointia. Ristivalidoinnissa data jaetaan useisiin osajoukkoihin (engl. *fold*), joista jokainen osajoukko toimii vuorollaan validaatiojoukkona ja loput koulutusjoukkona (Scikit-learn, 2024, Luku 3.1).



Kuvio 4. Ristivalidointi parametrioptimoinnissa (Scikit-learn, 2024, Luku 3.1).

Kuviossa 4 on hahmotettu ristivalidoinnin toiminta. Tutkielmassa kuvion 4 vihreät osajoukot kuvaavat koulutusdatasettiä ja siniset osajoukot puolestaan validointidataa. Erillinen kuviossa 4 esitetty beige syötejoukko kuvaa tutkielman vertailudataa. Tutkielmassa jokainen ristivalidointi suoritetaan viidellä osajoukolla käyttäen *StratifiedShuffleSplit*-menetelmää. Menetelmä muodostaa osajoukot satunnaisesti varmistaen, että kunkin luokan näyteosuudet säilyvät osajoukoissa samoina (Scikit-learn, 2024, Luku 3.1.2.2.2). Menetelmä on valittu tutkielmaan, koska se varmistaa osajoukkojen tasapainon pienikokoisellakin datasetillä.

4.6.3 Tarkkuuden mittaus

Mallien suorituskyvyn mittarina tutkielmassa käytettiin tarkkuustulosta (engl. *accuracy score*). Moniluokkaisessa luokittelussa tarkkuustulos saadaan selvittämällä kunkin osajoukon oikeiden ennusteiden y osuus yhtälöllä

$$tarkkuus(y, \hat{y}) = \frac{1}{n_s}\sum_{i=0}^{n_s-1} 1(\hat{y}_i = y_i), \quad (12)$$

missä $\hat{y}$ on ennustettu luokka ja termi n_s ennustettavien syötteiden määrä (Scikit-learn, 2024, Luku 3.4.2.2.2). Jos ennustettu syöte vastaa todellista luokkaa, on tarkkuustulos 1; muussa tapauksessa se on 0. Tutkielmassa kullekin viidestä osajoukosta lasketaan tarkkuustulos. Mallin lopullinen tarkkuustulos saadaan laskemalla viiden osajoukon tarkkuustuloksen keskiarvo. Alla olevassa taulukossa 10 on esitetty tutkielmassa koulutettujen mallien validointivaiheen tarkkuudet. Malli muodostuu valitusta luokittelualgoritmista, esikäsittely-, ja ominaisuudenpoimintamenetelmästä.

Taulukko 10. Luokittimien validoinnin tarkkuustulokset.

Luokitin	***Esikäsittely***	***Ominaisuus***	***Tarkkuus %***
LogReg	*Stem*	*TF-IDF*	98.67
LogReg	*Stem*	*Embedding*	83.55
LogReg	*Stem*	*Yhdistetty*	98.22
LogReg	*Lemma*	*TF-IDF*	95.56
LogReg	*Lemma*	*Embedding*	87.56
LogReg	*Lemma*	*Yhdistetty*	97.33
SVM	*Stem*	*TF-IDF*	97.33
SVM	*Stem*	*Embedding*	81.78
SVM	*Stem*	*Yhdistetty*	95.56
SVM	*Lemma*	*TF-IDF*	97.33
SVM	*Lemma*	*Embedding*	88.89
SVM	*Lemma*	*Yhdistetty*	96.00

Taulukosta 10 nähdään, että validoinnissa paras suorituskyky on TF-IDF-menetelmällä riippumatta käytetystä luokittelualgoritmista tai esikäsittelymenetelmästä. Viitaten että yksinkertaisempi, yksittäisten sanojen tärkeyteen keskittyvä menetelmä olisi sopivin valittuun luokittelutehtävään. Word embeddingien suorituskyky on jokaisessa toteutuksessa merkittävästi heikompi. On mahdollista, että pienestä datasetistä muodostettu sanasto ei tuota riittävästi luokittelupäätökselle merkittävää tietoa, tai valitut menetelmät

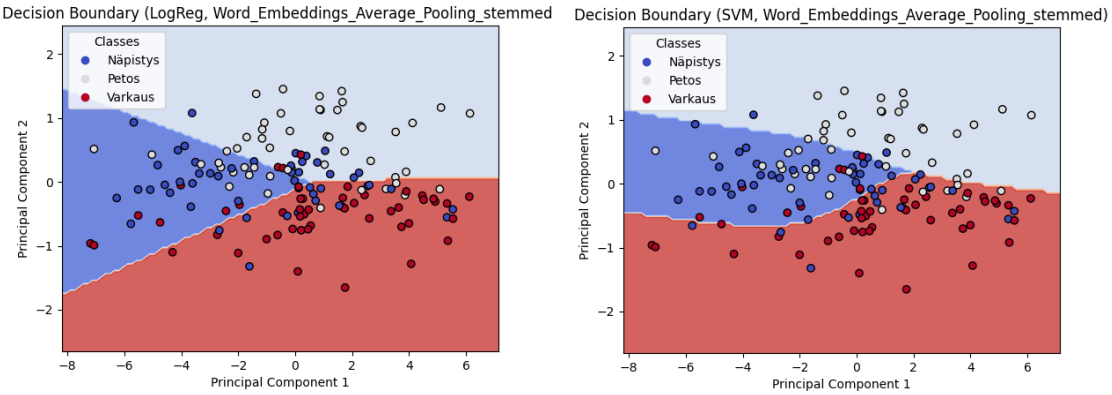
kohinan poistamiseen eivät ole riittäviä. On myös mahdollista, että word embedding tuo yhdistettyyn ominaisuuteen lisäarvon sijaan kohinaa, joka vaikuttaa alentavasti tarkkuustulokseen. Valittujen luokittelualgoritmien tai esikäsittelymenetelmien välillä tarkkuustuloksissa ei ole suuria eroavaisuuksia. Pienistä eroista on huomattavissa, että logistinen regressio on tukivektorikonetta tarkempi, mikäli syöte on stemmattu (Stem). Lemmatisoinnissa (Lemma) tukivektorikone on suorituskyvyltään hieman tarkempi TF-IDF ja word embedding ominaisuuksilla.

Vaikka validointidatasetillä toteutetun ristivalidoinnin perusteella voidaan tehdä johtopäätöksiä mallien suorituskyvystä, eivät tulokset ole riittäviä arvioimaan mallien yleistyskykyä eli kykyä luokitella täysin uusia selostuksia. Tämä johtuu siitä, että datasetin ollessa pienikokoinen myös validointidatasetti jää rajalliseksi. Rajallisella datalla validointi ja koulutussetin välille saattaa jäädä merkittäviä yhtäläisyyksiä, jotka voivat johtaa suorituskyvyn ylipositiiviseen arviointiin. Ristivalidoinnilla saatuja tarkkuuksia voidaan käyttää varmistaessa, ettei malli ylisovitu. Ylisovittumisen seurauksena lopulliset tarkkuudet olisivat matalia, sillä joidenkin ristivalidoinnin osajoukkojen tarkkuustulokset olisivat heikkoja, mikä laskisi keskiarvoa.

Mallien toiminnan ymmärtämiseksi, voidaan niiden koulutusdatan perusteella muodostamia päätösrajapintoja visualisoida kaksiulotteisella kuvaajalla. Kuvion 5 vasemmassa kuvaajassa on esitetty logistisen regression ja oikealla tukivektorikoneen muodostamat päätösrajat heikoiten suoriutuneista word embedding -ominaisuuksista. Visualisointia varten myös ominaisuudet täytyi muuntaa kaksiulotteiseksi. On tärkeä ymmärtää johtopäätöksiä tehdessä, ettei visualisointi kykene esittämään todellista mallin moniulotteista päätöspintaa. On siis mahdollista, että syöte on luokiteltu oikein, vaikka kaksiulotteisessa visualisoinnissa vastaava syöte asettuisi toisen luokan päätöspinnalle.

Kuvion 5 kuvaajissa on havaittavissa, että luokkakohtaiset päätösrajat sisältävät paljon toiseen luokkaan kuuluvia syötteitä. Tämä voi tarkoittaa, että mallilla on vaikeuksia löytää syötteistä luokkakohtaisia eroavaisuuksia, johtaen huonoon tarkkuuteen. Huono

ominaisuuksien erotettavuus voi johtua pienestä koulutusdatasetin koosta tai liiallisesta kohinasta. Vaikutus ei kuitenkaan ole yhtä merkittävä kuin kuvion 5 kaksiulotteiset kuvaajat indikoivat, sillä suorituskyvyltään heikoimmassa mallissa tarkkuus oli melko korkea: 83.55 %.



Kuvio 5. Päätösrajapintojen kaksiulotteinen visualisointi.

Huomioiden esitetyt perustelut sekä validointivaiheessa saavutetut korkeat tarkkuustulokset, voidaan toteutettujen mallien olevan toimintakuntoisia, ja siirtyä arvioimaan niiden yleistyskykyä vertailudatan avulla.

5 Tulokset

Tässä luvussa on esitetty tutkielman kokeellisen osan lopullisen, vertailudatan luokittelun tarkkuustulokset. Vertailudatasetti koostuu 44 malleille täysin uusista selostuksista, jonka kokoonpano on tarkemmin esitetty yhdessä vertailudatan alaluvussa ja liitteessä 2. Mallin todellista suorituskykyä on asianmukaisempaa arvioida erillisellä vertailudatasetillä, sillä hyvä yleistyskyky on tutkittavan käyttökohteen eli rikosilmoitusten luokitteluun kriittistä. Lisäksi kuratoidulla vertailudatalla voidaan tehdä tarkempia havaintoja mallin suorituskyvystä syötetarkkuudella.

Luvussa esitetään mallien suorituskykyä, ja verrataan vertailudatan tarkkuuslukemia validointivaiheessa saatuihin arvoihin. Alaluvussa esitetään johtopäätöksiä tukeutuen luokittelua kuvaaviin konfuusiomatriiseihin. Matriisien perusteella tehtyjä havaintoja konkretisoidaan tutkimalla tarkemmin virheellisesti luokiteltuja syötteitä.

Mallien vertailudatasetin luokittelutarkkuudet on esitetty alla olevassa taulukossa 11. Taulukossa on esitetty jokaisessa mallissa käytetty luokittelualgoritmi, esikäsittely- ja ominaisuuksienpoimintamenetelmä. Lisäksi taulukossa 11 on esitetty oikeiden ennusteiden lukumäärä 44 syötteestä. Tarkkuuden mittaamiseen on käytetty samaa tarkkuustulosta, eli oikeiden ennusteiden osuutta suhteessa kaikkiin syötteisiin. Myös vertailu- ja validointitarkkuuden välinen muutos on laskettu taulukkoon analyysin tukemiseksi. Jälkimmäisenä taulukkoon on sisällytetty hienosäädetyn FinBERT-mallin saavuttama tarkkuus vertailudatan luokittelussa.

Taulukko 11. Vertailusetin luokittelun tarkkuusarvot.

Luokitin	*Esikäsittely*	*Ominaisuus*	*Oikein #*	*Tarkkuus %*	*Muutos %*
LogReg	*Stem*	*TF-IDF*	39	88.64	*-10.03*
LogReg	*Stem*	*Embedding*	34	77.27	*-6.28*
LogReg	*Stem*	*Yhdistetty*	39	88.64	*-9.58*
LogReg	*Lemma*	*TF-IDF*	39	88.64	*-6.92*
LogReg	*Lemma*	*Embedding*	31	70.45	*-17.11*
LogReg	*Lemma*	*Yhdistetty*	38	86.36	*-10.97*
SVM	*Stem*	*TF-IDF*	39	88.64	*-8.69*
SVM	*Stem*	*Embedding*	34	77.27	*-4.51*
SVM	*Stem*	*Yhdistetty*	39	88.64	*-6.92*
SVM	*Lemma*	*TF-IDF*	38	86.36	*-10.97*
SVM	*Lemma*	*Embedding*	29	65.91	*-22.98*
SVM	*Lemma*	*Yhdistetty*	36	81.82	*-14.18*
FinBERT			41	93.18	*-6.82*

Vertailusetin tuloksista on havaittavissa, että tarkkuus on toteutuksesta riippumatta validointitarkkuutta heikompi. Tämä on oletettavaa, sillä validointidata on alkuperältään eritelty otos koulutusdatasta, ja siten myös epäsuorasti heijastuva koulutukseen käytetystä datasta. Vertailudatasetti on täysin muista dataseteistä itsenäinen kokonaisuus, ja on malleille täysin uutta. Siten on mahdollista, että vertailudata on koulutus- ja validointidataan verrattuna mallille vaikeampaa.

Heikommasta tarkkuustuloksesta huolimatta taulukossa 11 esitetyt tulokset ovat yhtenäisiä validointivaiheen taulukosta 10 tehtyjen havaintojen kanssa: Toteutetuista malleista tarkkuus on paras TF-IDF-luokittimilla ja heikoin word embeddingiä käyttävillä malleilla. Aiemmin esitetty perustelu siitä, että yhdistetyssä ominaisuudessa word embedding tuo lisäarvon sijaan dataan kohinaa, ei ole yhtä selkeästi näkyvä taulukon 11 tuloksissa. Tulosten mukaan yhdistetyn ominaisuuden tarkkuus on lähes identtinen TF-IDF:n

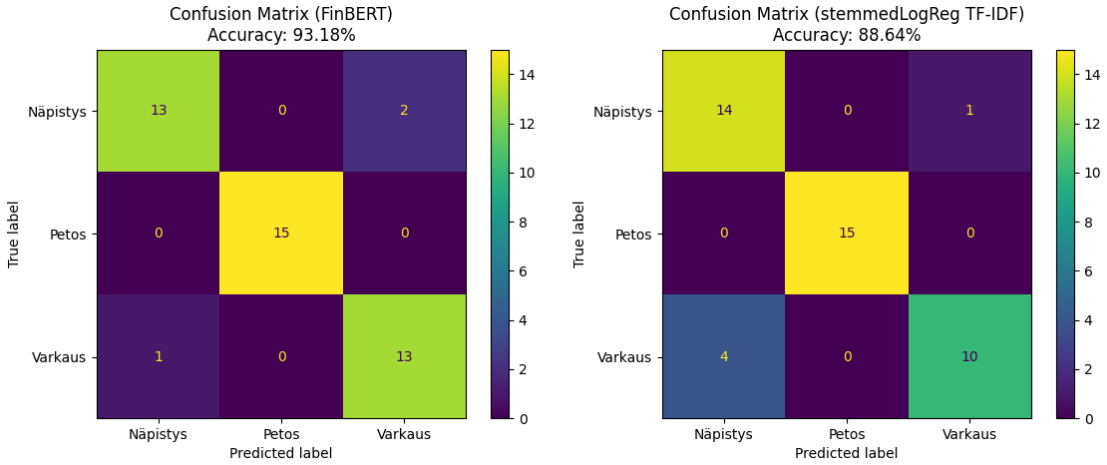
kanssa, vaikkakin hieman matalampi syötteen ollessa lemmatisoitu. On todennäköistä, että lemmatisointi yksinkertaistaa syötteiden sanoja liikaa, jolloin mallien on vaikea muodostaa tarkkaa päätöspintaa syötteiden luokittelemiseksi. Perustelua tukee taulukon 11 tulokset, jossa lemmatisointia käyttävät mallit suoriutuvat huonommin verrattuna stemmausmalleihin, pois lukien logistisen regression TF-IDF mallia. Ero stemmauksen ja lemmatisoinnin välillä on näkyvä etenkin word embedding -malleissa. Stemmaus poistaa sanoista vain loppuosan taivutusmuodon, jonka seurauksena mallin sanastoon voi muodostua erilaisia variaatioita samasta sanasta. Näiden sanojen välillä voi olla luokittelupäätöksen kannalta merkittävää tietoa, joka edesauttaa stemmausmallien luokittelutarkkuutta.

Valitun luokittelualgoritmin vaikutus luokittelutarkkuuteen on minimaalinen validointi- ja vertailudatan tarkkuustuloksien perusteella. Taulukossa 11 on kuitenkin havaittavissa, että logistinen regressio suoriutuu hieman tukivektorikonetta tarkemmin, kun ominaisuutena ei ole pelkkä TF-IDF. Logistisen regression hieman parempi suorituskyky vihjaa siihen, että yksinkertaisempi lineaarinen malli on valittuun luokittelutehtävään sopivampi. On mahdollista, että epälineaarinen tukivektorikone muodostaa luokittelutehtävään liian monimutkaisen päätösrajapinnan, jolloin se voi ylisovittua pienempien tai kohinaa sisältävien ominaisuusjoukkojen kohdalla.

Luokittelutehtävään hienosäädetyllä FinBERT-mallilla oli vertailudatasetin luokittelussa paras suorituskyky, ja virheellisiä ennusteita malli teki vain kolme. Mallin erittäin hyvä suorituskyky johtuu todennäköisesti sen kyvystä ymmärtää lauseiden kontekstuaalista merkitystä. Lisäksi mallin esikoulutuksen takia se ei ole sanaston muodostamisessa riippuvainen koulutusdatasta, kuten muut tutkielman toteutukset. Malli kykenee ymmärtämään koulutusdatassa esiintymättömiä sanoja, jolloin sen yleistyskyky on myös parempi.

5.1 Konfuusiomatsiisit

Tässä alaluvussa esitellään mallien luokitteluprosessia kuvaavia konfuusiomatriiseja, jotka toimivat tukena luokittelusta tehtäviin johtopäätöksiin. Konfuusiomatriisin avulla voidaan hahmottaa, miten luokitin ennustaa sille annettujen syötteiden luokat. Konfuusiomatriisin y-akseli kuvaa syötteen oikeaa luokkaa, ja x-akseli mallin tekemää ennustetta. Oikein luokitellut syötteet asettuvat matriisiin siten, että ne ovat molempien akseleiden nähden saman luokan kanssa linjassa.



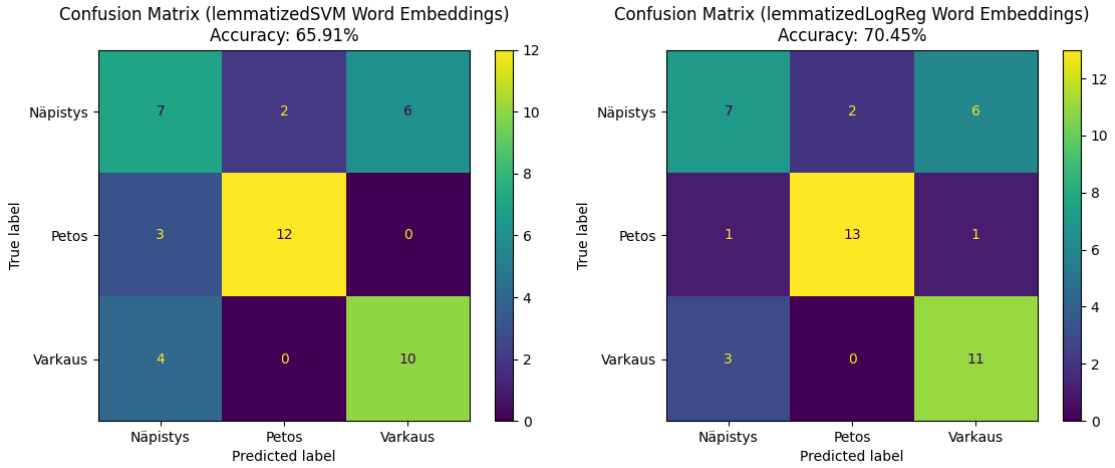
Kuvio 6. Suorituskyvyltään tarkimpien mallien konfuusiomatriisit

Yllä olevassa kuviossa 6 on esitetty suorituskyvyltään tarkimpien mallien vertailudatan luokittelun konfuusiomatriisit. Vasemmassa matriisissa on esitetty FinBERT-malli, ja oikeanpuoleisessa matriisissa stemmausta ja TF-IDF-ominaisuuksia käyttävä logistisen regression malli. Muiden 88.64 % tarkkuusarvon saaneiden mallien konfuusiomatriisit ovat identtisiä kuvion 6 oikeanpuoleisen matriisiin kanssa. Matriiseista on havaittavissa, että kaikki virheelliset luokitukset tapahtuvat luokkien näpistys ja varkaus kesken. Mallit kykenevät erottelemaan petosnimikkeeseen kuuluvat selostukset tarkasti. Nimikkeinä varkaus ja näpistys ovat tekotavaltaan hyvin samanlaisia, ja niiden tyypillinen erottava tekijä on anastetun omaisuuden arvo. Esikäsittelyssä selostuksista kuitenkin poistetaan numerot, jolloin menetetään mahdollinen tarkka tieto anastetun omaisuuden arvosta. Pelkkä numeroiden säilyttäminen todennäköisesti heikentäisi suorituskykyä, koska selostuksiin

tulisi samalla lisää luokittelulle merkityksetöntä kohinaa, kuten kellonaikoja. Kahden samankaltaisen luokan välillä mallin on vaikeampi löytää oikean luokituspäätöksen kannalta merkittäviä ominaisuuksia, jolloin syötteet asettuvat erittäin lähelle luokkien välistä päätöspintaa, tai väärän luokan puolelle.

Täysin identtiset tarkkuustulokset ja konfuusiomatriisit indikoivat, että malleilla on toteutuksesta riippumatta vaikeuksia erotella samankaltaisia syötteitä oikeaan luokkaan. Tarkastellessa luokittelua syötetasolla huomataan, että virheellisiä ennustuksia esiintyy kaikissa liitteessä 2 esitellyissä laatumääritelmissä. Tätä voidaan perustella sillä, että tekotavan luokittelun kannalta merkittävät sanat esiintyvät jo suppeassakin selostuksessa. Pidemmät selostukset sisältävät usein kertomusta tukevia lisätietoja tai yksityiskohtia, jotka eivät ole rikosnimikkeen tunnistamiseen yhtä oleellisia.

Virheellisesti luokitelluista selostuksista jokainen kuului liitteessä 2 esitettyihin tunnisteisiin *eriävä* tai *sama*. Esimerkiksi tarkimmissa, 86.64 % tarkkuustuloksen malleissa kolme virheellistä ennustusta kuuluivat tunnisteeseen *sama*, ja kaksi tunnisteeseen *eriävä*. Tunnisteeseen *sama* kuuluvat selostukset ovat ilmoituksia, joissa on käytetty mahdollisimman paljon samoja sanoja riippumatta siitä, mihin nimikkeeseen ilmoitus kuuluu. Havainto tukee aiempaa arviota mallien vaikeuksista erotella samankaltaisia syötteitä. Tunnisteeseen *eriävä* kuuluvat syötteet ovat täysin uniikkeja selostuksia. On todennäköistä, että selostuksissa on paljon malleille tuntemattomia sanoja tai että selostuksesta karsiutuu oleellinen tieto omaisuuden arvosta esikäsittelyn aikana, johtaen virheellisiin luokittelupäätöksiin. Tarkkuustuloksen parantamiseksi tulisi kehittää etenkin syötteiden esikäsittelyä ja ominaisuudenpoimintaa.



Kuvio 7. Suorituskyvyltään heikoimpien mallien konfuusiomatriisit

Yllä olevassa kuviossa 7 on esitetty konfuusiomatriisit malleille, jotka saivat heikoimmat tarkkuustulokset. Molemmat matriisit kuvaavat lemmatisointia käyttäviä malleja, ja tukee aiempaa perustelua stemmauksen hyödystä. Myös kuvion 7 matriiseissa on havaittavissa mallien vaikeudet erotella etenkin varkaus- ja näpistysselostuksia. Word embedding -malleilla on myös vaikeuksia petosrikosten luokittelussa. Näiden mallien suorituskyvyn parantaminen vaatisi esikäsittelyn kehittämisen lisäksi huomattavasti laajempaa koulutusdataa, jolloin mallin sanastosta saadaan kattavampi. On myös mahdollista, että word embedding on toimintaperiaatteensa vuoksi huonompi valinta kyseiseen luokittelutehtävään. Luokittelutehtävän kannalta yksinkertaisempi, yksittäisiin sanoihin keskittyvä ratkaisu voi olla tehokkaampi.

6 Johtopäätökset

Työssä tutkittiin luonnollisen kielen käsittelyä rikostutkinnan kontekstissa. Tutkielman tavoitteena oli tutkia teknologian soveltuvuutta rikosilmoitusten luokitteluun. Tavoitteen johdonmukaistamiseksi tutkielmalle asetettiin kolme tutkimuskysymystä:

1. Soveltuuko avainsanapohjainen luonnollisen kielen käsittely rikosilmoitusten luokitteluun?
2. Miten eri menetelmät ja luokittelijat vaikuttavat luokittelun tarkkuuteen?
3. Suoriutuuko esikoulutettu syväoppimismalli paremmin kuin käyttökohteeseen rajallisella datalla koulutettu luokittelija?

Kysymykset 2 ja 3 toimivat kysymyksen 1 alikysymyksinä. Johtopäätöksien perustana toimivat kokeellisen osan toteutuksen aikana tehdyt havainnot ja esitetyt tulokset. Lisäksi tutkielman alaluvussa on esitetty jatkotutkimuksen kannalta mahdollisia kehityssuuntia.

Käsiteltäessä ensimmäistä tutkimuskysymystä, voidaan kokeellisen osan yleistyskykyä mittaavien tulosten perusteella todeta, että NLP soveltuu käytettäväksi rikosilmoitusten luokitteluun. Vaikka toteutettujen mallien suorituskykyjen välillä oli suuriakin eroja, on jokainen malli kykenevä luokittelemaan tarkasti toisistaan eroavia rikoksia. NLP-mallit kykenevät löytämään rikosselostuksista kullekin luokalle merkittäviä ominaisuuksia, joiden perusteella voidaan tehdä luokittelupäätös. Rikosilmoitukset ovat siten luokittelukelpoisia syötteitä luonnollisen kielen käsittelyssä. Tutkielman tavoitteena ei ollut suoraan luokittelutarkkuuden maksimointi, mutta kokeellisessa osassa tunnistettiin konkreettisia esikäsittelyn ja ominaisuudenpoiminnan kehityskohteita, joilla parantaa suorituskykyä. Nämä johtopäätökset viittaavat siihen, että teknologia on itsessään riittävän kypsä käyttötarkoitukseen.

Verrattaessa tuloksia käytettyjen menetelmien välillä voidaan todeta, että valittuun luokittelutehtävään parhaaksi ominaisuudenpoimintamenetelmäksi osoittautui yksittäisten sanojen merkitystä korostava TF-IDF-menetelmä. Myös korkeasti TF-IDF-arvolla painotettu word embeddingien kanssa yhdistetty ominaisuus tuotti varsin korkeita

tarkkuustuloksia. Word embedding osoittautui heikoimmaksi menetelmäksi, jota käytettäessä malleilla oli eniten vaikeuksia oikean ennustuksen tuottamisessa. Valituista luokittelualgoritmeista lineaarinen regressio oli tulosten perusteella hieman tarkempi verrattuna epälineaariseen tukivektorikoneeseen, vaikkakin valitun luokittelijan vaikutus suorituskykyyn oli minimaalinen. Stemmaus osoittautui lemmatisointiin verrattuna tarkkuuden kannalta paremmaksi esikäsittelyn menetelmäksi. Viitaten siihen, että lemmatisointi yhdenmukaistaa syötteitä liikaa ja menettää siten luokittelupäätökselle merkittävää tietoa.

Transformer-arkkitehtuuriin perustuva ja suomen kielellä esikoulutettu FinBERT-malli suoriutui luokittelusta tarkemmin muut toteutetut mallit. Mallin esikoulutettu sanasto ja kyky ymmärtää käsiteltävän sekvenssin eli lauseen kontekstuaalista merkitystä mahdollisti erittäin hyvän yleistämiskyvyn saavuttamisen pienelläkin koulutusdatalla.

6.1 Kehityssuunnat ja suositukset

Päällimmäisenä parannuskohteena on mallien esikäsittelyn ja ominaisuuksienpoiminnan kehittäminen. On oleellista tunnistaa rikosnimikekohtaiset merkittävät ominaispiirteet, jotka tulisi saada poimittua syötteistä ominaisuuksiksi. Nämä ominaisuudet pitää tunnistaa, poimia ja muuntaa muotoon, joka ei lisää ominaisavaruuteen kohinaa ja heikennä luokittelutarkkuutta. Lisäämällä syötteen käsittelyyn nimettyjen entiteettien erottelun (engl. *Named entity recognition*, NER), voisi malli tunnistaa ja yhdistää harvinaisia mutta luokittelun kannalta merkityksettömiä tietueita yhtenäiseksi ominaisuudeksi. Esimerkiksi selostuksissa esiintyvät luvut välillä [0, 500] voitaisiin käsitellä yhtenä tunnisteena, ja vastaavasti luvut välillä [501, 5000] toisena tunnisteena. Näin malli saisi käyttöön varkauden ja näpistyksen erottavan ominaisuuden ilman jokaisen numeron erikseen käsittelystä aiheutuvaa kohinaa.

Teknologian käytännön soveltamisen edistämiseksi tulisi tutkia, miten mallit suoriutuvat suuremmalla luokkamäärällä. Tutkielmassa tarkasteltiin petosta, varkautta ja näpistystä,

mutta Suomen rikoslaissa on huomattavasti enemmän nimikkeitä. Suurin haaste luokkien lisäämisessä on datan määrän kasvaminen, sillä koulutusdatan tulisi olla tasapainoinen kaikkien luokkien välillä. Jotta luokkien lisääminen pysyisi työmäärältään realistisena ja säilyttäisi datan korkean laadun, tunnistetaan tarve saada tutkimuskäyttöön oikeita rikosilmoituksia.

Viimeisimpänä parannuskohteena esitetään luokittelulogiikan kehittäminen. Tutkielmassa toteutetut mallit olettavat, että syöte voi kuulua vain yhteen luokkaan. Todellisuudessa rikosilmoituksesta voidaan tunnistaa usea rikosnimike, mikäli selostus sisältää usean rikosnimikkeen täyttävät tunnusmerkistöt. Tutkielmassa käytetty logistisen regression luokittelualgoritmi olettaa syötteen kuuluvan vain yhteen luokkaan, jonka takia se ei sovellu vastaavaan käyttökohteeseen. FinBERT-mallin hyvän suorituskyvyn perusteella myös monikielisten transformer-mallien, kuten XML-RoBERTan suorituskykyä luokittelutehtävään voisi tutkia laajemmin.

Lähteet

Beuker, A. (2023). Rikollisuustilanne 2022: rikollisuuskehitys tilastojen ja tutkimusten valossa. Helsingin yliopisto, kriminologian ja oikeuspolitiikan instituutti. Noudettu 18.10.2024 osoitteesta http://hdl.handle.net/10138/568863

Bonisoli, G., Rollo, F., & Po, L. (2021). Using Word Embeddings for Italian Crime News Categorization. 2021 16th Conference on Computer Science and Intelligence Systems (FedCSIS), 461–470. Noudettu 22.10.2024 osoitteesta https://doi.org/10.15439/2021F118

Cortes, C., Vapnik, V. (1995). Support-vector networks. Machine Learning 20, 273–297 (1995). Noudettu 28.10.2024 osoitteesta https://doi.org/10.1007/BF00994018

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv. Noudettu 26.10.2024 osoitteesta https://doi.org/10.48550/arXiv.1810.04805

Eisenstein, J. (2018). Natural Language Processing. Noudettu 16.10.2024 osoitteesta https://cseweb.ucsd.edu/~nnakashole/teaching/eisenstein-nov18.pdf

Esitutkintalaki 22.7.2011/805. Finlex. Noudettu 17.10.2024 osoitteesta https://finlex.fi/fi/laki/ajantasa/2011/20110805

Finanssiala. (2024). Huijareilla oli aktiivinen vuosi 2023 – Pankit saivat estettyä digihuijauksia lähes 33 miljoonan euron edestä. Noudettu 19.10.2024 osoitteesta https://www.finanssiala.fi/uutiset/huijareilla-oli-aktiivinen-vuosi-2023-pankit-saivat-estettya-digihuijauksia-lahes-33-miljoonan-euron-edesta/

Finanssiala. (2024). Huijaukset kovassa kasvussa – pankit onnistuivat pysäyttämään yli 18 miljoonaa euroa huijattua rahaa. Noudettu 19.10.2024 osoitteesta https://www.finanssiala.fi/uutiset/huijaukset-kovassa-kasvussa-pankit-onnistuivat-pysayttamaan-yli-18-miljoonaa-euroa-huijattua-rahaa/

Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. Teoksessa C. Nédellec & C. Rouveirol (Toim.), Machine Learning: ECML-98 (Vsk. 1398, s. 137–142). Springer Berlin Heidelberg. Noudettu 25.10.2024 osoitteesta https://doi.org/10.1007/BFb0026683

Jokela, J. (2022). Classification of Finnish injury descriptions using BERT. Perustieteiden korkeakoulu, Aalto-yliopisto. Noudettu 3.10.2024 osoitteesta https://aalto-doc.aalto.fi/handle/123456789/115208

Jurafsky, D., Martin, J. Speech and Language Processing. (2024). Noudettu 20.10.2024 osoitteesta https://web.stanford.edu/~jurafsky/slp3/

Kielipankki. (n.d). Kielipankki. Noudettu 2.10.2024 osoitteesta https://www.kielipankki.fi/

Mansikka, M. (2020). Uutisdatan luokittelu: Suomenkieliset uutisotsikot. Tampereen yliopisto. Noudettu 3.10.2024 osoitteesta https://trepo.tuni.fi/handle/10024/123571

Marcus, M. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. Defense Technical Information Center. Noudettu 1.11.2024 osoitteesta https://doi.org/10.21236/ADA273556

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space (No. arXiv:1301.3781). arXiv. Noudettu 22.10.2024 osoitteesta http://arxiv.org/abs/1301.3781

Mishra, S., & Misra, A. (2017). Structured and Unstructured Big Data Analytics. 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), 740–746. Noudettu 23.10.2024 osoitteesta https://doi.org/10.1109/CTCEEC.2017.8454999

Mähönen, M. (2013). Tekstin luokittelu. Noudettu 3.10.2024 osoitteesta https://trepo.tuni.fi/handle/123456789/21584

NLTK. (2024). Natural Language Toolkit. Noudettu 3.11.2024 osoitteesta https://www.nltk.org/

Poliisi. (n.d). Rikoksen tutkinta—Tutustu esitutkintaan. Noudettu 28.9.2024, osoitteesta https://poliisi.fi/rikoksen-tutkinta

Rikoslaki 19.12.1889/39. Finlex. Noudettu 17.10.2024 osoitteesta https://finlex.fi/fi/laki/ajantasa/1889/18890039001

Robertson, S. (2004). Understanding inverse document frequency: On theoretical arguments for IDF. Journal of Documentation, 60(5), 503–520. Noudettu 20.10.2024 osoitteesta https://doi.org/10.1108/00220410410560582

Savolainen, A. (2024). Varkaus vai näpistys? : Varkauden ja näpistyksen välinen rajanveto anastetun omaisuuden arvon ollessa vähäinen. Noudettu 14.10.2024 osoitteesta http://www.theseus.fi/handle/10024/820922

Scikit-Learn. (n.d). User Guide. Noudettu 10.10.2024, osoitteesta https://scikit-learn/stable/user_guide.html

Shilaskar, S., Bhalerao, S., Chakole, S., & Sharma, R. (2024). AI-Based Police Penal Code Identification Based on Keywords in FIR Report. 2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE), 1–6. Noudettu 26.10.2024 osoitteesta https://doi.org/10.1109/icETITE58242.2024.10493233

Tilastokeskus. (2024). Suomen virallinen tilasto (SVT): Rikos- ja pakkokeinotilasto. Noudettu 6.10.2024, osoitteesta https://stat.fi/tilasto/rpk

Tomar, M. S., & Gupta, V. (2023). Legal Case Classification Using Machine Learning with NLP. 2023 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI), 1–6. Noudettu 22.10.2024 osoitteesta https://doi.org/10.1109/ICDSAAI59313.2023.10452618

TurkuNLP. (2024). Turku-Neural-Parser-Pipeline. Noudettu 4.11.2024 osoitteesta http://turkunlp.org/Turku-neural-parser-pipeline/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). Attention Is All You Need. arXiv. Noudettu 18.10.2024 osoitteesta https://doi.org/10.48550/arXiv.1706.03762

Virtanen, A., Kanerva, J., Ilo, R., Luoma, J., Luotolahti, J., Salakoski, T., Ginter, F., & Pyysalo, S. (2019). Multilingual is not enough: BERT for Finnish. arXiv. Noudettu 29.10.2024 osoitteesta https://doi.org/10.48550/arXiv.1912.07076

Vysotska, V., Mazepa, S., Chyrun, L., Brodyak, O., Shakleina, I., & Schuchmann, V. (2022). NLP Tool for Extracting Relevant Information from Criminal Reports or Fakes/Propaganda Content. 2022 IEEE 17th International Conference on Computer

Sciences and Information Technologies (CSIT), 93–98. Noudettu 29.9.2024 osoitteesta https://doi.org/10.1109/CSIT56902.2022.10000563

Liitteet

Liite 1. Kyselytutkimuksen ohjeistus vastaajalle

Ohjeet vastaajalle

Kyselyn vastausnäkymässä pyydetään kirjoittamaan näennäinen selostus tapahtuneesta rikoksesta. Tutkielmassa rikosnimikkeiksi on rajattu varkaus, näpistys ja petos. Lisärajauksena on varmaa, että kussakin näennäisilmoituksessa rikos on tapahtunut. Oleellista on myös se, että rikos on tapahtunut Suomessa, ja ilmoitus on suomenkielinen. Valitse monivalintakysymyksessä mihin rikosnimikkeeseen selostuksesi mielestäsi kuuluu, ja kirjoita itse selostus vapaaseen tekstikenttään.

Voit siis itse kuvitella tai keksiä rikoksen tapahtumasarjan, ja kirjoittaa siitä selostuksen ikään kuin olisit tekemässä rikosilmoitusta. Voit itse päättää selostuksen tarkkuudesta ja tiedoista. Älä miettiessäsi rikosta tai sen tapahtumasarjaa ajattele, ettet itse voisi joutua vastaavaan tilanteeseen.

Ethän kirjoita selostukseen oikeita henkilötietoja tai muuta arkaluonteista tietoa itsestäsi tai muista. Voit halutessasi sisällyttää ilmoitukseen kuvitteellisia henkilöitä tai tietoja, jota sisällyttäisit oikeaan rikosilmoituksen selostukseen.

Vastausnäkymän ohjeistus

Valitse monivalintakysymyksessä mihin rikosnimikkeeseen selostuksesi mielestäsi kuuluu, ja kirjoita itse selostus vapaaseen tekstikenttään.

Voit siis itse kuvitella tai keksiä rikoksen tapahtumasarjan, ja kirjoittaa siitä selostuksen ikään kuin olisit tekemässä rikosilmoitusta. Voit itse päättää selostuksen tarkkuudesta ja tiedoista. Älä ajattele rikosta tai sen tapahtumasarjaa miettiessäsi, ettet juuri itse voisi joutua vastaavaan tilanteeseen.

Muista, että voit itse päättää mitä tietoa pystyt ilmoituksessa antamaan. Rikos on kuitenkin varmasti tapahtunut. Rikosilmoituksista voi puuttua jotain ratkaisevia tai oleellisia tietoja, joten selostusta kirjoittaessa voi "vahingossa" unohtaa jotain tärkeitä asioita, joita itselle ei tule mieleen tai joita on noloa paljastaa.

Käytä selostuksessa ilmaisumuotoja ja sanoja**,** jotka ovat sinun mielestä sopivia, mutta ota huomioon selostuksen kuvitteellinen asiayhteys (Selostus tapahtuneesta rikoksesta, josta teet rikosilmoitusta). Kirjoita selostus suomeksi.

Liite 2. Kuratoidun vertaidulatan määritykset

Termi	Määritelmä	Tunnusmerkit	Soveltuvuus
Laatumääritelmä: Kattava selostus	100-300 sanaa. Selostus on erittäin yksityiskohtainen ja kattava. Mukana on tarkka kuvaus tapahtumista, ajankohdasta, paikasta ja henkilöiden tuntomerkeistä. Ilmoittaja kertoo tapahtumien kulun selkeästi.	Tarkka kuvaus tapahtumasta. Sisältää keskeiset yksityiskohdat. Ilmoittaja varma tapahtumien kulusta.	Laatumääritelmien avulla voidaan arvioida, miten tiedon määrä vaikuttaa luokittelupäätöksen tarkkuuteen.
Laatumääritelmä: Normaali selostus	50-100 sanaa. Selostus on kohtalaisen kattava ja antaa perustiedot tapahtumasta, mutta joitakin tarkkoja yksityiskohtia saattaa puuttua. Selostus on riittävän selkeä rikosnimikkeen määrittämiseen.	Perustiedot tapahtumasta, voivat olla epätarkkoja. Joitain tärkeitä yksityiskohtia saattaa puuttua. Selostus on johdonmukainen.	Laatumääritelmien avulla voidaan arvioida, miten tiedon määrä vaikuttaa luokittelupäätöksen tarkkuuteen.
Laatumääritelmä: Suppea selostus	10-50 sanaa. Selostus on lyhyt ja yksityiskohdat ovat vähäisiä. Tärkeimmät tiedot tapahtuneesta, kuten mitä tapahtui, on mainittu, mutta muita yksityiskohtia, kuten ajankohta, paikka tai tekijän tuntomerkit, puuttuu tai ne ovat epämääräisiä.	Tarkat yksityiskohdat puuttuvat. Ei tarkkoja tuntomerkkejä tai lisätietoja. Ei tarkkoja tuntomerkkejä tai lisätietoja. Selostuksessa tulkinnanvaraisuutta.	Laatumääritelmien avulla voidaan arvioida, miten tiedon määrä vaikuttaa luokittelupäätöksen tarkkuuteen.
Tyyppimääritelmä: Synonyymi	Selostukset kuvaavat samaa tapahtumasarjaa, käyttäen eriäviä ilmaisumuotoja ja sanojen synonyymejä.	Nimikekohtainen. Sama ilmoitus uudelleenkirjoitettu eriävästi. Käytetty eriäviä laatumääritelmiä.	Voidaan arvioida mallien kykyä tehdä yhtenäinen luokittelupäätös sanaston vaihtu-
Tyyppimääritelmä: Erilaiset	Jokainen selostus kuvaa täysin uudenlaista tapahtumasarjaa.	Nimikekohtainen. Tapahtumasarja uniikki. Käytetty eriäviä laatumääritelmiä.	Voidaan arvioida mallien luokittelutarkkuutta yleisesti.
Tyyppimääritelmä: Samanlaiset	Selostuksissa pyritty käyttämään samoja sanoja ja lauserakennetta riippumatta rikosnimikkeestä. Sanoja muutettu niin, että selostus kuitenkin kuvaa rikosnimikkeen rikosta.	Nimikkeestä riippumatta samankaltainen. Selostuksien kesken paljon samoja sanoja. Käytetty eriäviä laatumääritelmiä.	Voidaan arvioida mallien kykyä hahmottaa luokat sisällön ollessa keskenään samankaltainen.