Selitettävä tekoäly luonnollisen kielen käsittelyssä ja kuvantunnistuksessa

Amanda Myntti

Pro gradu -tutkielma
Kesäkuu 2023

Tarkastajat:
Prof. Ion Petre
Prof. Marko Mäkelä

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

AMANDA MYNTTI: Selitettävä tekoäly luonnollisen kielen käsittelyssä ja kuvantunnistuksessa
Pro gradu -tutkielma, 56 s.
Matematiikka
Kesäkuu 2023

Tutkielmassa tutustutaan koneoppimismallien selitettävyyteen ja selitettävään tekoälyyn (eng.*explainability, explainable AI, XAI*) ohjatun oppimisen kontekstissa. Selitettävyyttä lähestytään sekä kvalitatiivisesti että matemaattisesta näkökulmasta, painottuen erityisesti post-hoc tyylisiin lokaaleihin selityksiin lajittelumalleissa. Lisäksi tarkastellaan luonnollisen kielen käsittelyn (eng.*natural language processing, NLP*) sovellusta, jossa selitysalgoritmia käytetään uuden tiedon eristämiseen tekstiaineistosta.

Tutkielma alkaa kirjallisuuskatsauksella selitettävien koneoppimismallien historiaan, erilaisiin selityskategorioihin ja nykyajan tarpeisiin luoda selitettäviä koneoppimismalleja sekä selittää jo olemassaolevia malleja. Seuraavaksi määritellään ohjatun oppimisen terminologiaa ja neuroverkkojen toiminnan perusteet sekä käsitellään luonnollisen kielen käsitellyssä käytettyjen mallien toimintaa.

Teoriapohjan rakentamisen jälkeen määritellään kolme erilaista lokaalia post-hoc selitysmenelmää, kerroksittainen relevanssi (eng.*Layerwise Relevance Propagation*), integroidut gradientit (eng. *Integrated Gradients*) sekä peittoanalyysi (eng.*Occlusion Analysis*). Lopuksi tarkastellaan sovellusta, jossa integroidut gradientit -selitysmenetelmää käytetään monirekisterisen tekstiaineiston avainsanojen määrittelemiseen rekistereittäin.

Asiasanat: koneoppiminen, selitettävä tekoäly, selitettävä koneoppiminen, XAI, selitettävyys, selitysalgoritmit, syväoppiminen, luonnollisen kielen käsittely, NLP.

Sisällys

1 Johdanto

Tekoäly tekee päätöksiä elämässämme kasvavissa määrin [1]. Olemmekin tottuneet sen käyttöön mm. kohdennetussa mainonnassa ja henkilökohtaisesti räätälöidyissä viihdesuosituksissa. Kuitenkin kriittisemmissä päätöksissä tekoälyn käyttöä vältellään, sillä usein sen toiminta on varsin läpinäkymätöntä. Tämä pätee erityisesti syviin koneoppimismalleihin [33]. Onkin väitetty, että käytetyn tekoälyn päätöksenteon monimutkaistuessa, sen saavuttaman suorituskyvyn ja tulkittavuuden välillä on aina tuleva olemaan kompromissi [34]. Selitettävä tekoäly (eng.*explainable AI, XAI*) on tästä huolesta syntynyt tieteenala, joka pyrkii luomaan ymmärrettävämpää tekoälyä säilyttäen nykyisen korkean suorituskyvyn sekä selittämään jo olemassa olevaa tekoälyä [1],[49].

Koneoppimismallia, jonka toiminta kätkeytyy monien kerrosten ja epälineaaristen operaatioiden taakse, kuvataan englanniksi sanalla *black box*. Tällaisten mallien toiminnan kuvaileminen ja sen tekemien päätösten perusteleminen on ongelma, joka riippuu mm. käytetystä mallista, kontekstista sekä selittäjän ja yleisön asiantuntijuudesta. Tästä syystä mallin selityksen ja selitettävyyden täsmällinen määrittäminen on vaikeaa [22]. Esimerkiksi artikkelin [33] mukaan selittäminen on koneoppimisen alalla niin huonosti määritelty ja monitahoinen asia, että sen käyttämistä voidaan jossain konteksteissa pitää tieteen periaatteiden vastaisena ja selitettävyyttä tulisikin tarkastella useana erillisenä XAI:n alaisena osaongelmana. Artikkelissa [27] selitettävyys määritellään olemaan sellaista tietoa mallin toiminnasta, joka ei pyri valaisemaan mallin sisäisiä rakenteita vaan sen toimintaa yleisellä tasolla, kun taas artikkelissa [22] se määritellään korkealla abstraktiotasolla systeemiksi, joka kuvaa mallin toimintaa tasapainotellen matemaattisen tarkkuuden ja tulkinnan helppouden välillä.

Selityksen määritelmä riippuu myös mallin koulutuksen kysymysasettelusta. Ihmisten kyky määrittää yksiselitteisesti paras kysymys kullekin mallille ja aineistolle on puutteellinen. Kysymysasettelusta riippuen selitys voi olla jotain, joka tuo korjaavaa lisäarvoa mallin toimintaan ja joka parantaa mallin ja ihmisen välistä dialogia [50]. On tärkeää myös huomioida, että ihmisten konsepti selitettävyydestä ei ole yksikäsitteinen, sillä esimerkiksi asiantuntijatkaan eivät aina pysty selittämään päätöksiään tai pääsemään yhteisymmärrykseen siitä, mikä annetuista selityksistä on paras [33]. Myös artikkeli [26] huomauttaa, että ihmiset suosivat usein erheellisesti yksinkertaisia ja nopeasti tulkittavia selityksiä informatiivisten selitysten sijaan eivätkä pysty tunnistamaan omia sisäisiä harhojaan ja ennakkoluulojaan analysoidessaan selityksen laatua. Artikkelien [20] ja [22] mukaan selitysten tuottaminen on aina kompromissi niiden tulkittavuuden ja niiden informatiivisuuden välillä.

Vielä 2010-luvun alussa on usein todettu syväoppimismallien olevan selitettävyyden saavuttamattomissa [27], [33]. Esimerkiksi lineaarisia malleja astetta monimutkaisempia malleja on pyritty selittämään sääntölähtöisesti [38] (artikkeli vuodelta 2009) eli eristämään niiden tekemät operaatiot ihmiselle seurattavaan kulkukaavioon. Artikkelissa [34] (artikkeli vuodelta 2012) selitettävyys määritellään tarkoittamaan sitä, että mallin tulkitsija pystyy päättelemään suoraan, mikä vaikutus päätökseen

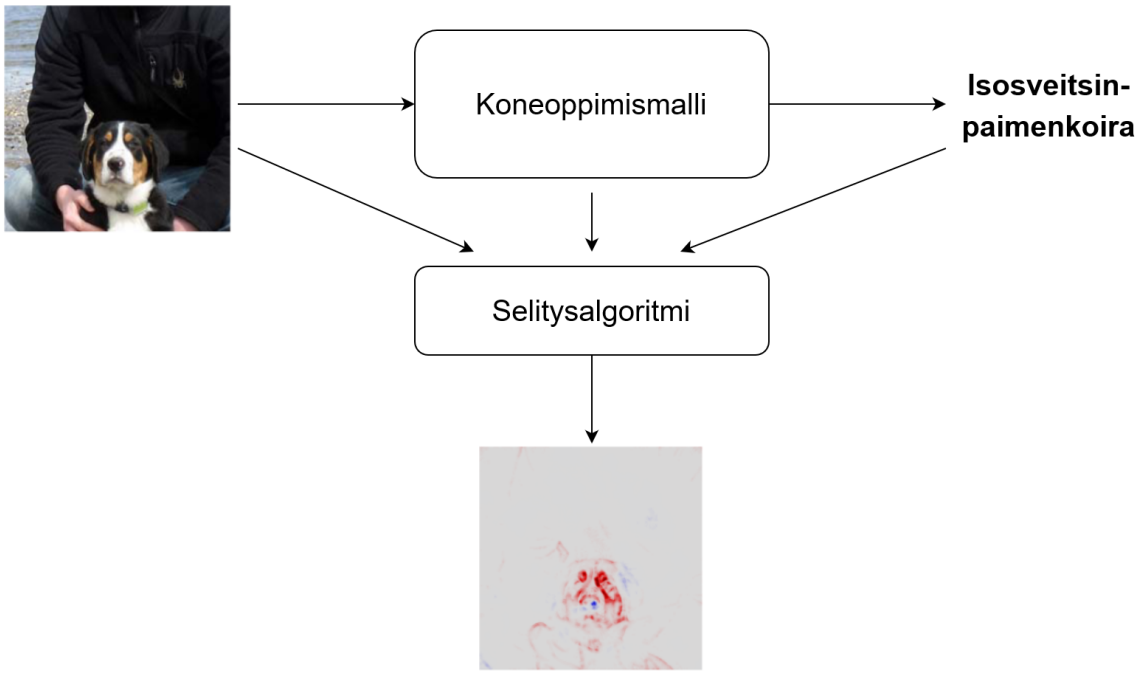
kullakin datapisteen piirteellä on. Tällainen selitettävyyden määritelmä sulkee ulos mallit, joissa piirteiden erilaiset yhdistelmät ovat olennaisessa roolissa. Nykyaikana on kuitenkin selvää, että syväoppimismallien suorituskyky yksinkertaisiin lineaarisiin malleihin verrattuna on niin huomattava, että niiden käyttämättömyydelle ei riitä perusteluksi ihmiselle epäintuitiivinen päätöksenteko.

Koneoppimisen tutkimuksen piirissä ollaan kuitenkin yksimielisiä, että pelkkä tarkkuuden mittaaminen ei ole hyväksyttävä ehto mallien käyttöönottamiselle [49], [30]. Esimerkiksi artikkelissa [13] tarkastellaan julkaisua 1990-luvulta [3], jossa koneoppimista käytettiin ennustamaan sairaalapotilaiden prognoosia. Vaikka saadut tulokset olivat hyviä, käytetty koneoppimismalli oli oppinut astman parantavan potilaan selviytymistodennäköisyyksiä. Todellisuudessa selviytymistä edistävä tekijä oli astmaan liittyvä aggressiivisempi hoito. Tällaiseen korrelaation ja kausaliteetin sotkemiseen ovat syynä artikkelin [33] mukaan ihmisten ongelmat kysymyksenasettelussa sekä tilanteet, joissa malli otetaan käyttöön sovelluksiin, joihin sen oletetaan virheellisesti sopivan.

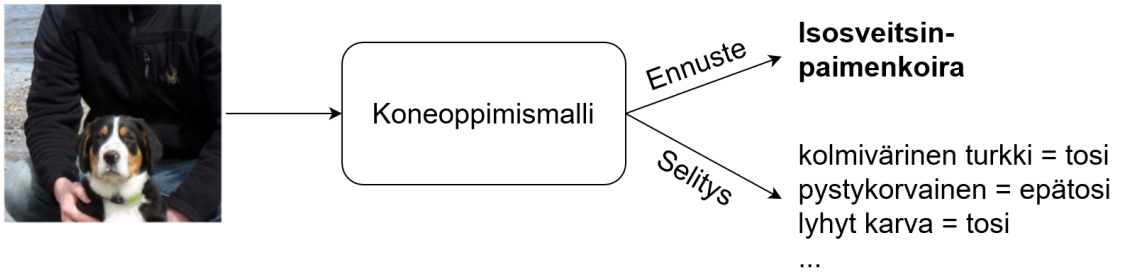
Artikkelin [1] mukaan koneoppimisen selittäminen on tärkeää neljästä eri syystä. Ensiksi selitykset auttavat korjaamaan mallien toimintaa, sillä selitysten avulla pystytään huomaamaan tehokkaasti virheitä, jotka voisivat muuten jäädä havaitsematta. Toiseksi selitysmenetelmiä käyttämällä voidaan perustella mallin toimintaa käyttäjille ja varmistaa, että ne täyttävät niille asetetut lailliset puitteet. Esimerkiksi Euroopan unionin GDPR-asetuksessa (eng. *General Data Protection Regulation*) datan käsittelijöitä velvoitettiin tarjoamaan asiakkailleen informaatiota siitä, miten heitä koskevia henkilötietoja käytetään algoritmeihin perustuvassa päätöksenteossa [24]. Kolmanneksi selitysmenetelmät tarjoavat tavan parantaa tulevia malleja tarjoamalla erilaisia näkökulmia tekoälyn toimintaan, jolloin sen vahvuudet ja heikkoudet tulevat paremmin tunnetuiksi. Neljänneksi selityksiä voidaan käyttää uuden tiedon eristämiseen systeemistä, josta esimerkki esitellään tutkielman luvussa 4. Artikkelissa [12] edellä mainittujen syiden lisäksi mainitaan mahdollisuus parantaa mallin soveltuvuuden arviointia, kun malli otetaan käyttöön eri ympäristössä kuin mihin se on luotu.

Selittämällä voidaan vaikuttaa myös ihmisten luottamukseen koneoppimismallin tuloksiin. Kuten edellä todettiin, yksinkertaisia koneoppimismalleja on suosittu aiemmin tulkittavuutensa takia, etenkin lääketieteessä [27]. Artikkeli [27] jatkaa, että mikäli koneoppimiseen perustuvaa järjestelmää käytettäisiin, tulee sen käyttäjien pystyä luottamaan sen antamiin tuloksiin. Jos järjestelmään ei luoteta, tullaan sen antamat tulokset ohittamaan tarkkuudesta huolimatta.

Selitysmenetelmät voidaan jakaa kahdella erillisellä akselilla: globaaleihin ja lokaaleihin sekä ante-hoc ja post-hoc selitysmenetelmiin [49], [27]. Tarkastellaan seuraavaksi kutakin kategoriaa, minkä jälkeen tutkielma keskittyy lokaaleihin post-hoc selityksiin. Kaikissa esimerkeissä koneoppimismalli, jota merkitään funktiolla F, on koulutettu lajittelemaan aineistoa, esimerkiksi tekstiä tai kuvia. Tällöin mallin antaman tulosteen voidaan sanoa kertovan, kuinka paljon "todistusaineistoa" mallilla on väittää annetun syötteen, x, kuuluvan johonkin luokkaan [49], jonka perusteella



Kuva 1: Visualisointi lajittelumallin ja lokaalin post-hoc selitysalgoritmin toiminnasta artikkelin [4] mukaisesti. Ensimmäisessä vaiheessa malli tuottaa ennusteen annetulle syötteelle, jonka jälkeen syöte, malli ja ennuste annetaan selitysalgoritmille. Selitysalgoritmi tuottaa visualisaation lämpökarttana, joka selittää mallin toimintaa. Visualisoinnissa punaisella on esitetty ne pikselit, jotka tukivat ennustetta "isosveitsinpaimenkoira", ja sinisillä ne jotka olivat ennustetta vastaan. Valokuva [45] ja lämpökartta [49].



Kuva 2: Visualisointi lokaalista ante-hoc selityksen muodostamisesta lajittelumallille. Ennuste ja selitys tuotetaan samanaikaisesti. Koneoppimismalli on valittu siten, että selityksen eristäminen siitä on helppoa ja se voidaan esittää esimerkiksi päätöspuuna tai listana ominaisuuksia, joihin päätös perustui. Valokuva [45].

lasketaan lopulle päätös, eli ennuste. Kun datapisteitä luokitellaan useampaan luokkaan, tuloste $F(x)$ on vektori, joka sisältää jokseenkin todennäköisyyksiä vastaavat arvot, että annettu syöte x kuuluu kuhunkin luokkaan.

Globaalit selitysmenetelmät pyrkivät selittämään mallin toimintaa isossa mittakaavassa ja pyrkivät antamaan käyttäjälle yleisen kuvan mallin toiminnasta. Globaali selitys voidaan antaa muun muassa esimerkkien muodossa. Yksi globaali selitysmenetelmä on aktivointimaksimointi [18]. Aktivointimaksimoinnissa koneoppimismallista tuotetaan kunkin luokan "malliesimerkki". Jos analysoitavana on malli, joka tunnistaa annetuista kuvista eläimiä, aktivointimaksimointi voi tuottaa jokaisesta mallille opetetusta luokasta maksimaalisesti luokkaa edustavan kuvan. Esimerkiksi jos kohteena on luokka "koira", aktivointimaksimointi kokoaa mallin mukaan kaikkein "koiraisimman" mahdollisen kuvan. Formaalisti määriteltynä aktivointimaksimointia voidaan kuvata luokalle i kaavalla [49]

$$x_i^* = \arg\max_x F(x)_i.$$

Esimerkkejä aktivointimaksimoinnilla tuotetuista kuvista on esitelty artikkelissa [36]. Globaali selitys voidaan myös antaa listana esimerkkejä, jotka on tuotettu datapisteillä, jotka edustavat jakaumaltaan mallin käytössä saamia syötteitä [12].

Lokaalit selitysmenetelmät keskittyvät yhteen syötteeseen liittyvien päätösten selittämiseen [49], [20]. Lokaali selitys voi olla esimerkiksi visualisointi siitä reitistä, jonka syöte kulki päätöspuussa tai lämpökartta kuvasta, joka näyttää mihin pikseleihin päätös voimakkaimmin perustui. Lokaalit selitykset ovat voimassa vain annetun syötteen välittömässä läheisyydessä [12], mutta ne mahdollistavat muun muassa GDPR:n vaatimien selitysten tarjoamisen yksityishenkilöille. Lokaaleja selityksiä voidaan myös keskiarvoistaa globaalimmaksi selitykseksi, mikä nähdään luvussa 4. Vaikka lokaalit ja globaalit selitykset toimivat eri lähtökohdista, on tärkeä huomata, että koneoppimismallin tekemä päätös johtuu yleensä sekä globaaleista että lokaaleista tekijöistä [49].

Post- ja ante-hoc selitykset eroavat toisistaan siten, että post-hoc selitys muodostetaan sen jälkeen, kun malli on tuottanut ennusteen ja ante-hoc selitykset samalla, kun ennuste muodostetaan [27]. Vertaus post- ja ante-hoc selityksistä on esitetty kuvissa 1 ja 2. Ante-hoc selitykset vaativat mallin olevan rakenteeltaan sellainen, että siitä on mahdollista muodostaa selitys. Tyypillisiä esimerkkejä tällaisista malleista ovat päätöspuut ja regressiomallit [12]. Post-hoc selitysten muodostaminen perustuu mallin analysointiin erilaisilla selitysalgoritmeilla eikä vaadi mallin rakenteelta eksplisiittistä selitettävyyttä. Tietyissä tapauksissa edes pääsyä mallin lähdekoodiin ei tarvita, kuten luvussa 3 nähdään.

Tutkielma keskittyy tarkemmin post-hoc tyylisiin lokaaleihin selitysmenetelmiin. Luvussa 2 käydään läpi tarvittavat konseptit liittyen koneoppimiseen ja luonnollisen kielen käsittelyyn, luvussa 3 esitellään kolme lokaalia post-hoc tyylistä selitysalgoritmia ja vertaillaan niiden toimintaa. Luvussa 4 tarkastellaan selitysmenetelmään perustuvaa algoritmia luonnollisen kielen käsittelyn sovelluksessa. Sovellus perustuu artikkeliin [46], jonka työstämisessä olen ollut itse mukana.

2 Koneoppimisesta

Koneoppimisella viitataan tekoälyn osa-alueeseen, jossa algoritmeja muodostetaan datan avulla [19]. Perinteisessä algoritmiikassa tavoitteena on määritellä jokaiselle mahdolliselle syötteelle (eng.*input*) sitä vastaava tuloste (eng.*output*). Algoritmin voi ajatella olevan kuin funktio, joka kuvaa syötejoukon tulosteiksi. Tällaisen funktion määrittely on kuitenkin useissa reaalimaailman tilanteissa hankalaa, sillä algoritmin luominen vaatii yksityiskohtaista ymmärrystä halutusta yhteydestä syötteen ja tulosteen välillä. Koneoppimisessa sitä vastoin määritellään prosesseja, jotka tuottavat saamastaan datasta algoritmeja. Oppimisprosesseja on esitetty lukuisia erilaisiin ongelmiin ja käyttökohteisiin ja näitä prosesseja nimitetään malleiksi (eng.*model*). Mallien koulutukseen kuuluu arviointivaihe, jossa varmistetaan algoritmin rakentuvan optimaaliseen suuntaan [19]. Optimointivaiheen myötä algoritmin antamat tulosteet ja tulosteiden perusteella muodostetut ennusteet (eng.*prediction*), vastaavat kasvavassa määrin käytetyn datan mukaisia arvoja. Tällöin sanotaan mallin *oppivan*.

Koneoppiminen jaetaan kirjallisuudessa kolmeen eri osa-alueeseen, jotka eroavat oppimisessa käytetyn datan perusteella [51]. Ohjattu oppiminen (eng.*supervised learning*) perustuu oppijan tekemiin hypoteeseihin saamiensa syötteiden ja niitä vastaavien tulosteiden yhteydestä. Ohjaamattomassa oppimisessa (eng.*unsupervised learning*) oppija ei näe syötettä vastaavia tulosteita, joten ohjaamattomat mallit pyrkivät tekemään luokitteluja saamastaan syötteestä samankaltaisuuden perusteella. Esimerkki ohjaamattomasta oppimisesta on klusterointi (eng.*clustering*), jossa oletuksena on, että syötteiden välisen etäisyyden pienentyessä niitä vastaavat tulosteet ovat yhä todennäköisemmin samat. Kolmas yleisesti tunnistettu koneoppimisen tyyppi on vahvistusoppiminen (eng.*reinforcement learning*). Vahvistusoppimisessa käytetään tulosteetonta dataa, josta oppija oppii mallista erillisen palkitsijan antaman palautteen avulla.

Tässä tutkielmassa keskitytään ohjattuun oppimiseen. Ohjattu oppiminen jakautuu vielä karkeasti sanottuna kahteen päähaaraan, regressio-ongelmiin ja lajitteluongelmiin (eng.*classification*), joissa aineistoa lajitellaan ennalta määriteltyihin kategorioihin [19]. Tässä luvussa käytetään merkintöjä, jotka parhaiten sopivat lajitteluongelmiin, mutta samat määritelmät pätevät myös regressio-ongelmiin pienin muutoksin.

2.1 Ohjattu oppiminen

Ohjatussa oppimisessa käytetty data muodostuu järjestetyistä pareista syötteitä x ja niitä vastaavia tulosteita r. Pareja kutsutaan datapisteiksi (eng.*data point*). Syöte on yleensä vektorimuodossa ja sen alkiot ovat tutkittavasta systeemistä mitattuja arvoja tai esimerkiksi näistä arvoista laskettuja tilastollisia suureita. Näitä arvoja kutsutaan piirteiksi (eng.*features*), ja vektoriin x viitataan usein piirrevektorina (eng.*feature vector*). Tuloste on yleensä skalaari, joka kertoo mitä tyyppiä kyseinen datapiste edustaa. Tulosteen tyyppiä kutsutaan luokaksi tai kategoriaksi (eng.*label*). Moniluokkalajittelussa, eli tilanteessa, jossa mahdollisia kategorioita on useita, tuloste voi olla myös pari tai lista kategorioita, jos datapiste edustaa useampaa luokkaa

samanaikaisesti. Datapisteiden joukko voidaan täten esittää muodossa [19]

$$\mathcal{X} = \{x_k, r_k\}_{k=1}^{N},$$

missä N on datan kardinaliteetti eli datapisteiden lukumäärä.

Joukko $\mathcal{X}$ jaetaan yleensä koulutus- tai harjoitusdataan (eng.*training data*) ja validointidataan (eng.*validation data*) [37]. Nimensä mukaisesti koulutusdataa käytetään mallin koulutukseen ja validointidataa mittamaan, kuinka hyvin koulutus onnistui. Jako tehdään tilannekohtaisesti, mutta on yleistä varata koulutusdataksi suurempi joukko datapisteitä, sillä suuremmalla koulutusdatalla saavutetaan yleensä paremmat oppimistulokset. Kaiken datan käyttö koulutuksessa ei ole kuitenkaan sopivaa, sillä tällöin malli oppii kuvailemaan pelkästään dataa $\mathcal{X}$, eikä välttämättä sitä populaatiota, josta $\mathcal{X}$ on otanta.

Koska koulutusta joudutaan usein koettamaan useamman kerran, ennen kuin malli suoriutuu validointidatalla halutusti, vaikuttaa validointidata myös mallin koulutukseen. Esimerkiksi mallin koulutukseen liittyvien parametrien säätäminen koulutusyritysten välillä saa mallin riippuvaiseksi niistä datapisteistä, jotka valikoitiin jaossa validointidataan. Tällöin datasta voidaan jättää pieni osa eli raportointi- tai testidata (eng.*test data, testing data, publication data*) pelkästään lopullisten tulosten raportointia varten [19]. Datan jako ja mallin tulosten riippumattomuus voidaan tehdä myös toisin, muun muassa ristiinvalidoinnilla (eng.*cross-validation*) [37]. Vaikka edellä esiteltyjen syiden takia datan jako kolmeen joukkoon on suositeltava menettelytapa koneoppimismallien koulutukseen, tämän tutkielman kattaman teoriasisällön tarkoituksiin jakoa kolmeen joukkoon ei tarvita. Täten tästä eteenpäin koulutukseen käytettyyn dataan viitataan koulutusdatana $\mathcal{X}_h$ ja tulosten laskemiseen käytettyyn dataan testidatana $\mathcal{X}_t$.

Kuten edellä todettiin, ohjattu oppiminen perustuu oppimiseen sekä piirrevektoreista että luokkavektoreista. Yksinkertaistetusti ohjattu oppiminen perustuu seuraavaan prosessiin:

- mallille annetaan koulutusdatan piirrevektori tai joukko piirrevektoreita x,
- malli antaa ennusteen $\hat{r}$ perustuen vektoriin x,
- malli vertaa luokkaa r ennusteeseen $\hat{r}$,
- mikäli ennuste ei vastaa luokkaa, muutetaan mallin toimintaa,
- jatketaan, kunnes malli suoriutuu toivotulla tavalla tai suoritus ei enää parane.

Yllä annetussa suurpiirteisessä algoritmissa viitataan ensimmäisessä askeleessa joukkoon piirrevektoreita, mikä onkin yleinen tapa antaa syötteitä. Eräkoko (eng.*batch size*) on useiden koneoppimismallien koulutuksen parametri, jonka avulla kontrolloidaan, montako piirrevektoria mallille annetaan ennen mallin optimointivaihetta. Pieni erä piirrevektoreita on laskennallisesti kevyempää, mutta suurempi erä mahdollistaa tarkemman optimoinnin, sillä mallin tekemästä virheestä tiedetään tällöin enemmän [37].

Seuraavat kappaleet perustuvat kirjan [19] määritelmiin. Vaikka mallin koulutukseen valittua käsittelemätöntä aineistoa voidaan joskus käyttää suoraan koneoppimismallin syötteenä, on sitä usein tarpeellista esikäsitellä. Käsittelemättömän aineiston saattamista muotoon, joka voidaan syöttää koneoppimismallille, kutsutaan piirreirrotukseksi (eng.*feature extraction*). Piirreirrotuksen tuloksena käsittelemätön aineisto muuttuu mallille syötettäviksi datapisteiksi $\mathcal{X}$.

Piirreirrotus voi yksinkertaisimmillaan olla niiden piirteiden valintaa, joiden uskotaan parhaiten auttavan mallin oppimista. Esimerkiksi sekä vuositulojen ja veroprosentin sisältävästä datasta on järkevää valita vain vuositulot, koska veroprosentti voidaan päätellä vuosituloista, eikä sen antaminen mallille siten todennäköisesti paranna oppimista. Monimutkaisemmassa tapauksessa, kuten kuvantunnistuksessa, voidaan kuvista laskea esimerkiksi värejä ja kontrastia kuvaavia tilastollisia suureita. Kuvantunnistukseen ja luonnollisen kielen käsittelyyn (eng.*Natural Language Processing, NLP*) on myös kehitetty monimutkaisempia, piirreirrotuksen kaltaisia mallien sisäisiä prosesseja.

Myös tulosteosalle täytyy usein tehdä jonkinlaista käsittelyä. Binäärisessä tapauksessa usein riittää luokan koodaaminen bitiksi $\{0, 1\}$. Mikäli aineisto on moniluokkainen, koodausta kokonaisluvuiksi ei voida kuitenkaan tehdä, sillä koodaaminen numeroiksi $\{1, 2, \ldots, k\}$ luo jonkinnäköisen järjestyksen kategorioiden välille: tällöin mallin käsityksen mukaan kategoria 2 on kauempana kategoriasta 6 kuin kategoriasta 3, vaikka datan perusteella näin ei olisi. Yksi tapa ratkaista ongelma on *one-hot* koodaus [19]. One-hot koodauksessa moniluokkaisen tulosteen r luokat numeroidaan ja jokainen muutetaan binääriseksi vektoriksi, jossa luokan indeksin mukainen alkio asetetaan arvoksi 1 ja muut indeksit nollaksi. One-hot koodaus mahdollistaa myös kahden samaan aikaan esiintyvän luokan esittämisen.

Tarkastellaan seuraavaksi, miten koneoppimismallin toimintaa voidaan arvioida. Merkitään koulutettavaa mallia, joka ottaa syötteenä piirrevektorin x ja laskee tulosteen ja sen perustella ennusteen $\hat{r}$ parametrien θ avulla

$$x \longmapsto F(x|\theta).$$

Jos malli F määritellään kaikille sille mahdollisille parametriyhdistelmille Θ, saadaan mallin F hypoteesijoukko $\mathcal{H}$ [19]. Jokainen hypoteesi joukosta $\mathcal{H}$ on mahdollinen yhteys syötteen ja tulosteen välillä ja oppimisen tavoitteena on löytää yhteyttä parhaiten tai tarpeeksi hyvin kuvaava hypoteesi. Mallin oppinen perustuu siis optimaalisen parametrijoukon $\theta' \in \Theta$ löytämiseen. Kaikki ohjatun oppimisen mallit eivät kuitenkaan ole parametrisia, eli perustu parametrien huolelliseen valintaan, jolloin hypoteesijoukkon määritelmän voidaan ajatella olevan kaikki mallin rajoissa mahdolliset yhteydet datan syötteen ja tulosteen välillä [19].

Tappiofunktiota $L(r, F(x|\theta))$ (eng.*loss function*) käytetään vertaamaan ennustetta ja datan antamaa luokkaa [37]. Tappiofunktio on tilanteeseen sopivasti valittu funktio, joka antaa pieniä arvoja, kun luokka ja ennuste ovat samat tai lähellä toisiaan ja suuria arvoja, kun eivät. Tappiofunktioksi voidaan määritellä mikä tahansa tilanteeseen sopiva metriikka. Mikäli tuloste on binäärinen, voi tappiofunktioksi riittää

yhtäsuuruustarkistus. Funktio voi olla myös painottunut joidenkin tiettyjen virheiden suhteen. Esimerkiksi lääketieteellisissä koneoppimissovelluksissa voi olla vaarallisempaa saada erheelinen negatiivinen kuin erheellinen positiivinen testitulos, jolloin tappiofunktion arvo voidaan asettaa suuremmaksi haitallisemmassa tapauksessa [3].

Kun tappiofunktion antamat arvot summataan yli erän syötteitä $B \subset \mathcal{X}_h$, saadaan virhe [19]

$$\mathcal{E}(\theta|B) = \sum_{(x,r)\in B} L(r, F(x|\theta)).$$

Suuri virheen arvo kertoo, että mallin pitää muuttua paljon ennen kuin se on löytänyt hypoteesin, joka kuvaa yhteyttä datan piirrevektorien ja luokkien välillä. Virheen perusteella edellä mainitun optimaalisen parametrijoukon θ' määritelmäksi saadaan [19]

$$\theta' = \arg\min_{\theta\in\Theta} \mathcal{E}(\theta|\mathcal{X}).$$

Miten virheen minimointi käytännössä tapahtuu, riippuu mallin rakenteesta.

Yllä määritelty virhe on suure, joka ohjaa mallin koulutusta. Virhe on tapana ilmoittaa jokaisen koulutuserän perään, mutta lopullisessa koulutustulosten raportoinnissa käytetään yleensä mallin tarkkuutta (eng.*accuracy*) testidatalla. Olkoon c indeksointi tulosteen r luokille, olkoon $(x, r) \in \mathcal{X}_t$ ja $\hat{r}$ vektorin $F(x)$ perusteella laskettu ennuste. Tällöin tarkkuus lasketaan kaavalla [37]

$$\text{tarkkuus} = \frac{|r = \hat{r}|}{|r|}.$$

Lisäksi määritellään kolme suuretta, jotka kuvaavat mallin suoriutumista testidatalla luokittain [37],

$$\begin{aligned}
\text{sisäinen tarkkuus}(c) &= \frac{|\{r = c\} \cap \{\hat{r} = c\}|}{|\{\hat{r} = c\}|}, \qquad (\text{eng.}\textit{precision}) \\
\text{herkkyys}(c) &= \frac{|\{r = c\} \cap \{\hat{r} = c\}|}{|\{r = c\}|}, \qquad (\text{eng.}\textit{recall}) \\
F1(c) &= \frac{2}{\frac{1}{\text{s. tarkkuus}(c)} + \frac{1}{\text{herkkyys}(c)}}.
\end{aligned}$$

Luokittainen arviointi on moniluokkalajittelussa keskeistä, sillä mikäli yhden luokan instansseja on selkeästi vähemmän kuin muita, voi malli saavuttaa hyvän tarkkuuden vain sillä, ettei se millekään syötteelle ennusta pienintä luokkaa. Tällöin kuitenkin luokkakohtaiset metriikat kertovat mallin toimivan epätoivotusti. $F1$-arvoa käytetään myös muissa solvelluksissa, kuten annotaattoreiden yhtenevyyden (eng.*interannotator agreement*) mittaamiseen [43].

2.2 Neuroverkot

Neuroverkoilla (eng.*neural networks*) tarkoitetaan koneoppimisen osa-aluetta, jossa koneoppimismalli pyrkii matkimaan ihmisaivojen neuronien toimintaa [51]. Aivot ovat erittäin taitavat esimerkiksi hahmontunnistuksessa, mikä on haluttu ominaisuus usealla eri teollisuuden kuin kaupallisellakin alalla. Määritellään neuroverkkojen peruskäsitteet yhdistettyihin soluihin perustuvalla mallilla, perseptroneilla.

Perseptroni (eng.*perceptron*) on yksi yksinkertaisimmista malleista neuronille. Se perustuu syötteen lineaariseen painotukseen ja siitä laskettavaan summaan, joka kuvataan käyttötarkoitukseen sopivalla aktivointifunktiolla (eng.*activation function*). Harjoitusdatan avulla painotustermit optimoidaan siten, että malli täyttää toivotut ominaisuudet [37]. Olkoon perseptroniin tuleva d-ulotteinen piirrevektori muotoa $x = [x_1, x_2, ..., x_d]^T$. Syötteet painotetaan vastaavilla kertoimilla $w = [w_1, w_2, ..., w_d]^T$, joita kutsutaan kytkentäpainoiksi (eng.*connection weights, synaptic weights*). Lopuksi lisätään vakiotermi b, jolloin perseptroniin tuleva kokonaissyöte on muotoa

$$y = \sum_{i=1}^{d} (x_i \cdot w_i) + b.$$

Koska kyseessä on lineaarinen kuvaus, voidaan se esittää myös vektoritulona

$$y = w^T x + b.$$

Vakiotermin b (eng.*bias*) lisääminen tekee kuvauksesta y affiinin. Näin varmistetaan, että esimerkiksi syötteestä $x \equiv 0$ saatava tulos ei ole automaattisesti $y = 0$, vaan riippuu termistä b. Koska mallin oppiminen perustuu kytkentäpainojen arvojen säätämiseen, on usein kätevää esittää vakiotermi muodossa

$$b = x_0 \cdot w_0,$$

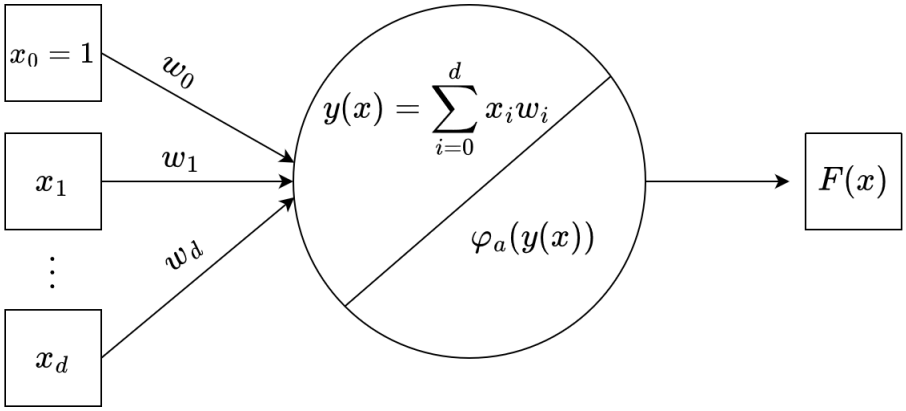
missä x_0 on kiinnitetty arvoon 1 ja painoa w_0 optimoidaan muiden painojen tavoin. Tällöin voidaan laajentaa vektorit x ja w sisältämään vakiotermin b

$$x = [1, x_1, x_2, \ldots, x_d]^T,$$
$$w = [w_0, w_1, w_2, \ldots, w_d]^T.$$

Painotettu summa y kuvataan aktivointifunktiolla φ_a, jolloin perseptronin antamaksi tulosteeksi saadaan

$$\varphi_a(y) = \varphi_a\left(w^T x + b\right).$$

Ihmisen hermosolu vie viestin seuraavalle solulle vain, jos tietty solulle ominainen raja-arvo viestin voimakkuudelle ylitetään [2]. Perseptronissa tällainen yksinkertainen aktivointifuntio on mahdollinen [19], mutta koulutusvaiheessa on hyötyä siitä,



Kuva 3: Perseptroni, jossa vakio b on esitetty muodossa $x_0 w_0$. Saapuvasta syötteestä $x = [1, x_1, \ldots, x_d]^T$ lasketaan ensin painotettu summa y, joka kuvataan aktivointifunktiolla $\varphi_a(y)$ ja palautetaan tulosteena $F(x)$.

että aktivaatiofunktio on differentioituva, mikä perustellaan myöhemmin tässä luvussa ja luvussa 3. Kun perseptroneja laitetaan useita peräkkäin, on myös mahdollista, että eri kerroksissa on käytössä eri aktivointifunktio. Etenkin viimeisessä kerroksessa on tavallista käyttää s-kirjaimen muotoista logistista funktiota (eng.*sigmoid, logistic function*) [37]

$$\mathrm{sigm}(y) = \frac{e^y}{e^y + 1},$$

tai K-luokkaisessa lajittelussa *softmax*-funktiota [37]

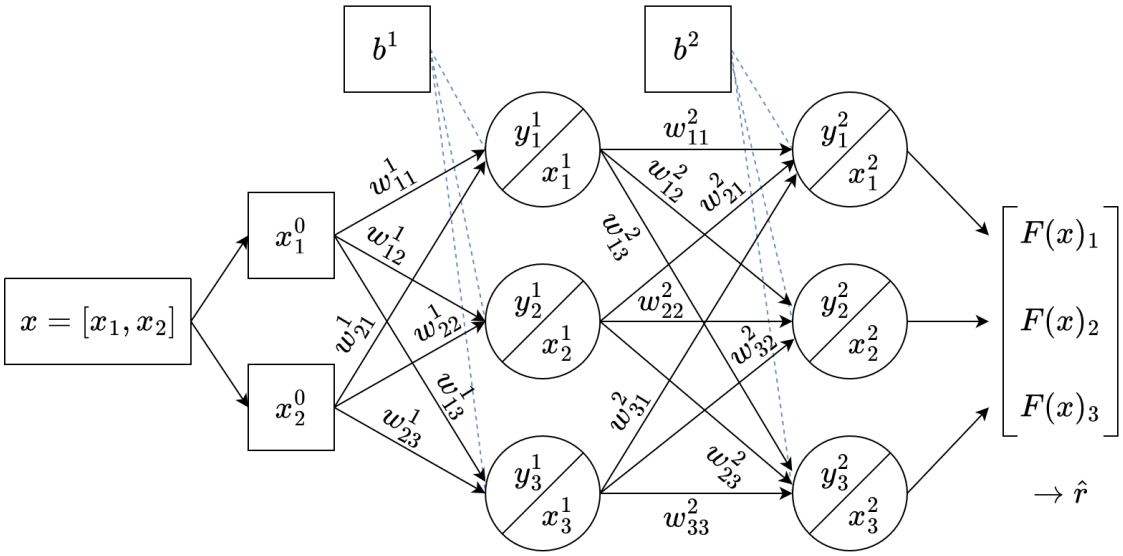
$$\sigma(y_i) = \frac{e^{y_i}}{\sum_{k=1\ldots K} e^{y_k}},$$

joka palauttaa tulosteen normalisoituna siten, että luokkia vastaavien arvojen summa on 1. Välikerroksissa voidaan käyttää esimerkiksi paloittain määriteltyä ReLu-funktiota (eng.*Rectified Linear Unit*) [23]

$$\mathrm{ReLu}(y) = \max(0, y),$$

joka on derivoituva kaikkialla paitsi pisteessä $y = 0$. Aktivoinnin tulos välitetään seuraavalle kerrokselle tai annetaan mallista ulos tulosteena.

Koska yksittäinen perseptroni pystyy vain matriisikertolaskuun sekä yhteen eilineaariseen operaatioon, joka kuvaa lineaarisen laskutoimituksen tulokseksi, sen oppimiskyvyt ovat rajalliset. Mallin kyvykkyys kasvaa, kun vierekkäin asetetuista perseptroneista kootaan peräkkäisiä kerroksia [19]. Tällöin puhutaan monikerroksisesta perseptronista (eng.*multilayer perceptron*). Jokainen edellisen kerroksen perseptroni on kytketty kaikiin seuraavan kerroksen perseptroneihin, jolloin kytkentäpainovektorin w sijaan puhutaan kykentäpainomatriisista W. Monikerrosmallin ensimmäistä kerrosta kutsutaan syötekerrokseksi (eng.*input layer*), viimeistä kerrosta tuloste- tai ennustekerrokseksi (eng.*output layer*) ja välissä olevia tasoja piilotetuiksi kerroksiksi (eng.*hidden layer*) [37].



Kuva 4: Visualisointi muuttujien nimistä monikerrosperseptronissa, joka palauttaa kolmearvoisen vektorin. Yläindeksi ilmoittaa kerroksen, jonka laskutoimitukseen kukin muuttuja osallistuu. Solujen tulosteiden x_i^l alaindeksit kertovat solujen numeron kerroksessa l, ja painojen alaindeksit implikoivat mistä solusta mihin soluun kukin kytkentä vaikuttaa.

Olkoon $l \in \{1, .., L\}$ L-kerroksisen monikerrosmallin kerroksen indeksi ja merkitään kerroksessa l olevia perseptroneja joukolla K^l. Käytetään syötekerroksesta indeksiä 0. Olkoon i indeksointi kerroksen K^{l-1} perseptroneille ja j kerroksen K^l perseptroneille. Tällöin kytkentäpainomatriisi, joka kytkee tasot K^{l-1} ja K^l, voidaan esittää muodossa [19]

$$W^l = \begin{bmatrix} w_{11}^l & w_{12}^l & \dots & w_{1j}^l \\ \vdots & \vdots & & \vdots \\ w_{i1}^l & w_{i2}^l & \dots & w_{ij}^l \end{bmatrix}$$

ja kerroksen l tulosteeksi saadaan rekursiivisesti

$$x^l = \varphi_a(y^l) = \varphi_a\left((W^l)^T x^{l-1} + b^l\right),$$

missä aktivointifunktio lasketaan pisteittäin, eli jokaisesta vektorin y^l alkiosta erikseen. Samoin kuin yksittäisessä perseptronissa, voidaan vakiotermien b^l painot lisätä matriisin W ensimmäiseksi riviksi, jolloin jokaisen kerroksen l tulosteeseen x^l lisätään myös ensimmäiseksi alkioksi 1. Monikerrosperseptronin painot on esitetty graafisesti kuvassa 4.

Tulostekerroksen rakenne määräytyy datan tulosteosan perusteella [37]. Binäärisessä tapauksessa yksi perseptroni tulostekerroksessa riittää, sillä aktivointifunktion palautusarvolle voidaan asettaa raja-arvo, jonka ylittyessä ennusteeksi saadaan toinen luokka ja alittuessa toinen. Kun koulutusdatassa mahdollisia luokkia on enemmän kuin kaksi, voidaan tulostekerrokseen valita yhtä monta perseptronia. Tällöin viimeisen kerroksen tuloste on vektori, jossa on yhtä monta alkiota kuin datassa olevia

luokkia, kuten kuvassa 4. Jokainen vektorin arvo kuvaa sitä, miten todennäköinen kukin luokka tälle syötteelle on mallin koulutuksessa saadun datan perusteella. Tämän takia usein mallin ennusteeksi valitaan esimerkiksi se luokka, jota vastaavan perseptronin ulosantama arvo on suurin, tai ne luokat, joita vastaavien perseptronien palauttama arvo ylittää jonkin rajan. Ennen päätöstä on myös tavallista kuvata viimeisen kerroksen tulostevektori aiemmin määritellyllä *softmax*-funktiolla $\sigma(y)$, jotta ennusteen tuottamiseen käytetyt ehdot voidaan valita yksiselitteisesti.

Viimeisenä huomiona neuroverkkojen rakenteesta todettakoon, että verkkorakenteisten koneoppimismallien ei tarvitse olla täysin kytkettyjä. Toisin sanoen on mahdollista, että kaikki kytkökset kerrosten välillä eivät ole aktiivisia, vaan osa kytköksistä on asetettu toimimattomiksi. Itseisarvoltaan hyvin pienipainoisten kytkösten poistaminen verkosta on myös itsessään optimointia käytettyjen laskenta- ja aikaresurssien suhteen [9]. Tässä tutkielmassa kuitenkin oletetaan, että kaikki käsiteltävät verkot ovat täysin kytkettyjä. Seuraavaksi esitelty ja luvun 3.1 algoritmi voidaan silti muokata muotoon, jossa kaikki kytkökset eivät ole aktiivisia.

Neuroverkkojen oppiminen perustuu aiemmin esiteltyyn yhtälöön [19]

$$\theta' = \arg\min_{\theta \in \Theta} \mathcal{E}(\theta|B),$$

missä B on erä datapisteitä ja joukko Θ sisältää kaikki mahdolliset parametriyhdistelmät kyseiselle mallille. Perseptronien yhteydessä parametrit θ ovat kytkentäpainot W.

Perseptronimallien kytkentäpainojen optimointiin sopiva valinta on gradienttimenetelmä (eng.*gradient descent*) [8], [37]. Kun erä syötteitä B viedään verkon läpi, niistä laskettu virhe $\mathcal{E}(W|B)$ on funktio verkon kytkentäpainoista W, joten on mahdollista tarkastella, mitä virheen arvoille tapahtuu, jos painoja muutetaan. Gradienttimenetelmässä virheen gradientti lasketaan kunkin kerroksen kytkentäpainojen suhteen. Painoja liikutetaan gradientin vastaiseen suuntaan, eli suuntaan, jossa virhe lokaalisti pienenee nopeiten. Lasku toistetaan seuraaville erille datapisteitä, joilloin usean iteraatiokerran aikana kytkentäpainomatriisien arvot suppenevat joihinkin virheen minimoiviin arvoihin.

Gradientin määritteleminen jokaiselle kerrokselle on laskennallisesti työlästä. Siksi gradienttimetelmän nopeuttamiseksi käytetään usein vastavirta-algoritmia (eng.*backpropagation*), joka esitteltiin ensimmäisenä neuroverkkojen yhteydessä artikkelissa [44]. Aikaisemmin samanlaisia, mutta eri tarkoituksiin kehitettyjä algoritmeja on esitelty muun muassa artikkeleissa [32] ja [60]. Vastavirta-algoritmi perustuu gradientin laskemiseen ketjusäännöllä ja etenee nimensä mukaisesti tulostekerroksesta kohti syötekerrosta eli vastavirtaan koulutusdatan liikkumiseen nähden. Kytkentäpainojen W^l muuttaminen tapahtuu käymällä läpi kerrokseen l saapuvat syötteet x_j^{l-1} ja niihin kuhunkin kytketyt painot $W^l_{\cdot,j}$. Painoja siirretään virheen gradientin vastaiseen suuntaan, ja muutoksen suuruus skaalataan vakiolla η (eng.*learning rate*) [8], [37].

Algoritmin kuvaus perustuu artikkelin [44] ja kirjan [8] esityksiin. Tarkastellaan ensin tulostekerrosta L. Olkoon mallin tekemä virhe $\mathcal{E}(W|B)$ kuten edellä ja x^L tulostekerroksen L tuloste. Tällöin derivaatta matriisin W^L j:nnen sarakkeen $W^L_{\cdot,j}$ suhteen, eli painojen jotka vievät kerroksen L perseptroniin j, voidaan laskea ketjusäännön avulla:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial W^L_{\cdot,j}} &= \frac{\partial \mathcal{E}}{\partial x^L_j}\frac{\partial x^L_j}{\partial y^L_j}\frac{\partial y^L_j}{\partial W^L_{\cdot,j}} \\ &= \frac{\partial \mathcal{E}}{\partial x^L_j} \cdot \varphi_a'\left(x^L_j\right) \cdot \left(\frac{\partial (W^L_{\cdot,j})^T x^{L-1}}{\partial W^L_{\cdot,j}}\right) \\ &= \frac{\partial \mathcal{E}}{\partial x^L_j} \cdot \varphi_a'\left(x^L_j\right) \cdot \left(x^{L-1}\right).\end{aligned}$$

Saadussa yhtälössä kaikki termit on suoraviivaista laskea, kun virhefunktio $\mathcal{E}$ mallille tunnetaan. Merkitään edeltävän kerroksen tulosteen x^{L-1} edessä olevaa termiä

$$\Delta^L_j := \frac{\partial \mathcal{E}}{\partial y^L_j} = \frac{\partial \mathcal{E}}{\partial x^L_j}\frac{\partial x^L_j}{\partial y^L_j} = \frac{\partial \mathcal{E}}{\partial x^L_j}\varphi_a'(x^L_j).$$

Nyt painot W^L voidaan kaikille j päivittää kaavalla

$$W^L_{\cdot j} \;\leftarrow\; W^L_{\cdot,j} - \eta\frac{\partial \mathcal{E}}{\partial W^L_{\cdot,j}} = W^L_{\cdot,j} - \eta\Delta^L_j x^{L-1}.$$

Kun painot W^L on saatu päivitettyä, voidaan siirtyä tuloskerrosta edeltävään kerrokseen. Toisin kuin tulostekerroksessa, nyt jokainen kerroksen $L-1$ perseptroni on kiinni koko tulosteessa, eikä vain yhdessä tulosteen alkiossa. Siksi vaikutukset tulee summata yhteen yli kerroksen L solujen. Kun solujen määrää kerroksessa L merkitään $|K^L|$, gradientiksi kerroksen $L-1$ perseptroniin j saapuvien painojen suhteen saadaan

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial W^{L-1}_{\cdot,j}} &= \sum_{k=0}^{|K^L|}\left(\frac{\partial \mathcal{E}}{\partial y^L_k}\frac{\partial y^L_k}{\partial x^{L-1}_j}\right)\frac{\partial x^{L-1}_j}{\partial y^{L-1}_j}\frac{\partial y^{L-1}_j}{\partial W^{L-1}_{\cdot,j}} \\ &= \underbrace{\sum_{k=0}^{|K^L|}\left(\Delta^L_k \cdot W_{j,k}\right)\varphi_a'(x^{L-1}_j)}_{=:\Delta^{L-1}_j}(x^{L-1}).\end{aligned}$$

Lausekketta sieventäessä huomataan, että tulostekerroksen laskun tulosta Δ^L_k päästiin hyödyntämään seuraavassa laskussa, mikä onkin vastavirta-algoritmin etu monikerroksisissa verkoissa. Kun jatketaan aiempaa nimeämiskäytäntöä ja merkitään kerroksen $L-1$ tulosteen x^{L-1} edessä olevaa termiä symbolilla Δ^{L-1}_j, voidaan kytkentäpainot päivittää samoin kuin edellä. Yleiselle kerrokselle l saadaan

$$W^l_{\cdot,j} \;\leftarrow\; W^l_{\cdot,j} - \eta\Delta^l_j x^l.$$

Vastavirta-algoritmin käytössä optimointiin on muutamia haasteita, esimerkiksi aktivointifunktion derivaattaa tarvitaan jokaisen kerroksen laskutoimituksessa. Aktivointifunktioiden tulee siis olla derivoituvia mittateorian termein melkein kaikkialla. Lisäksi minimi, johon painot konvergoivat, riippuu satunnaisista arvoista, kuten kytkentäpainojen alkuarvoista [23]. On siis mahdollista, että minimi, johon painot konvergoivat, on vain lokaali ja eri alkuarvoilla saavutettaisiin parempi, ehkä jopa globaali minimi. Oppimisnopeutta η säätämällä voidaan kuitenkin vaikuttaa konvergenssiin [37]. Pienellä oppimisnopeuden arvolla konvergointi on tasaista, mutta painot voivat jäädä lokaaliin minimiin, joka on globaalia minimiä suurempi. Suuri oppimisnopeus mahdollistaa suuremmat muutokset painoissa ja siten siirtymisen minimistä toiseen, mutta konvergenssi on tällöin hidasta. Oppimisnopeutta voidaan myös säätää koulutusvaiheen aikana, jotta viimeisten optimointikierrosten aikainen konvergointi olisi tasaisempaa [19]. Esimerkiksiksi artikkelissa [59] oppimisnopeutta pienennetään koulutuksen aikana.

2.3 Koneoppiminen luonnollisen kielen käsittelyssä

Luonnollisen kielen käsittely (eng.*Natural Language Processing, NLP*) on koneoppisen sovellusalue, jossa koneoppimismalleja käytetään mallintamaan ihmisten välisissä sosiaalisissa kanssakäymisissä syntyneitä kieliä [23]. Toisin kuin formaaleissa kielissä, kuten ohjelmointikielissä, joissa ilmaisujen tarkkuus on ratkaiseva ominaisuus, luonnolliselle kielelle tyypillisiä ominaisuuksia ovat monitulkintaisuus ja ilmaisujen epätarkkuus. Tästä syystä koneoppiminen on usein nykyaikaisen luonnollisen kielen käsittelyn pohjana. Luonnollisen kielen käsittelyyn liittyviä koneoppimisongelmia ovat automaattiset käännökset, kuvien tekstittäminen ja tekstin lajitteluongelmat [23]. Tässä luvussa käydään läpi luonnollisen kielen käsittelyn peruspilareja sekä määritellään terminologiaa tutkielman tarkoituksiin.

Luonnollisesta kielestä saadun datan yhtenä erikoispiirteenä on datan sisäiset viittaussuhteet, sanojen väliset yhteydet, jotka estävät virkkeen yksittäisten sanojen käsittelyn erillisinä piirteinä. Useimmissa NLP:n käsittelemissä ongelmissa ei siis ole mahdollista syöttää tekstiä sana kerrallaan tavallisen neuroverkon sisään ja olettaa mallin suoriutuvan halutulla tavalla. Yhtenä ratkaisuna viittaussuhdeongelmaan on käytetty takaisinkytkettyjä neuroverkkoja (eng.*recurrent neural networks*), joista ensimmäiset esitteltiin artikkelissa [28] ja jotka popularisoitiin artikkelissa [44]. Edellä esiteltyihin perseptroniverkkoihin verrattuna takaisinkytketyssä verkossa lasketaan tulosteen lisäksi solun "tila", joka vaikuttaa solun seuraavaksi tekemään laskutoimitukseen. Tila toimii siis solun muistina.

Luvussa 2.1 käsiteltiin piirreirrotusta ja miten sen avulla voidaan esimerkiksi saattaa kuva numeeriseen muotoon, jotta se voidaan antaa syötteenä koneoppimismallille. Herääkin kysymys, miten tämä on mahdollista tehdä luonnollisen kielen lauseille. Oletetaan, että on olemassa kuvaus, joka kuvaa yhden lauseen d-mittaiseksi vektoriksi. Millaisia ehtoja kuvauksen tulisi noudattaa? Kuvien kohdalla annettiin esimerkiksi värien jakautumien laskemista, minkä takana on oletus, että samaa asiaa

kuvaavissa kuvissa on keskenään samanlaisia värejä. Sama pätee sanoille: sopiva kuvaus on sellainen, joka antaa yhdensuuntaisia vektoreita lauseille, joiden merkitys on samanlainen. Tapoja tehdä tällainen vektorisointi (eng.*vector embedding*) on esimerkiksi GloVe [42] ja Word2vec [40]. Vektorisointi voidaan tehdä kokonaisille dokumenteille, yksittäisille lauseille tai jopa yksittäisille sanoille. Lisäksi monikielisissä toteutuksissa vektorien yhdensuuntaisuus tulisi säilyä yli kielien.

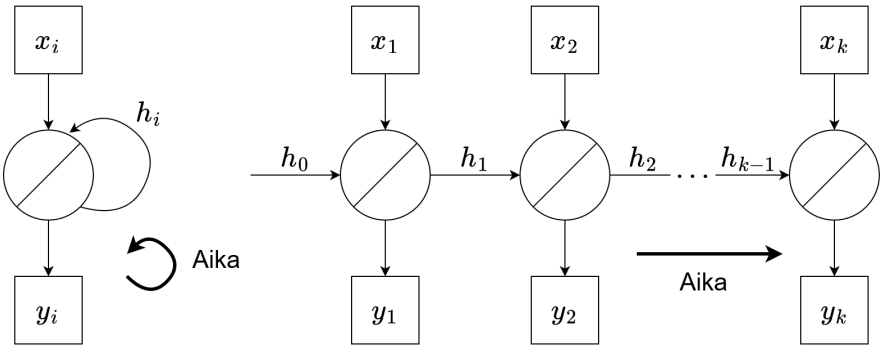
Määritellään takaisinkytketyt neuroverkot luonnollisen kielen käsittelyn kontekstissa artikkelin [58] esityksen pohjalta. Olkoon vektori h_0 solun alkutila, $x = [x_1, \ldots, x_k]^T$ soluun tuleva syöte, esimerkiksi vektorisoitu lause, φ aktivointifunktio ja W, H, Y optimoitavia kytkentäpainomatriiseja. Matriisia W käytetään edelleen syötteen painottamiseen ja matriiseja H ja Y käytetään uuden tilan ja tulosteen laskemiseen. Solun uudeksi tilaksi h_1 ja tulosteeksi $y = [y_1, \ldots, y_k]^T$ saadaan muuttujan i käydessä läpi vektorin x indeksit

$$\begin{aligned} h_i &= \varphi(W^T x_i + H^T h_{i-1}) \\ y_i &= Y^T h_i. \end{aligned}$$

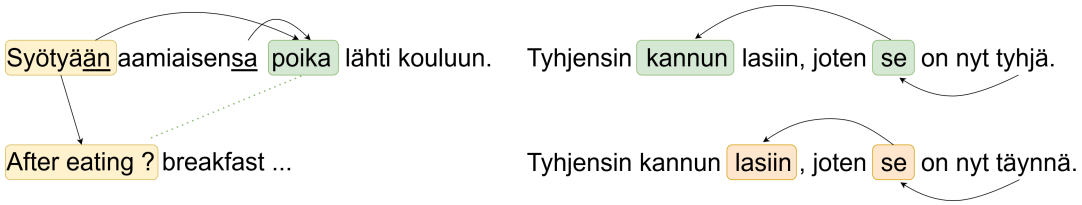
Kaksi visualisointia takaisinkytketystä verkosta on esitelty kuvassa 5. Syötettäessä koko vektori x mallin läpi, syntyy joukko tilavektoreita $\{h_i\}_{i=1}^k$, missä k on syötteen pituus. Joukkoa kutsutaan syötteen x osittaiskoodauksiksi. Jokainen yksittäinen osittaiskoodaus h_i pitää sisällään jonkin kuvauksen siitä, miten edelliset syötteet vaikuttivat tulosteen y_i generointiin [5]. Viimeinen osittaiskoodaus on siis koko vektorin x koodaus (eng.*encoding*). Intuitiivisesti voi ajatella, että vektori h_i ilmaisee siihen asti saaneensa syötteen $[x_1, \ldots, x_i]^T$ sellaisessa yksinkertaistetussa muodossa, joka antaa mallille mahdollisimman paljon informaatiota siihen tehtävään, johon mallia koulutetaan. Tällainen koodaus ei ole useinkaan riittävä, sillä kielellisessä datassa viittaussuhteet voivat olla datan syöttösuuntaan tai sitä vastaan, mutta solun tila päivittyy vain edeltävien syötteiden perusteella. Lisäksi syötteen ollessa pitkä, takaisinkytketty verkko kärsii lyhytmuistisuudesta [5]. Esimerkiksi kuvassa 6 sanan "se" suhde päälauseeseen voidaan päätellä vasta virkkeen viimeisen sanan kohdalla, joilloin solun tila todennäköisesti sisältää enää rajallisen määrän informaatiota sanoista "lasi" ja "kannu". Annetusta kaavasta on myös mahdollista havaita, että yksinkertainen takaisinkytketty neuroverkko pystyy palauttamaan vain syötteen mittaisia tulosteita eikä siis ole ihanteellinen käännösongelmiin, sillä luonnolliset kielet ilmaisevat samoja asioita eri sanamäärillä [58].

Ajatus koodauksesta esitysmuotona, joka kiteyttää syötteen haluttuihin ominaisuuksiin, realisoituu koodaus-dekoodaus -arkkitehtuurissa. Se on esitelty artikkelissa [14]. Koodaus-dekoodausmalli koostuu kahdesta eri takaisinkytketystä neuroverkosta, joista ensimmäinen koodaa syötteen vektoriksi c, joka välitetään seuraavalle verkolle, joka dekoodaa vektorin c tulosteeksi.

Määritellään koodaus artikkelin [14] mukaisesti. Koodaus tapahtuu kuten edellä. Olkoon x k-mittainen koodattava lause, johon lisätään viimeiseksi arvoksi jokin erikoissymboli x_∞, joka kertoo syötteen päättyneen. Tällöin vektoriksi c, eli lauseen



Kuva 5: Kaksi erilaista esitystapaa takaisinkytketylle neuroverkolle. Vasemmalla: Takaisinkytketty neuroverkko esitettynä suljetussa muodossa. Syötteen saapuessa lasketaan tulosteen lisäksi uusi tila, jota käytetään seuraavassa laskussa. Oikealla: Sama rakenne esitelty avatussa muodossa. Vaikka solut on esitetty erillään, kyseessä on kuitenkin sama solu, joka tekee samat laskutoimitukset. Avattu esitystapa mahdollistaa tilamuutosten ja muiden laajennusten selvemmän esittämisen.



Kuva 6: Kaksi viittaussuhdetta, jotka ovat vaikeita takaisinkytketylle mallille. Ensimmäisessä kuvassa englanninkielinen virke tarvitsee tietoa subjektista aikaisemmin kuin suomenkielinen virke. Toisessa sana, johon "se" viittaa, voidaan päätellä vasta lauseen lopussa, jolloin sana "kannu" vaikuttaa solun tilaan enää rajallisesti.

x koodaukseksi, saadaan

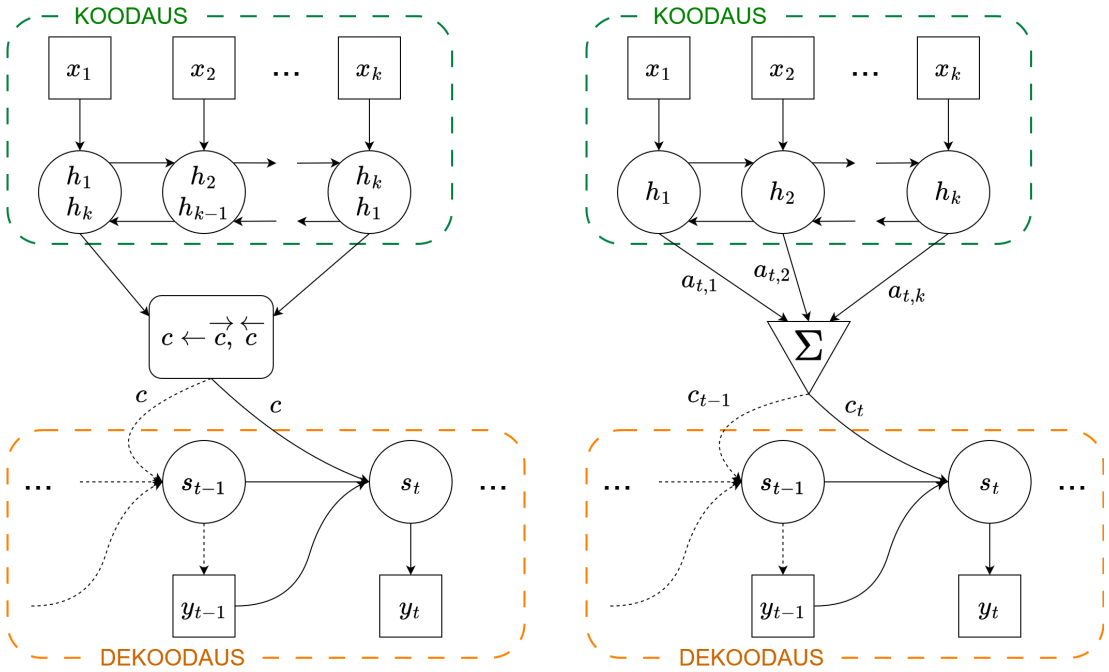
$$h_i = f\left(x_i, h_{i-1}\right), \quad \forall i \in [1, \ldots, k, \infty],$$
$$c = h_\infty,$$

missä f on jokin verkolle mahdollinen funktio, esimerkiksi jo esitelty painotetun summan kuvaus aktivointifunktiolla. Koodaus voidaan tehdä myös toisinpäin antamalla syötettä x lukusuuntaa vastaan, jolloin saadaan syötteelle saadaan kaksi koodausta, $\overrightarrow{c}$ ja $\overleftarrow{c}$, ja varmistetaan, että tekstinsisäiset viittaukset molempiin suuntiin huomioidaan [5]. Tällöin c on jokin kuvaus kummastakin viimeisestä tilasta $\overrightarrow{c}$ ja $\overleftarrow{c}$.

Artikkelin [14] mukaan dekoodaus tapahtuu toisella verkolla hyödyntäen dekoodausverkon tilaa s_t, vektoria c ja edellistä dekoodattua sanaa y_{t-1}. Olkoon g dekoodausverkon funktio. Tällöin tulosteeksi saadaan

$$y_t = g(c, s_t, y_{t-1}).$$

Tulostelauseen y pituus ei riipu enää syötteen pituudesta, vaan koko vektori c vaikuttaa jokaiseen tulosteen komponentin generointiin. Tulostealkioiden generointi lo-



Kuva 7: Vasemmalla: Artikkelin [14] mukainen koodaus-dekoodausmalli. Oikealla: Huomioon perustuva koodaus-dekoodausmalli mukaillen artikkelia [5]. Kuvissa huomioitavaa on perinteisen mallin koodauksen c staattisuus verrattuna huomioon perustuvaan malliin, jossa vektori c lasketaan erikseen jokaiselle tulosteen generoinnille. Molemmissa esityksissä erikoismerkit x_∞ ja y_∞ ovat merkitsemättä.

petetaan, kun malli tulostaa jonkin erikoissymbolin y_∞, johon dekoodattu lause lopetetaan, samoin kuin x_∞ mallin koodauspuolella [5].

Riippumatta funktioiden f ja g optimoimisesta, takaisinkytketyissä neuroverkoissa vierekkäiset sanat vaikuttavat toisiinsa aina enemmän kuin kauempana olevat, mikä rajoittaa mallin suorituskykykä merkittävästi [23]. Tämän takia yksi tärkeimmistä NLP:n alalla 2010–luvulla esitellyistä konsepteista on huomion käsite.

Huomion (eng.*attention*) käyttöönotto koodaus-dekoodausmalleissa mahdollistaa sen, että jokaiselle dekooderin tilalle saadaan valittua ne koodaustilat, joiden halutaan eniten vaikuttavan dekoodaukseen. Arkkitehtuuri on esitelty artikkelissa [5]. Huomion käyttö dekoodauksessa tapahtuu käytännössä niin, että koodausvektori c lasketaan erikseen jokaiselle dekoodaustilalle s_t painottaen koodaustiloja h_k eri tavoin. Huomiollisen ja huomiottoman mallin ero on esitetty visuaalisesti kuvassa 7.

Määritellään huomioon liittyvät laskukaavat artikkelin [5] pohjalta. Koodaus ja dekoodaus toimivat kuten yllä. Olkoon kahteensuuntaan kytketyn koodausverkon tilat $\overrightarrow{h_1}, \ldots, \overrightarrow{h_k}$ ja $\overleftarrow{h_1}, \ldots, \overleftarrow{h_k}$, ja olkoon kunkin solun kokonaistila jokin yhdistelmä kummastakin tilasta. Artikkelissa [5] yhdistämiseen on valittu konkatenointi $h_i = [\ \overrightarrow{h_i}^T, \overleftarrow{h_i}^T\ ]^T$. Tällöin jokaiselle dekoodaustilalle s_t voidaan laskea tilan huo-

mioiva koodaus c_t kaavalla

$$c_t = \sum_{i=0}^{k} \alpha_{t,i} h_i, \tag{A1}$$

missä α on huomiomatriisi ja sen alkiot huomiopainoja (eng.*attention weights*)[5]. Huomiomatriisi kertoo, mitä koodauksia on relevanteinta tarkastella kussakin dekoodaustilassa. Esimerkiksi kuvassa 6 tulosteelle "after eating his" huomiopainot olisivat suurimmat koodauksille sanoista "syötyään" ja "poika" (vrt. kuva 8). Huomiomatriisi α lasketaan kaavalla

$$\alpha_{i,j} = \frac{\exp A(s_{i-1}, h_j)}{\sum_{i'=1}^{k} \exp\ A(s_{i-1}, h_{i'})} = \sigma(A(s_{i-1}, h_j)), \tag{A2}$$

missä A on linjausfunktio (eng.*alignment function*). Artikkelissa [5] linjausfunktioksi annetaan

$$A(s_{i-1}, h_j) = v^T \tanh W s_{i-1} + U h_j,$$

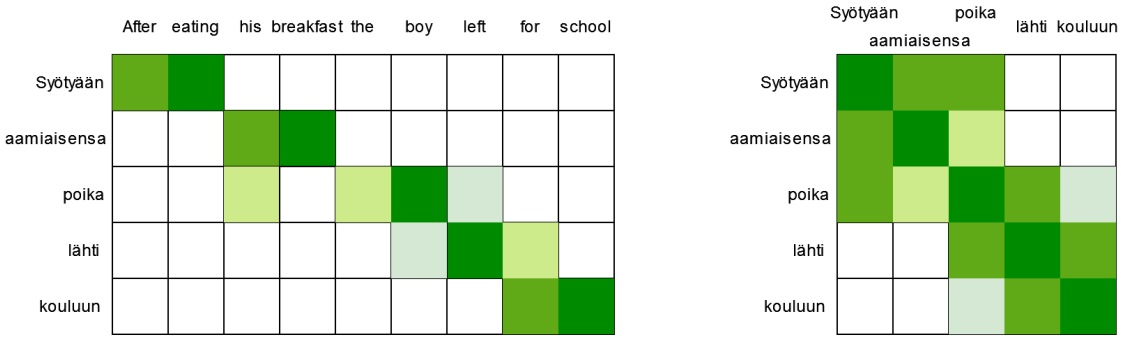
missä v, W ja U ovat optimoitavia painomatriiseja. Muita erilaisiin malleihin sopivia linjausfunktioita on esitetty muun muassa

$$\begin{aligned} A(s_{i-1}, h_j) &= s_{i-1}^T h_j, && \text{(Pistetulolinjaus, [35])} \\ A(s_{i-1}, h_j) &= \sigma(W s_{i-1}), && \text{(Paikallinen linjaus, ([35])} \end{aligned} \tag{PL}$$

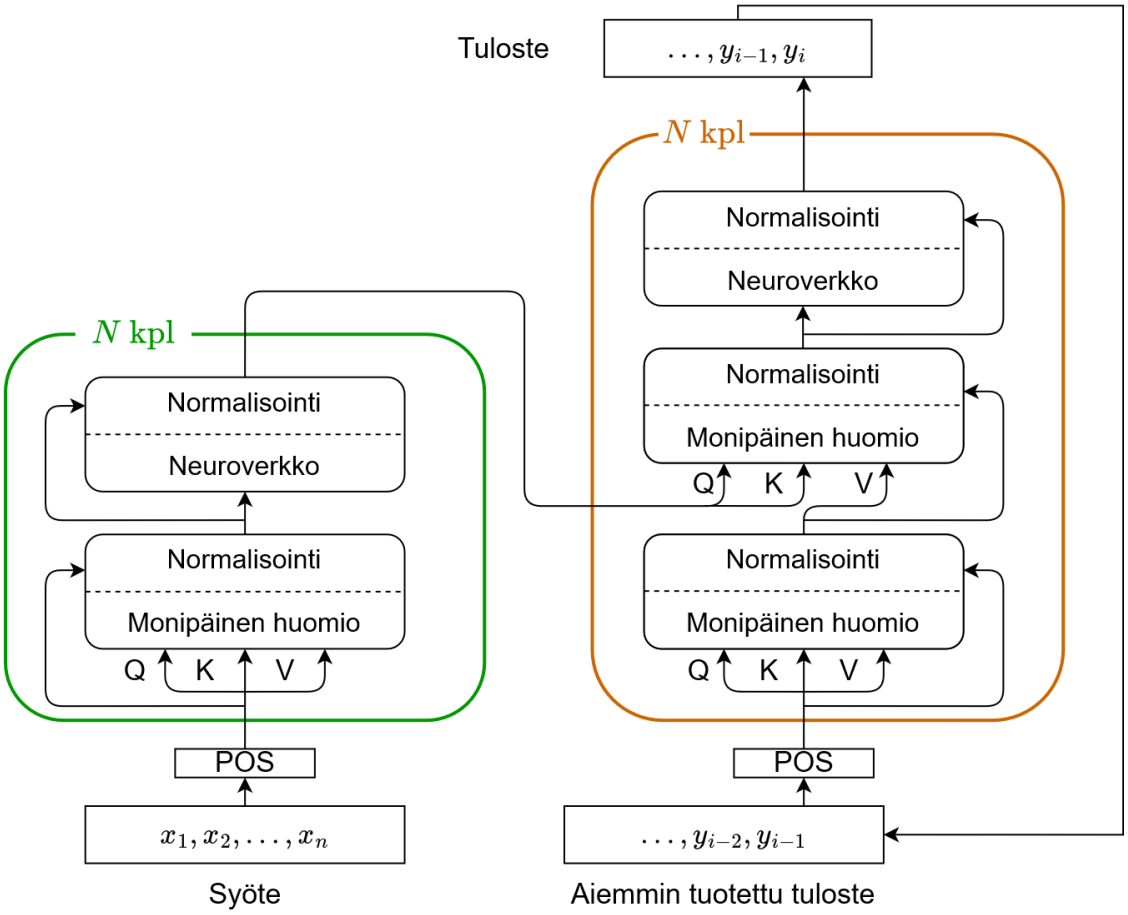
missä W on optimoitava painomatriisi, σ on aiemmin esitelty *softmax*-funktio. Linjausfunktio valitaan sovelluksen mukaan ja sen sisältämien matriisien optimointi on osa mallin koulutusta.

Huomiomatriisiin α perustuvalla koodauksella saatiin ratkaistua osa kuvaan 6 liittyvistä ongelmista. Huomioon perustuva koodaus mahdollistaa sanojen painottamisen eri järjestyksessä, sanojen yhteyksien esittämisen kumpaankin lukusuuntaan ja pääsyn kaikkiin koodauksen tiloihin samanarvoisina. Näistä jälkimmäinen ominaisuus parantaa erityisesti kuvan 6 vasemmanpuoleisen osan kaltaisia tilanteita. Ei kuitenkaan vielä yksinään riitä, että mallilla on käytössään pääsy koodauksiin sanoista "kannun" ja "lasiin", jotta tiedetään, kumpaan esimerkin sana "se" viittaa. Tällaiset kontekstiin liittyvät viittaukset mahdollistuvat hyödyntämällä sisäistä huomiota (eng.*self-attention*). Sisäinen huomio kuvaa lauseen sisäisiä viittaussuhteita ja kertoo kunkin lauseen sanan kohdalla, mihin muihin sanoihin kyseisellä sanalla on kielellisiä riippuvuuksia. Esimerkiksi kuvan 6 vasemmalla puolella sisäinen huomio tulisi olla eri sanalle "se" kummassakin lauseessa. Sisäinen huomio ei laskennallisesti poikkea takaisin kytketyn verkon "ulkoisesta" huomiosta, vaan se muodostetaan samaan tapaan kuin edellä kuvatussa koodaus-dekoodaus -arkkitehtuurissa, mutta lähtökohtana ja maalina käytetään kumpanakin saman lauseen koodauksia. Vertaus koodaus-dekoodaus -huomiomatriisista ja sisäisen huomion matriisista samalle lauseelle on esitetty kuvassa 8.

Koska huomio on voimakas työkalu, on 2010-luvulla luotu takaisinkytketyistä neuroverkoista poikkeavia arkkitehtuureja, kuten Transformer-mallit [59]. Transformer-mallit ovat johtaneet suureen kasvuun alalla ja niitä pidetään yhtenä merkityksellisimmistä muutoksista koneoppimisessa luonnollisen kielen käsittelyn saralla [61].



Kuva 8: Vasemmalla: Artikkelin [5] mukainen esimerkki huomiomatriisista α koodaus-dekoodaus -arkkitehtuurissa. Tummat värit ilmaiset suuria arvoja. Oikealla: Vastaava esitys sisäisestä huomiosta, jonka avulla voidaan huomioda lauseen sisäiset viittaussuhteet. Esimerkiksi "lähti"-"kouluun" -pari saa suuremman pisteytyksen kuin "lähti"-"aamiaisensa" -pari.



Kuva 9: Esitys Transformer-mallista artikkelin [59] mukaisesti. Koodauslohko on esitetty vihreällä ja dekoodauslohko oranssilla. Kumpikin lohko suoritetaan N kertaa. Symbolilla "POS" tarkoitetaan sijaintikoodausta.

Rakenteeltaan ne perustuvat päällekäisiin huomiota hyödyntäviin koodaus- ja dekoodauslohkoihin, joihin on yhdistetty viimeiseksi vaiheeksi täysin kytketty neuroverkko. Edellä käsiteltyihin takaisinkytkettyihin neuroneihin perustuvasta arkkitehtuurista Transformer-mallit eroavat merkittävästi kahdella tavalla. Ensiksi Tranformer-malli ei pidä yllä tilaa samoin kuin takaisinkytketty verkko, joten tieto syötteen järjestyksestä ilmoitetaan erityisellä sijaintikoodauksella. Toiseksi yhden huomiomatriisin sijaan huomiota lasketaan koodauksessa sisäisesti ja dekoodauksessa sekä sisäisesti että ulkoisesti.

Kuvassa 9 on esitetty Transformer-mallin rakenne. Vihreällä on esitetty koodauslohko, jossa ensimmäiseksi lasketaan sisäinen huomio ja sen jälkeen saatu tulos syötetään täysin kytkettyyn lyhyeen neuroniverkkoon. Kummankin vaiheen jälkeen kerros normalisoidaan, mikä on selitetty artikkelissa [31]. Syöte tuodaan myös kummankin vaiheen yli ja se summataan tulokseen niin kutsuttuna jäännösyhteytenä (eng.*residual connection*), jonka tarkoitus on parantaa koulutuksen suppenemista [25]. Koodauslohkon tulokset välitetään dekooderille, joka on esitetty kuvasssa 9 oranssilla. Dekoodauslohkossa lasketaan aiemmin saaduille tuloksille sisäinen huomio samoin kuin koodauslohkossa, jonka jälkeen koodauslohkosta saatujen tulosten kanssa lasketaan ulkoinen huomio. Tulokset viedään koodauslohkon tavoin eteenpäin neuroverkolle, jonka tulos annetaan ulos tulosteena.

Transformer-arkkitehtuuri käyttää huomiona monipäistä huomiota (eng.*multihead attention*), joka perustuu kaavan (PL) määrittelemään pistetulolinjaukseen [59]. Monipäisessä huomiossa huomio lasketaan erikseen M kertaa kuvaamalla huomion laskemiseen tarvittavat arvot M:lla eri lineaarikuvauksella ja laskemalla näistä jokaisesta erillinen huomio pistetulolinjauksella. Kuvaamalla arvot eri matriiseilla saadaan jokaisessa laskutoimituksessa, eli huomion "päässä", tietyt yhteydet alkioiden välillä korostumaan eri tavoin. Saadut huomiot skaalataan tekijällä $1/\sqrt{d_k}$, missä d_k on kunkin lineaarikuvauksen dimensio. Lopuksi tulokset konkatenoidaan lopulliseksi monipäiseksi huomioksi. Koska Transformer-arkkitetuurissa ei ole takaisin kyketyn neuroverkon kaltaista tilaa, käytetään kaavojen (A1) ja (A2) tilojen h_i, h_j ja s_{i-1} tilalla vektoreita V (eng.*value*), K (eng.*key*) ja Q (eng.*query*). Formaalisti monipäinen huomio siis lasketaan vektoreille Q, K ja V

$$\text{Multihead}(Q, K, V) = [head_1, \ldots, head_M] \cdot W, \qquad \text{(konkatenointi)}$$

$$head_k = \sigma(\frac{Q_k K_k^T}{\sqrt{d_k}})V_k, \qquad \text{(A1 + A2+ PL)}$$

$$Q_k,\ K_k,\ V_k = QW_k^Q,\ KW_k^K,\ VW_k^v, \qquad \text{(lineaarikuvaus)}$$

missä W_k^Q, W_k^K ja W_k^v ovat matriiseja, jotka kuvaavat syötteitä aiemmin kuvaillulla tavalla, ja W kuvaa konkatenoidun tuloksen dimensioon, jota seuraava operaatio vaatii. Kun vektorit Q, K ja V ovat samasta lohkosta, saadaan aikaan sisäinen huomio. Kun taas Q ja K ovat koodauksen tuottamia ja V on dekooderin puolelta, saadaan samanlainen ulkoinen huomio kuin koodaus-dekoodaus–verkossa. Kuvassa 9 tämä näkyy dekoodauslohkon toisessa operaatiossa.

Huomion laskemiseksi on toivottavaa, että kaikki syötteen osat ovat saatavilla samanarvoisesti. Vaikka edellä on todettu, että luonnollisissa kielissä läheinen sijainti ei välttämättä implikoi relaatiota sanojen välillä, on sanojen järjestyksellä silti merkitystä lauseen ymmärtämisessä ja generoimisessa. Transformer-arkkitehtuurissa syötteiden järjestyksen ylläpitämiseksi tehdään syötteille ennen koodaus- ja dekoodauslohkoa sijaintikoodaus, joka on kuvassa 9 esitetty symbolilla "POS". Sijaintikoodaus tehdään summaamalla syötteeseen ortogonaaliset sini- ja kosinifunktiot, jolloin sanan sijainti suhteessa muihin sanoihin voidaan päätellä.

Transformer-mallit ovat mahdollistaneet alan räjähdysmäisen kasvun, sillä ne soveltuvat erityisesti esikoulutukseen [61]. Esikoulutuksessa malli koulutetaan ensiksi isolla aineistolla, jonka jälkeen sitä hienosäädetään haluttuun tehtävään. Artikkeli [59] on innoittanut uutta tutkimusta moneen eri suuntaan, kuten kielen koneelliseen ymmärtämiseen, mistä esimerkkinä BERT [16], joka käyttää Transformer-arkkitetuurin koodauslohkoa osana rakennettaan, sekä kielen konegenerointiin, mistä esimerkkinä GPT-3 [11], joka puolestaan käyttää Transformer-mallin dekooderia.

3 Selitysalgorimit

Selitysalgoritmit ovat algoritmeja, joiden avulla koneoppimismallien päätöksentekoa pyritään selittämään. Luvussa 1 käsiteltiin, miksi selitysalgoritmeja tarvitaan, millaisia selitysalgoritmeja on olemassa ja mihin niitä käytetään, kun taas tässä luvussa perehdytään post-hoc-tyylisten selitysalgoritmien matemaattisiin lähtökohtiin ja ominaisuuksiin.

Post-hoc selitysalgoritmien kohdalla puhutaan datan piirteiden vaikutuksesta (eng.*feature attribution, feature relevance*) [4]. Vaikutus on datapisteen piirteiden ominaisuus, joka kuvaa kunkin piirteen myötävaikutusta mallin tulosteen tuottamisessa. Koska koneoppimimalleja voidaan käsitellä moniulotteisina funktioina saamiensa datapisteiden piirteistä, on mahdollista tutkia, miten jokainen erillinen piirre vaikuttaa tulosteeseen. Intuitiivinen tapa lähteä vastaamaan tähän kysymykseen on muuttaa tutkittavan datapisteen yksittäisten piirteiden arvoa ja tarkastella, kuinka voimakkaasti ja mihin suuntaan piirteen säätäminen vaikuttaa mallin tulosteeseen. Piirre on vaikuttava, jos sen muuttaminen vaikuttaa tulosteeseen voimakkaasti [48], [50]. Tässä kontekstissa selitys (eng.*explanation*) voidaan määritellä olevan jokin piirteiden vaikutuksista koottu esitys, jonka pohjalta voidaan tehdä päätelmiä mallin toiminnasta (vrt. Luku 1 sekä esimerkiksi [22]).

Vaikuttavat piirteet ovat eri asia kuin globaalisti informatiiviset piirteet, joskin nämä kaksi konseptia ovat silloin tällöin osittain päällekäisiä. Informatiivisia piirteitä ovat piirteet, joita käytetään jokaisen ennusteen luomisessa muita piirteitä enemmän. Tätä tietoa voidaan käyttää mallin globaalissa selittämisessä tai resurssien kannalta tehokkaampien mallien muodostamiseen poistamalla vähemmän tärkeitä piirteitä, mikä on osa piirreirrotusta. Vaikutuksesta puhuttaessa tarkoitetaan yksittäisen ennusteen muodostumista: piirreavaruudessa on kohtia, joissa tuloste $F(x)$ on enemmän riippuvainen joistain piirteistä verrattuna toisiin. Malli voi siis käyttää yhden luokan ennustamiseen tiettyjä piirteitä ja toisen luokan ennustamiseen toisia, ja mallin selittäminen perustuu näiden erojen analysointiin.

Vaikutus voidaan formaalisti määritellä funktioksi, joka antaa jokaiselle datapisteen piirteelle skalaarisen pisteytyksen sen mukaan, kuinka paljon kukin piirre tulosteeseen vaikuttaa [4]. Olkoon vaikutus $\mathcal{A}_F : \{x_i\}_{i=1}^d \to \mathbb{R}$, missä x_i on d-ulotteisen datapisteen $\{x, r\}$ i:s piirre ja F on koneoppimismallin funktio. Eri selitysmenetelmät esittävät erilaisia ehtoja funktiolle $\mathcal{A}$, mutta yleinen vaatimus on, että $\mathcal{A}$ on normalisoitu piirteiden suhteen [57], toisin sanoen

$$\sum_{i=1\ldots d} \mathcal{A}_F(x_i) = F(x). \tag{N1}$$

Lisäksi mikäli funktio F ei riipu piirteestä x_k, on luonnollista vaatia [57], että

$$\mathcal{A}_F(x_k) = 0. \tag{N2}$$

Esitellään seuraavaksi rakenteiltaan kolme hyvin erilaista selitysalgoritmia, kerroksittainen relevanssi, integroidut gradientit ja peittoanalyysi, sekä todistetaan näille menetelmille merkittäviä ominaisuuksia. Lopuksi vertaillaan menetelmien toimintaa kuvantunnistussovelluksessa. Koska moniluokkalajittelussa mallin antama tuloste on vektori ja vaikutus lasketaan aina tiettyä luokkaa kohden, tässä luvussa funktiolla $F(x)$ tarkoitetaan aina yhteen luokkaan liittyvää tulostetta. Esimerkiksi jos $F(x)$ on kolmialkioinen vektori, suoritetaan vaikutuslasku $F(x)_1$, $F(x)_2$ ja $F(x)_3$ suhteen riippuen siitä, mitä luokkaa halutaan tarkastella.

3.1 Kerroksittainen relevanssi

Artikkelissa [29] esitelty selitysmenetelmä, kerroksittainen relevanssi (eng.*layer-wise relevance*), on nimensä mukaisesti kerroksittain etenevä algoritmi. Suurimpana oletuksena sen käyttämiseen on, että mallin funktio F voidaan esittää yhdistettynä funktiona

$$F(x) = F_L \circ \cdots \circ F_1(x),$$

missä jokainen funktio F_i kuvaa mallin kerrosta i [49]. Kerroksittainen relevanssi onkin siis erityisesti neuroverkoille soveltuva selitysmenetelmä. Koska muitakin kerroksittaisia koneoppimismalleja on olemassa, artikkeli [29] tarjoaa tarkemmin sanottuna reunaehdot menetelmälle, joka etenee verkossa kerroksittain, sekä esittelee muutamalle mallille sopivan implementoinnin.

Kerroksittainen relevanssi –menetelmä pohjautuu selitysmenetelmien normalisointiehtoon (N1). Normalisointiehto ottaa kantaa siihen, miten mallin ensimmäisen kerroksen syötteet x ja mallin viimeisen kerroksen tulosteet $F(x)$ riippuvat toisistaan vaikutusfunktion $\mathcal{A}(x_i)$ suhteen. On kuitenkin täysin mahdollista käsitellä myös jotakin toista mallin kerrosta uutena syötekerroksena: yksi luonnollinen tapa ajatella tällaista jakoa on, että aiemmat kerrokset tekevät datalle esikäsittelyä ja itse verkko alkaa vasta kyseisestä kerroksesta, jolloin normalisointiehdon tulisi myös päteä tälle kerrokselle. Valitsemalla jokainen kerros vuorotellen syötekerrokseksi ja soveltamalla ehtoa (N1) kyseiseen kerrokseen, saadaan vahvempi ehto, johon kerroksittainen relevanssi perustuu. Tässä luvussa oletetaan, että verkossa ei ole *bias*-vakiotermejä b^l, kunnes ne otetaan selvästi tekstissä huomioon.

Lemma 1. (**K1**) Olkoon verkko määritelty kuin luvussa 2, olkoon siinä $l \in \{1, \ldots, L\}$ kerrosta sekä syötekerros $l = 0$, ja olkoon $K^l = \{x_1^l, x_2^l, \ldots, x_d^l\}$ tulosteiden joukko kerroksessa l. Tällöin vaikutuksille $\mathcal{A}$ pätee kaavan (N1) nojalla

$$F(x) = \sum_{x' \in K^{L-1}} \mathcal{A}(x') + \cdots + \sum_{x' \in K^0} \mathcal{A}(x'). \tag{K1}$$

Artikkelissa [29] kuitenkin osoitetaan esimerkillä, ettei saatu ehto vielä yksinään riitä toteuttamaan merkityksellisiä tulkintoja syötteen ja tulosteen yhteydestä. Intuition vahvemman ehdon johtamiseen tarjoaa artikkeli [49]. Vaikutuksen voi ajatella virtaavan verkossa kuin virran sähköpiirissä. Sähköpiirissä kulkeva virta noudattaa

Kirchhoffin virtalakia, jonka mukaan jokaiseen piirin pisteeseen tulee yhtä paljon virtaa kuin sieltä poistuu. Saman voi ajatella pätevän vaikutukselle. Määritellään siis artikkelia [29] mukaillen osittaisvaikutus, joka liikkuu kerroksen l solusta i kerroksen $l-1$ soluun j

$$A^l_{j\leftarrow i}.$$

Merkitään yksinkertaisuuden vuoksi myös kunkin kerroksen $l \in \{0, 1, \ldots, L\}$ syötteen vaikutusta

$$\mathcal{A}(x^l_j) =: \mathcal{A}^l_j.$$

Jotta vaikutus säilyisi kulkiessaan solun läpi, täytyy vaikutuksen yhdessä verkon pisteessä olla summa siihen saapuvista osittaisvaikutuksista. Määritellään tämä seuraavaksi osittaisvaikutusten kautta.

Määritelmä 2. (**K2**) Olkoon verkko kuten määritelmässä 1. Tällöin kaikille kerroksille, paitsi kerrokselle L, pätee

$$\mathcal{A}^l_j = \sum_{1\leq i\leq |K^{l+1}|} \mathcal{A}^{l+1}_{j\leftarrow i}. \tag{K2}$$

Tällöin esimerkiksi kuvan 10 ensimmäisen piirteen vaikutus osittaisvaikutusten avulla ilmaistuna on

$$\mathcal{A}^0_1 = \mathcal{A}^1_{1\leftarrow 1} + \mathcal{A}^1_{1\leftarrow 2}.$$

Koska (K2) käsittelee kerrokseen saapuvia osittaisvaikutuksia, määritellään vastaava yhtälö myös kerroksesta lähteville osittaisvaikutuksille.

Määritelmä 3. (**K3**) Olkoon verkko kuten määritelmässä 1. Tällöin kaikille kerroksille, paitsi kerrokselle 0, pätee

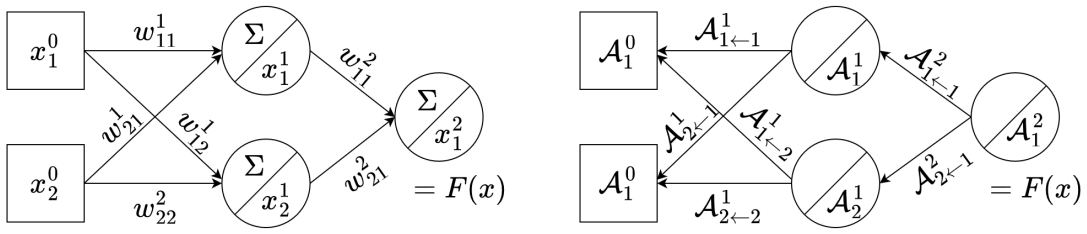
$$\mathcal{A}^l_j = \sum_{1\leq i\leq |K^{l-1}|} \mathcal{A}^l_{i\leftarrow j}. \tag{K3}$$

Kuvassa 10 tämän yhtälön perusteella pätee esimerkiksi

$$\mathcal{A}^1_2 = \mathcal{A}^1_{1\leftarrow 2} + \mathcal{A}^1_{2\leftarrow 2}.$$

Yhtälöt (K1), (K2) ja (K3) ovat yhdessä tarpeeksi vahvat tuottamaan merkityksellisen selitysmenetelmän raamit. Todistetaan vielä ennen esimerkkien tarkastelua artikkelia [29] mukaillen, että (K1) seuraa ehdoista (K2) ja (K3). Ehdon (K1) poistaminen mahdollistaa verkkojen, joissa *bias*-vakiot b^l ovat mukana, selittämisen, sillä ehto (K1) ei päde termien b^l ollessa mukana verkossa [49]. Osittaisvaikutusten määritteleminen termeille b^l on suoraviivaista, joskin vain ehdon (K2) mukaiset vaikutukset voidaan määritellä.

Todistus. Olkoon x^l_i i:nnen solun tuloste kerroksessa l ja olkoon sen vaikutus $\mathcal{A}(x^l_i) =: \mathcal{A}^l_i$. Olkoon K^l kerroksen l tulosteiden joukko ja olkoot osittaisvaikutukset



Kuva 10: Visualisointi verkosta ennustus- ja selitysvaiheessa. Vasemmalla kaksikerroksinen verkko, jossa kytkentäpainojen $w_{(\cdot)}^{(\cdot)}$ ylempi indeksi ilmoittaa kerroksen ja alempi niiden solujen indeksit, jotka on kytketty kyseisellä painolla. Oikealla on esitetty samalla indeksoinnilla solujen vaikutukset $\mathcal{A}_{(\cdot)}^{(\cdot)}$ sekä osittaisvaikutukset $\mathcal{A}_{(\cdot)\leftarrow(\cdot)}^{(\cdot)}$.

määritelty kuten edellä. Tällöin

$$\begin{aligned}
\sum_{x' \in K^{l+1}} \mathcal{A}(x') &= \sum_{1 \leq k \leq |K^{l+1}|} \mathcal{A}_k^{l+1} \\
&\overset{(K3)}{=} \sum_{1 \leq k \leq |K^{l+1}|} \sum_{1 \leq i \leq |K^l|} \mathcal{A}_{i \leftarrow k}^{l+1} \\
&= \sum_{1 \leq i \leq |K^l|} \sum_{1 \leq k \leq |K^{l+1}|} \mathcal{A}_{i \leftarrow k}^{l+1} \\
&\overset{(K2)}{=} \sum_{1 \leq i \leq |K^l|} \mathcal{A}_i^l \;=\; \sum_{x' \in K^l} \mathcal{A}(x'),
\end{aligned}$$

eli kerrosten vaikutusten summa on yhtäsuuri ja ehto (K1) toteutuu. □

Tarkastellaan lopuksi vielä tapoja määritellä vaikutusfunktio, joka toteuttaa halutut ehdot. Yksi yksinkertaisimmista tavoista määritellä kerroksittaisen relevanssin mukainen selitysalgoritmi perseptroniverkolle on

$$\mathcal{A}^l = \sum_k \frac{a_j w_{jk}}{\sum_j a_j w_{jk}} \mathcal{A}^{l-1},$$

missä a_j on kerroksen aktivoinnin tulos ja w_{jk} ovat painot kerroksen l ja $l-1$ välillä, olettaen, että varmistetaan nimittäjän olevan erisuuri kuin nolla [49]. Esimerkiksi artikkelissa [50] nimittäjän nollakohdista aiheutuva ongelma kierretään yksinkertaisesti lisäämällä nimittäjään pieni positiivinen termi. Tällöin saatu kaava vaikutukselle ei täytä ehtoja (K2) ja (K3), mutta asettaessa termi tarpeeksi pieneksi menetelmä tuottaa edelleen merkityksellisiä tulkintoja mallin toiminnasta.

Kerroksittaisen relevanssin suurimpana etuna on, että edellisen yhtälön kaltainen määrittely vaikutuksille ei tarvitse aktivointifunktion derivoituvuutta eikä edes jatkuvuutta. Kerroksittainen relevanssi sopiikin erityisesti tapauksiin, joissa näitä oletuksia ei voida tehdä aktivointifunktiosta.

3.2 Integroidut gradientit

Integroidut gradientit -algoritmi (*IG*, eng. *Integrated Gradients*) on koneoppimismallin funktion F gradienttiin perustuva selitysmenetelmä, joka on esitelty artikkelissa [57]. Gradientin käyttö selitysmenetelmässä on luonnollista, sillä mikäli piirre x_i on merkittävä mallin toiminnalle, niin tällöin funktion F gradientti piirteen x_i suhteen on erisuuri kuin nolla. Mikäli näin ei olisi, ei piirteen x_i arvon vaihtaminen vaikuttaisi funtion F arvoon, eikä se silloin olisi merkittävä annetun vaikuttavuuden määritelmän mukaan.

Gradienttiin perustuvia menetelmiä on olemassa muitakin kuin integroidut gradientit, esimerkiksi artikkeleissa [53] ja [55] esitellyt algoritmit. Erona muihin gradienttiin perustuviin menetelmiin, integroidut gradientit -menetelmä on rakennettu aksiomaattisesti kahdesta ehdosta. Määritellään aluksi kaksi artikkelin [57] antamaa ominaisuutta, joihin integroidut gradientit -algoritmin rakentaminen perustuu.

Määritelmä 4. (**Täydellisyys**) Olkoon F koneoppimismallin oppima funktio ja $\mathcal{A}_F$ vaikutusfuktio funktiolle F. Olkoon $x, x' \in \mathcal{X} \subset \mathbb{R}^d$. Tällöin vaikutusfunktio on täydellinen, mikäli

$$\sum_{k=1}^{d} \mathcal{A}_F(x_k, x'_k) = F(x) - F(x').$$

Määritelmässä käytetty piste x' on menetelmän *vertailupiste*. Vertailupiste toimii vaikutusfunktion "nollakohtana", ja siksi sen on tärkeä olla sekä selityksiä tulkitsevan ihmisen mielestä että mallin toiminnan mukaan merkityksetön. Mallin näkökulmasta merkityksettömuus voi tarkoittaa esimerkiksi, että sen tuloste kyseiselle vertailupisteelle on nollavektori [56], [57]. Tällöin määritelmä yksinkertaistuu

$$\sum_{k=1}^{d} \mathcal{A}_F(x_k, x'_k) = F(x),$$

joka on sama kuin aiemmin esitelty normalisointiehto (N1). Kuvantunnistuksessa käytettyjä vertailupisteitä ovat muun muassa musta tai harmaa kuva ja luonnollisen kielen käsittelyn ongelmissa tyhjän lauseen vektorisointi. Vertailupisteen valinta on kuitenkin monimutkainen ongelma, kuten artikkelista [56] nähdään: esimerkiksi vertailupisteen ollessa musta kuva, korostuvat selityksissä vaaleat pikselit tummia enemmän.

Lopuksi vielä huomataan, että täydellisyydestä seuraa suoraan (N2). Mikäli funktion F arvot ei riipu piirteestä x_i, niin $F(x) - F(x') = 0$.

Määritelmä 5. (**Invarianssi toteutuksen suhteen**) Olkoot F ja G mallin oppimia funktioita. Jos kaikille pisteille x ehdosta $F(x) = G(x)$ seuraa $\mathcal{A}_F(x_i) = \mathcal{A}_G(x_i)$ kaikille pisteen x piirteille i, niin vaikutusfunktio $\mathcal{A}$ on invariantti toteutuksen suhteen. Funktioita F ja G kutsutaan tällöin funktionaalisesti yhtäsuuriksi.

Invarianssi vie integroidut gradientit menetelmää vastakkaiseen suuntaan verrattuna luvussa 3.1 esiteltyyn kerroksittaiseen relevanssiin. Kerroksittainen relevanssi pureutuu käytetyn mallin rakenteisiin, kun taas integroidut gradientit -menetelmän toisena kulmakivenä on oletus, että kaksi mallia voivat olla rakenteeltaan hyvin erilaisia, mutta mikäli ne antavat samat tulosteet datapisteille x, tulisi myös niiden antamisen selitysten olla samat. Määritellään seuraavaksi poluittain integroidut gradientit -menetelmä.

Määritelmä 6. (**Poluittain integroidut gradientit**) Olkoon funktio $\gamma(\alpha) : [0,1] \to \mathbb{R}^d$, missä $\gamma(0) = x' \in \mathbb{R}^d$ ja $\gamma(1) = x \in \mathbb{R}^d$, jatkuva sileä polku syötteen x ja vertailupisteen x' välillä. Merkitään i:nnettä komponenttia polusta merkinnällä γ_i. Tällöin piirteen i vaikutus poluittain integroidut gradientit -menetelmällä on

$$\mathrm{IG}_i^{\gamma}(x) := \int_{\alpha=0}^{1} \frac{\partial F}{\partial \gamma_i(\alpha)} \left(\gamma(\alpha)\right) \frac{\partial \gamma_i(\alpha)}{\partial \alpha} \mathrm{d}\alpha,$$

missä $\frac{\partial F}{\partial \gamma_i(\alpha)} \left(\gamma(\alpha)\right)$ on funktion F derivaatta suunnassa $\gamma_i(\alpha)$ arvolla $\gamma(\alpha)$.

Poluittain integroidut gradientit -menetelmä siis muodostaa perheen funktioita, joissa jokaisessa integroidaan yli eri polun γ. Jokainen perheen jäsen toteuttaa ehdon (N2) triviaalisti, sillä $\frac{\partial F}{\partial \gamma_i} = 0$, jos F ei riipu piirteestä i. Samoin toteutusinvarianssi seuraa suoraan siitä, että poluittain integroidut gradientit -algoritmi käyttää vain mallin viimeisen kerroksen antamaa funktiota eikä välituloksia. Todistetaan, että poluittain integroidut gradientit -menetelmä toteuttaa täydellisyyden.

Todistus. (Täydellisyys) Olkoot funktioiden IG_i^{γ}, F sekä polun $\gamma(\alpha)$ määrittelyt kuten yllä. Tällöin

$$\begin{aligned}
\sum_{k=1}^{d} \mathrm{IG}_k^{\gamma}(x) &= \sum_{k=1}^{d} \int_{\alpha=0}^{1} \frac{\partial F}{\partial \gamma_k(\alpha)} \left(\gamma(\alpha)\right) \frac{\partial \gamma_k(\alpha)}{\partial \alpha} \mathrm{d}\alpha \\
&= \int_{\alpha=0}^{1} \nabla F \left(\gamma(\alpha)\right) \cdot \gamma'(\alpha) \mathrm{d}\alpha \\
&= F(\gamma(1)) - F(\gamma(0)) = F(x) - F(x'),
\end{aligned}$$

missä toiseksi viimeisessä yhtäsuuruudessa käytetään analyysin peruslausetta. □

Määritellään vielä yksi selitysmenetelmältä toivottava ominaisuus, joka annetaan artikkelissa [57].

Määritelmä 7. (**Symmetrian säilyminen**) Kaksi muuttujaa x_i ja x_j ovat symmetriset funktion F suhteen, mikäli $F(x_i, x_j) = F(x_j, x_i)$ kaikilla x_i, x_j. Vaikutusfunktio säilyttää symmetrian funktion F suhteen, mikäli

$$\mathcal{A}_F(x_i) = \mathcal{A}_F(x_j)$$

aina, kun vertailupistelle x' ja syötteelle x pätee $x'_i = x'_j$ ja $x_i = x_j$.

Symmetrian säilyvyys on toivottu ominaisuus, sillä mikäli kaksi piirrettä toimivat mallissa keskenään symmetrisesti, tulisi niiden saada yhtäsuuret vaikutukset selitettäessä mallia [57]. Tarkastellaan esimerkin avulla, säilyttääkö määritelmän 6 mukainen menetelmä symmetrisyyden.

Esimerkki. Olkoon $x' = [0, \ldots, 0]^T$ ja $x = [1, \ldots, 1]^T$. Olkoon $\gamma(\alpha) : \alpha \in [0, 1] \to \mathbb{R}^d$ jatkuva polku pisteiden x' ja x välillä, missä $\gamma(0) = x'$ ja $\gamma(1) = x$ ja d syötteen kokonaisdimensio. Olkoon $F(x) = x_i \cdot x_j$. Selvästi F on symmetrinen piirteiden i ja j suhteen, sekä $\frac{\partial F}{\partial x_i}(x) = x_j$ ja $\frac{\partial F}{\partial x_j}(x) = x_i$. Kun kyseisille piirteille lasketaan poluittain integroitujen gradienttien mukainen vaikutus, saadaan

$$\mathrm{IG}^\gamma(x) = \begin{cases} \int_{\alpha=0}^{1} \gamma_j(\alpha)\gamma_i'(\alpha)\mathrm{d}\alpha, \text{ piirteelle } i, \\ \int_{\alpha=0}^{1} \gamma_i(\alpha)\gamma_j'(\alpha)\mathrm{d}\alpha, \text{ piirteelle } j, \\ 0, \text{ muille piirteille.} \end{cases}$$

Koska $x_i' = x_j'$ ja $x_i = x_j$ ja F on symmetrinen piirteiden x_i ja x_j suhteen, tulee vaikutusten olla samat. Yksi ratkaisu on $\gamma_i(\alpha) = \gamma_j(\alpha)$ kaikilla $\alpha \in [0, 1]$, sillä

$$\begin{aligned} \gamma_j(\alpha)\gamma_i'(\alpha) &= \gamma_i(\alpha)\gamma_j'(\alpha) \\ \int \frac{\gamma_i'(\alpha)}{\gamma_i(\alpha)}\mathrm{d}\alpha &= \int \frac{\gamma_j'(\alpha)}{\gamma_j(\alpha)}\mathrm{d}\alpha \\ \gamma_i(\alpha) &= \gamma_j(\alpha) + C, \end{aligned}$$

missä $C = 0$ seuraa funktion γ määrittelystä, esimerkiksi pisteessä $\alpha = 0$. Jatkuvan polun tapauksessa se, että jokainen komponentti on yhtä suuri kuin toiset, tarkoittaa polun olevan suora viiva pisteiden x' ja x välillä. △

Tässä esimerkissä symmetrian säilyminen toteutui vain, jos polku γ on suora viiva pisteiden x' ja x välillä. Osoittautuu, että tämä on ainoa tapa säilyttää symmetria. Todistus, joka toimii annetun esimerkin inspiraationa, on esitetty artikkelin [57] liitteessä A. Sijoittamalla määritelmään 6 poluksi symmetrian säilyttävä suora viiva $\gamma(\alpha) = x' + \alpha(x - x')$, saadaan integroidut gradientit -menetelmälle seuraava määritelmä.

Määritelmä 8. (**Integroidut gradientit**) Olkoon funktio $\gamma(\alpha) = x' + \alpha(x - x'), \alpha \in [0, 1]$. Merkitään i:tä komponenttia polusta merkinnällä γ_i. Tällöin piirteen i vaikutus integroidut gradientit -menetelmällä on

$$\begin{aligned} \mathrm{IG}_i(x) &:= \int_{\alpha=0}^{1} \frac{\partial F(\gamma(\alpha))}{\partial \gamma_i(\alpha)} \frac{\partial \left(x_i' + \alpha(x_i - x_i')\right)}{\partial \alpha}\mathrm{d}\alpha \\ &= (x_i - x_i') \cdot \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i}\mathrm{d}\alpha. \end{aligned}$$

Määritelmän 8 mukainen selitysalgoritmi voidaan implementoida tietokoneella helposti laskemalla integraalille likiarvo summana.

3.3 Peittoanalyysi

Peittoanalyysi (eng.*occlusion analysis*) on yksinkertainen selitysmenetelmä, jossa mallilta peitetään iteratiivisesti osa syötteen piirteistä. Jokaisen osapeiton jälkeen mallin toiminnan muutosta analysoidaan ja mikäli peitettäessä piirre x_i mallin toiminta muuttuu, kyseinen piirre selittää mallin toimintaa [62]. Esimerkiksi kuvantunnistussovelluksessa voidaan kuvasta peittää $n \times n$–kokoinen neliö pikseleitä ja katsoa, miten mallin tuloste muuttuu. Tekstidatassa saman asian ajaa muutaman vierekkäisen sanan poisto kerrallaan. Formaalisti artikkeli [49] määrittelee peitevaikutuksen

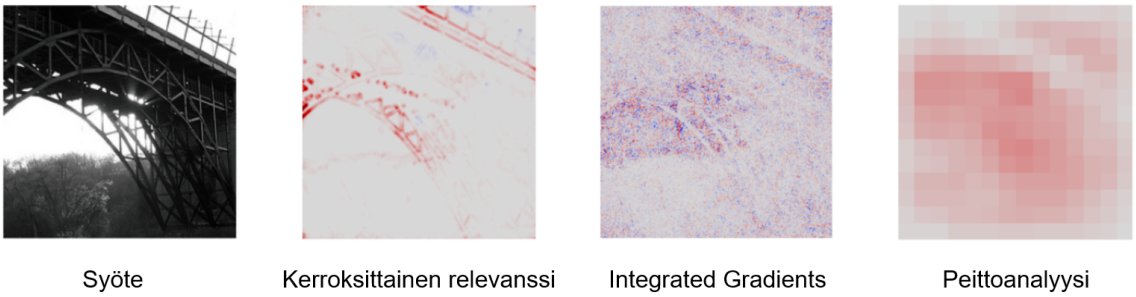
$$\mathcal{A}(x_i) = F(x) - F(x \odot m_i),$$

missä m_i on peite, jonka keskellä piirre x_i on. Merkinnällä $\odot$ tarkoitetaan pisteittäistä kertolaskua syötteen x ja peiton m_i välillä, missä m_i saa esimerkiksi arvon 0 peitettävällä alueella ja arvon 1 muualla. Peitto m_i parametrisoidaan riippuen sen muodosta, mutta esimerkiksi neliöpeite voidaan ilmoittaa sen koon ja askelpituuden funktiona: askelpituus määrittää, kuinka paljon peitto liikkuu iterointien välillä ja koko peiton sivupituuden, ja täten kuinka yksityiskohtainen selitys on. Koska peittoanalyysi ei tee mitään oletuksia tutkitun mallin rakenteesta, on sitä mahdollista soveltaa myös malleihin, joiden lähdekoodi ei ole saatavilla esimerkiksi kaupallisten syiden takia [49].

3.4 Algoritmien vertailu

Artikkelin [4] mukaan selitysalgoritmien tuottamien selitysten arvioiminen ja vertaaminen toisiinsa on hankalaa, sillä kirjallisuudessa ei ole esitetty yleisesti hyväksyttäviä metriikoita selitysten erilaisille ominaisuuksille. Tällaisten metriikoiden kehittäminen on myös vaikeaa jo aiemmin tarkasteltujen ongelmien takia, kuten sen, että yhtä objektiivisesti oikeaa selitystä, johon tuloksia verrata, ei ole olemassa. Myös yritykset verrata selityksiä kvalitatiivisesti sisältävät samoja ongelmia [21]. Tarkastellaan kuitenkin yhtä yritystä vertailla määriteltyjen algoritmien tuottamia selityksiä. Artikkelissa [49] algoritmeja tutkitaan kolmen eri kriteerin kautta: täsmällisyys, tulkittavuus ja käyttökelpoisuus. Vertailujen pohjana käytetään ImageNet-kuva-aineistoa [45] ja koneoppimismallina *VGG-16*-konvoluutiomallia [54].

Artikkeli [49] esittelee selitysmenetelmillä tuotetut selitykset lämpökarttoina, joista esimerkki on annettu kuvassa 11. Lämpökartoissa on punaisella esitetty ne piirteet, kuva-aineiston tapauksessa pikselit, jotka tukevat kuvan ennustamista oikein. Sinisellä on taas esitetty pikselit, jotka eivät tue lajittelua oikeaan luokkaan. Integroidut gradientit -menetelmällä saatuja vaikutuksia kuvaillaan hienojakoisiksi, sillä menetelmä antaa usein vierekkäisille pikseleille eri suuruisia vaikutuspisteytyksiä. Se antaa myös vaikutuksia tasaisesti sekä ennusteen puolesta että sitä vastaan. Kerroksittaisen relevanssin vaikutuksia voidaan kuvailla jatkuvammiksi, sillä lämpökarttaesityksessä menetelmä korostaa kokonaisia alueita kuvasta. Peittoanalyysi antaa ikkunakoonsa mukaisesti vähemmän tarkkoja vaikutuspisteytyksiä, mutta visualisointi muistuttaa muodoltaan paljon kahden muun algoritmin tuottamia lämpökarttoja.



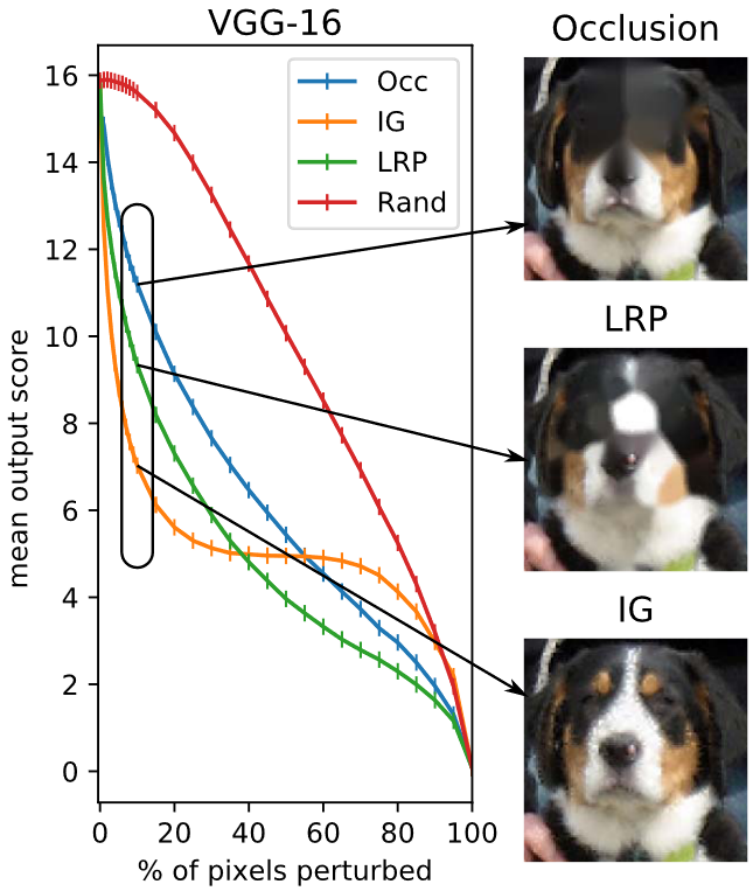
Kuva 11: Artikkelin [49] tulokset esitettynä lämpökarttana ImageNet-kuvasyötteelle [45], jonka luokka on "viaduct" eli maasilta. Punaisella värillä on esitetty positiivisen vaikutuksen piirteet ja sinisellä negatiivisen.

Selitysalgoritmin täsmällisyys määritellään artikkelissa [49] tarkoittamaan sitä, kuinka paljon mallin ennusteen varmuus heikkenee, kun selityksen mukaan vaikuttavimmat piirteet peitetään mallilta. Ennusteen varmuudella tarkoitetaan aineiston mukaista "oikeaa" luokkaa vastaavaa palautusarvoa. Esimerkiksi neuroverkossa tarkasteltaisiin viimeisen kerroksen luokkaa vastaavaa tulosteen arvoa ennen kuin koko viimeiseen kerrokseen sovelletaan *softmax*- tai vastaavaa funktiota lopullisen ennusteen tuottamiseen.

Täsmällisyyttä tutkitaan artikkelissa [49] poistamalla piirteitä datasta niiden vaikutuksen suuruusjärjestyksessä ja samalla tarkastelemalla, miten mallin ennustuksen varmuus muuttuu. Tätä menetelmää kutsutaan artikkelissa pikselifiltteröinniksi, sillä käytössä oleva aineisto koostuu kuvista. Mikäli ennusteen varmuus alenee poistettaessa vaikuttavimmat piirteet, on selitysalgoritmi onnistuneesti löytänyt mallin toiminnan kannalta oleellisimmat piirteet datassa eli menetelmää voidaan kutsua täsmälliseksi. Kuvassa 12 vertaillaan läpikäytyjä selitysmenetelmiä pikselifiltteröinnin avulla: x-akselilla on prosentti vaikuttavimmista pikseleistä, jotka on peitetty mallilta, ja y-akselilla on mallin viimeisen kerroksen tulosteen suuruus oikeaa luokkaa kohden. Arvot on keskiarvoistettu yli testiaineiston. Mitä nopeammin käyrä putoaa, sitä täsmällisempi menetelmä on.

Integroidut gradientit -menetelmä osoittautuu annetun määritelmän mukaan täsmällisimmäksi. Kuvassa 12 mallin ennusteen varmuus putoaa voimakkaasti jo 10% peittoasteella. Kuten aiemmin todettiin, integroidut gradientit -algoritmi tuottaa hienojakoisia selityksiä, joten vaikka mallin ymmärrys datapisteestä heikkenee vaikuttavia pikseleitä poistettaessa, kuva pysyy ymmärrettävänä ihmissilmälle, joka pystyy näkemään kuvan häiriöistä välittämättä. Tällaista esimerkkiä, joka on ihmistulkitsejalle helppo, mutta koneoppimismallille vaikea kutsutaan ristiriitaiseksi esimerkiksi (eng.*adversarial example*) [33]. Artikkeli [63] huomauttaa kuitenkin, että myös kerroksittainen relevanssi pystyy tuottamaan esimerkkejä, jotka voidaan katsoa ristiriitaisiksi, eikä ongelma ole siis vain integroidut gradientit -algoritmissa.

Kerroksittaisen relevanssin tuottamat vaikutuspisteytykset eivät kuvan 12 mukaan ole yhtä täsmällisiä kuin integroidut gradientit -menetelmän. Ennusteen varmuuden keskiarvo kuitenkin laskee aidosti poistettujen pikselien funktiona, toisin kuin in-



Kuva 12: Artikkelin [49] esitys pikselifiltteröintikokeesta VGG-16 mallilla. Sinisellä on esitetty peittoanalyysi, oranssilla integroidut gradientit ja vihreällä kerroksittainen relevanssi. Punaisella merkitään satunnaisten pikselien poistamista, mikä toimii vertailukohtana. Nuolien osoittamissa kohdissa esitetään, miltä yksittäinen datapiste näyttää, kun 10% vaikutukseltaan suurimmista pikseleistä on peitetty.

tegroidut gradientit -menetelmällä, jolla arvot tasoittuvat 30–70% poistettujen pikseleiden välillä. Kerroksittainen relevanssi tuottaa siis vaikutuspisteytyksiä, jotka järjestävät piirteitä vaikuttavuuden mukaan, kun taas integroidut gradientit tunnistavat parhaiten kaikkein merkittävimmät piirteet. Kerroksittaisella relevanssilla myös annettu esimerkkikuva 10% poistoasteella osoittaa, miten sen tuottamat vaikutuspisteytykset hämärtävät kokonaisia alueita kuvasta, esimerkiksi kuvan koiran silmiä. Peittoanalyysi käyttäytyy samoin kuin kerroksittainen relevanssi, mutta sen täsmällisyys on huonompi. Jokainen algoritmi suoriutuu kuitenkin merkittävästi vertailukohtana käytettyä satunnaista pikselien poistoa paremmin.

Selitysalgoritmin käytettävyyteen vaikuttaa myös niiden resurssien kulutus ja kuinka ihmiselle ymmärrettäviä selityksiä niiden perusteella kyetään tuottamaan. Artikkeli [49] ottaa kantaa myös näihin ominaisuuksiin. Selitysmenetelmiä verrataan vaikutuslaskentoja per sekunti. Kunkin menetelmän toteuttamiseen on käytetty PyTorch-kirjastoa Python ohjelmointikielelle.

Peittoanalyysin nopeus riippuu käänteisen neliöllisesti käytetystä ikkunakoosta. Vaikka kuvan 11 mukaan ikkunakoko on suuri, on peittoanalyysi menetelmistä hitain. Sen etuna on kuitenkin yksinkertainen toteutus, jonka ymmärtäminen on helppoa. Integroidut gradientit -menetelmää ja kerroksittaista relevanssia kuvataan artikkelissa [4] nopeiksi selitysalgoritmeiksi, huomauttaen kuitenkin, että integroidut gradientit -menetelmässä iterointi polun yli suurella tarkkuudella hidastaa algoritmia. Artikkelin [49] tulokset ovat samansuuntaisia: kerroksittainen relevanssi tuottaa vaikutuspisteytyksiä nopeiten.

Tulkittavuutta artikkelissa [49] tarkastellaan tuotetun selityksen pakatun tiedostokoon kautta. Oletuksena pidetään sitä, että mitä suurempi tiedostokoko selityksessä syntyy, sitä vähemmän tulkittava selitys on ihmiselle. Integroidut gradientit -algoritmin tuottamat selitykset pärjäävät tällä metriikalla huonosti, sillä rakeisen kuvan pakkauskoko on suuri. Peittoanalyysille ja kerroksittaiselle relevanssille pätee toisin. Koska niiden tuottamat lämpökartat koostuvat isoista yhtenäisistä alueista, on ne helppo pakata pieneen tiedostokokoon.

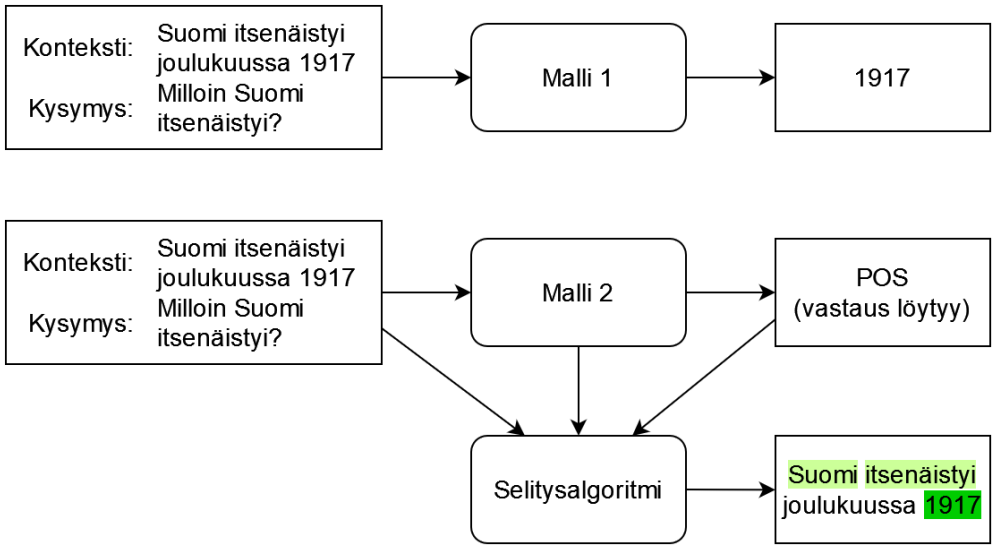
Lopputuloksena saatiin, että jokaisella selitysalgoritmilla on sekä etuja että haittoja. Selitysalgoritmia valitessa onkin hyvä tarkastella, millaisia selityksiä haluaa tuottaa ja millaiset selitykset antavat juuri kyseessä olevasta systeemistä parhainta tietoa. Esimerkiksi selityksen tulkittavuuden vertailu tuotetun selityksen tiedostokoolla ei ole kaikissa tilanteissa tai kaikille datatyypeille sopiva tapa: seuraavassa luvussa integroidut gradientit -algoritmia sovelletaan tekstidataan yksittäisten vaikuttavien sanojen löytämiseksi, jolloin hienojakoiset ja tarkkarajaiset selitykset ovat eduksi.

4 Selitysalgoritmit tiedonlouhinnassa

Selitysalgoritmien käyttökohteet eivät rajoitu vain tapoihin parantaa mallien suorituskykyä tai saada kiinni sovellukseen nähden haitallisia toimintatapoja. Selitysmenetelmät tarjoavat myös tapoja tehdä tiedonlouhintaa (eng.*knowledge discovery*), joka tarkoittaa uuden informaation keräämistä aineistosta. Aineistosta kerätään siis jotain tietoa, jota ei esimerkiksi pelkillä tilastollisilla menetelmillä ole saatu eristettyä. Selitysmenetelmien avulla on esimerkiksi saatu uutta tietoa molekyylisidoksista [52].

Tarkastellaan esimerkkinä mallia, joka on opetettu vastaamaan saamansa (konteksti, kysymys) -parin kysymykseen kontekstin perusteella. Tyypillisiä tämän tyyppisiä malleja voivat olla mallit, jotka etsivät kuvasta tiettyjä objekteja tai mallit, jotka etsivät tekstikontekstista vastauksen annettuun kysymykseen. Esimerkiksi tekstikontekstin ollessa "Suomi itsenäistyi joulukuussa 1917", kysymykselle "Minä vuonna Suomi itsenäistyi?" hyvä ennuste on "1917". Jos tällaista mallia ja kyseistä syötettä analysoidaan selitysalgoritmilla, on todennäköistä, että korkeat vaikutusarvot annetaan sanoille "Suomi", "itsenäistyi" ja erityisesti "1917". Selitysalgoritmi toisin sanoen pystyy osittain vastaamaan annettuun kysymykseen.

Kuvitellaan, että samalla koulutusdatalla on koulutettu malli, joka vastaa kysymyksen sijaan siihen, löytyykö kontekstista vastaus annettuun kysymykseen. Parille ("Helsingin kesäolympialaiset järjestettiin vuonna 1952", "Missä järjestettiin vuoden 1956 kesäolympialaiset?") ei vastausta tulisi löytyä, kun taas edellä annettuun itsenäisyysteemaiseen esimerkkiin malli todennäköisesti ennustaisi vastauksen löytyvän kontekstista. Mitä tapahtuu, kun tällaista mallia selitetään?



Kuva 13: Kaksi mallia, jotka on koulutettu eri tehtävään. Malli 1 vastaa kysymykseen kontekstin perusteella, kun taas malli 2 kertoo, onko kysymykseen mahdollista vastata. Kun mallia 2 selitetään lokaalilla post-hoc menetelmällä, saavutetaan mallia 1 vastaava tulos.

Visualisointi tilanteesta on esitetty kuvassa 13, jossa ensimmäisen mallin ennuste vastaa toisen mallin selitystä. Täten siis tilanteissa, joissa mallilta kysyttävän "kysymyksen" asettelu voi olla vaikeaa esimerkiksi systeemin monimutkaisuuden takia, on mahdollista opettaa malli vastaamaan yksinkertaisempaan kysymykseen. Tämän jälkeen mallia voidaan selittää ja saada systeemistä uutta tietoa.

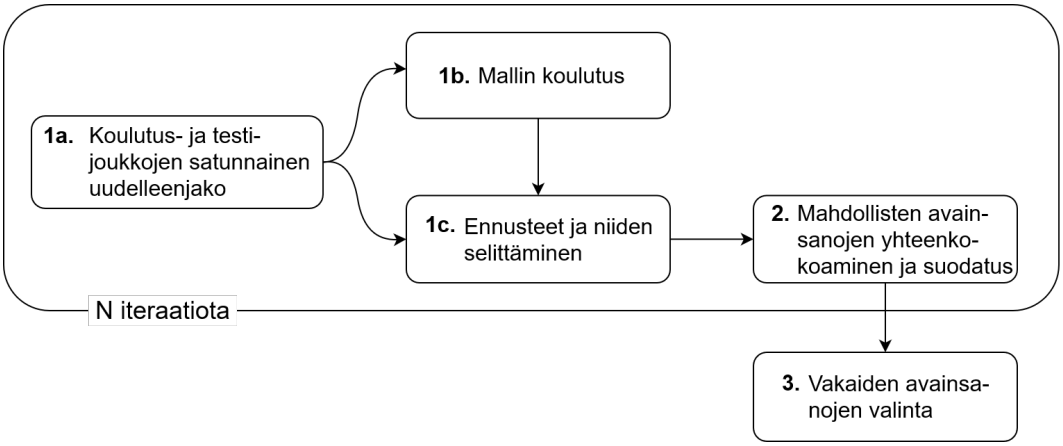
Käydään seuraavaksi läpi sovellus, jossa integroidut gradientit -menetelmää on käytetty tekstejä parhaiten kuvailevien sanojen löytämiseen. Sovellus perustuu artikkeleihin [46] ja [47], joissa määritellyn avainsanamenetelmän kehittämisessä olen ollut itse mukana implementoimassa selitysalgoritmia sekä avainsanojen laatua mittaavien metriikoiden laskemisessa.

4.1 Tekstirekisterien avainsanojen tunnistaminen

Kielitieteessä rekisteri (eng.*register*) määritellään sellaiseksi kielenkäyttötavaksi, joka on ominainen tietylle kielenkäyttötilanteelle. Esimerkiksi samaa aihetta käsittelevässä sanomalehtiartikkelissa ja oppikirjassa esiintyvät kielenkäyttötavat muodostavat kaksi erilaista rekisteriä. Rekisterien välillä olevia eroja, kuten sanavalintojen ja syntaksin eroa, kutsutaan rekisterivaihteluksi (eng.*register variation*). Rekisterivaihtelun tutkimista kutsutaan taas rekisterianalyysiksi [10]. Yksi tapa tutkia rekisteriä on tutkia sanojen frekvenssiä eri rekisterien välillä. Sanoja, jotka esiintyvät tietyssä rekisterissä useammin kuin muissa, kutsutaan kyseessä olevan rekisterin avainsanoiksi (eng.*keywords*) [10]. Avainsanat kuvaavat rekisterien välisiä eroja sanavalinnoissa sekä ilmaisuissa ja on tyypillistä, että ne lasketaan käyttäen tilastollisia menetelmiä.

Rekisterin tunnistaminen annetusta dokumentista koneoppimismalleja käyttäen on rekisterianalyysiongelma. Rekisterin tunnistaminen on hyödyllistä, sillä tekstien lajitteleminen luokkiin rekisterin perusteella mahdollistaa jatko-ongelmien, esimerkiksi sanaluokkatunnistuksen (eng.*part-of-speech tagging*), tehokkaamman ratkaisemisen [17]. Lajittelu juuri rekisterin pohjalta on kannattavaa, sillä artikkelin [6] mukaan rekisteri jaottelee tekstit paremmin jatkoanalysointia varten kuin johonkin muihin tekstien ominaisuuteen perustuva jako. Vaikka moderneilla kielimalleilla saavutetaankin merkittäviä tuloksia, ne ovat erittäin herkkiä saamilleen alkuarvoille, kuten aineiston koulutus-testijaolle sekä mallin sisäisten satunnaisten parmetrien alkuarvoille. Tällöin voi käydä siten, että kaksi tarkkuuksiensa perusteella samoin suoriutuvaa mallia oppivat samasta aineistosta eri asioita, jolloin mallit voivat tuottaa yhdelle esimerkkille hyvin erilaisia tuloksia [39]. Rekisterianalyysissa keskeistä on kuitenkin dokumenttien piirteiden yhteen kokoaminen, joka on koneoppimismalleilla hankalaa edellä mainitun ongelman takia.

Artikkeleissa [46] ja [47] esittelemme *SACX*-menetelmän, jossa rekisterien piirteitä kootaan yhteen perinteisen dokumenttitason lisäksi yli koneoppimismallien koulutuskertojen. SACX perustuu usean koneoppimismallin selittämiseen integroidut gradientit -menetelmällä, minkä jälkeen merkityksellisimmät sanat kerätään yhteen yli koulutettujen mallien, jolloin mallin alkuarvojen merkitys keskiarvoistuu ja on lopputulokselle mahdollisimman pieni. Lisäksi esittelemme saaduille avainsanoille suodatusmenetelmiä sekä kolme metriikkaa saatujen avainsanojen laadulle.



Kuva 14: SACX-menetelmä mukaillen artikkelin [46] kuvaa 1.

SACX-menetelmän yleisrakenne on esitelty kuvassa 14. Vaiheet 1a ja 1b esitellään tarkemmin alaluvussa 4.2, vaiheet 1c ja 2 alaluvussa 4.3. Nämä vaiheet toistetaan N kertaa, jonka seurauksena saadaan N kappaletta avainsana-dokumentti-luokka joukkoja. Artikkelissa [46] käytimme arvoa $N = 100$. Vaiheessa 3, joka esitellään alaluvussa 4.4, valitsimme saaduista joukoista vakaimmat avainsanat.

4.2 Koulutus ja data

SACX-menetelmän kehityksessä käytetty aineisto on *Corpus of Online Registers of English* (*CORE*) [17]. CORE koostuu internetistä kerätyistä dokumenteista, jotka on annotoitu niiden rekisterin mukaan kahdeksaan pääluokkaan ja useisiin alaluokkiin. Menetelmän kehityksessä keskityttiin aineiston pääluokkiin, jotka on esitelty taulukossa 1.

Koneoppimismallina käytettiin *XLM-RoBERTa*-mallia [15]. XLM-RoBERTa on ylikielellinen (eng.*cross-lingual*) kielimalli, joka on koulutettu sadalla eri kielellä Common Crawl ja Wikipedia-aineistolla. XLM-RoBERTa saavuttaa parannuksia erityisesti monikielellisissä tehtävissä, mutta on myös yksikielisissä tehtävissä edeltäjiään tarkempi. Malli koulutettiin moniluokkalajittelutehtävään, toisin sanoen ennustamaan dokumentin rekisteri luokista NA, OP, IN, ID, HI, IP, LY ja SP. XLM-RoBERTa mallin vektorisoijan tapauksessa sanojen sijasta puhutaan tokeneista, eli pienimmistä yksiköistä, joiden vektorisointi katsoo sisältävän kielellistä informaatiota. Esimerkiksi sana “epäsopiva” voidaan jakaa tokeneihin “epä” ja “sopiva”. Avainsanat ovat kuitenkin kokonaisia sanoja. Tästä syystä kumpaakin sanaa, “sana” ja “tokeni”, käytetään tässä luvussa tarkoittamaan eri asioita.

Koulutus toistettiin sata kertaa eri koulutus-testijaoilla. Koska luokien koossa oli suurta vaihtelua, jaot tasaistettiin niin, että luokat jakautuivat tasaisesti molempien joukkojen välillä. Kaikkien koulutuskertojen yli yhdistetyt tulokset on koottu taulukkoon 2 luokittain. Puhekielen SP luokka osoittautui mallille hankalaksi, minkä voi huomata esimerkiksi sen F1-arvon ja keskihajonnan poikkeavuudesta muihin luokkiin nähden. Artikkeleissa [46] ja [47] poikkeavuutta perustellaan luokan SP

Rekisteri	Kuvaus
Narrative (NA)	Uutiset, kerronnalliset blogikirjoitukset, elämänkerrat, kaunokirjallisuus, lehtiartikkelit, muistokirjoitukset
Opinion (OP)	Mielipidekirjoitukset ja blogit, uskonnolliset kirjoitukset, itseapuoppaat
Informational Description (IN)	Asioita ja ihmisiä kuvailevat tekstit, tutkimusartikkelit, tiivistelmät ja yhteenvedot, käyttöehdot, usein-kysytyt-kysymykset, opetusmateriaalit, tietosanakirjat
Interactive Discussion (ID)	Keskustelufoorumit, kommenttikentät
How-to/Instructional (HI)	Oppaat, reseptit, tekninen tuki
Informational Persuasion (IP)	Myyntitarkoitukseen tuotetut tekstit, esseet, pääkirjoitukset
Lyrical (LY)	Laulujen sanat, runous, rukoukset
Spoken (SP)	Haastattelut, puheet, käsikirjoitukset, sanellut tekstit

Taulukko 1: CORE-aineiston rekisterien pääluokat ja niiden kuvaukset.

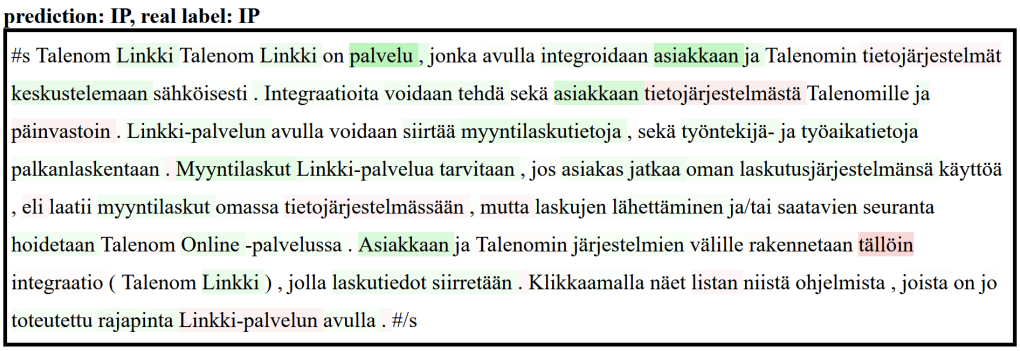
Luokka	F1	SD	#
Lyrical (LY)	0.8292	0.0866	172
Narrative (NA)	0.7870	0.0795	5775
Interactive discussion (ID)	0.7623	0.0787	876
Informational description (IN)	0.6336	0.0662	3399
How-to (HI)	0.5515	0.0719	521
Opinion (OP)	0.5379	0.0839	2854
Informational persuasion (IP)	0.4094	0.0573	531
Spoken (SP)	0.2645	0.2709	206
Micro AVG	0.6510	0.0672	–

Taulukko 2: Mallien (N=100) ennustukyky (F1, keskiarvo), hajonta ja luokan alkioiden määrä artikkelista [46].

instanssien vähyydellä. Vielä pienempi luokka, LY, ei kuitenkaan kärsi instanssien vähyydestä samalla tavalla. Koska luokka LY sisältää esimerkiksi ruonoutta, sen dokumenttien rekisteri eroaa merkittävästi muista luokista, sen tunnistaminen on keskimääräisesti helpompaa.

4.3 Pisteytys integroidut gradientit -menetelmällä

Rekisteritunnistusmallia selitettäessä jokainen syötteen tokeni saa vaikutuspisteytyksen sen mukaan, kuinka paljon kyseinen tokeni vaikutti päätöksentekemiseen mallin sisällä. Korkeat pisteet saavat siis tokenit, joiden myötävaikutus mallin päätökseen oli suurin ja joiden muuttaminen johtaisi suurimpaan muutokseen tulosteessa. Yksinkertaistettuna esimerkkinä mainoksessa sanat, jotka paljastavat tekstin mainokseksi ja joiden poistaminen johtaisi erilaiseen päätökseen tekstin tyypistä, ovat



Kuva 15: Oikea ennuste IP-luokan tekstiin. Korkeat vaikutuspisteet saaneet sanat on väritetty vihreällä ja matalat pisteet saaneet sanat punaisella. Luokan IP teksteissä tyypillisiä sanoja ovat “palvelu”, “asiakas” ja “myynti”, jotka malli on onnistuneesti tunnistanut vaikuttamaan IP-luokittelun suuntaisiksi sanoiksi.

todennäköisesti “hinnan” tai “edullisen” kaltaisia sanoja. Koska yksi sana voi sisältää useamman tokenin, laskettiin sanojen vaikutuspisteytys maksimina kaikista sanan muodostavien tokeneiden vaikutuspisteitstä.

SACX-menetelmässä dokumenttien tokenoidut sisällöt pisteytettiin luvussa 3.2 esitellyllä integroidut gradientit -menetelmällä. IG-vaikutus voidaan laskea jokaista luokkaa c kohden ja se pisteyttää tokeneita positiivisesti, mikäli tokeni viittaa luokkaan c, ja negatiivisesti, mikäli se viittaa muihin luokkiin kuin c. Ohjelmointitoteutus tehtiin käyttämällä *Python*-ohjelmointikielen *Captum*-kirjastoa. Vertailupisteeksi valittiin tyhjän lauseen vektorisointi, kuten luvussa 3.2 perusteltiin.

Kuvassa 15 on esitetty dokumentti, jossa ennuste ja luokka vastaavat toisiaan, joten vaikutus on laskettu luokkaa IP kohden. Vihreällä värillä on esitetty tokenit, joiden vaikutuspisteet ovat korkeita, ja punaisella ne, joiden pisteet ovat matalat. Samoin kuin lopullista avainsana-analyysia tehdessä, on kuvassa luettavuuden parantamiseksi koko sana värjätty sen itseisarvoltaan suurimman tokenin pisteiden mukaan. Kuvasta nähdään selkeästi, että luokan IP kuvaukseen sopivat sanat, kuten “asiakas” ja “palvelu” saavat integroidut gradientit -menetelmällä korkeat pisteet.

SACX-menetelmässä huomioidaan vain tekstit, joiden ennuste on oikea, mutta käänteisesti huomiomalla vain virheelliset ennusteet voi menetelmää käyttää virheanalyysiin. Integroidut gradientit -algoritmin tuottamat vaikutuspisteet lasketaan kyseistä oikeaa luokkaa kohden. Lopputuloksena saadaan jokaiselle tokenille t, dokumentille d ja luokalle c määritelmän 8 mukaiset pisteytykset

$$s_{t,d,c} = \mathrm{IG}_t(d),$$

missä integroitavana funktiona on kutakin luokkaa c vastaava dimensio mallin funktiosta F. Pisteet sanoille lasketaan samoin kuin edellä jo todettiin: sanan sisältäessä useamman tokenin, käytetään itseisarvoltaan suurinta arvoa koko sanalle. Jokaisen dokumentin tulokset normalisoidaan L_2-normilla, jotta pisteet ovat verrattavissa dokumenteittain. Lopulta valitaan jokaisesta dokumentista n sanaa, jotka saivat kor-

keimmat pisteet. Kun tokenit yhdistetään tällä tavalla sanoiksi, säilytetään muutakin kuin sanojen merkitykseen liittyvää informaatiota, kuten sanojen taivutusmuodot. Englanninkielisessä tekstidatassa tämä ei muodosta suurta ongelmaa, mutta esimerkiksi suomenkielisten dokumenttien kanssa sanojen lemmatisointi, eli perusmuotoon palauttaminen, voi olla yhdistämisen jälkeen tarpeellista. Toisaalta myös sanojen taivutusmuodot voivat kertoa rekisteristä, erityisesti luokan LY tapauksessa. Lopputuloksena iterointivaiheesta saadaan $N = 100$ kappaletta listoja, joissa jokaisesta oikeasta dokumentti-ennuste -parista on valittuna n kappaletta parhaita avainsanakandidaatteja.

4.4 Avainsanat ja tulosten arviointi

Avainsanakandidaatit N:n iteraation jälkeen kerätään yhteen ja niistä valitaan luokan c avainsanoiksi ne, jotka on valittu ainakin t prosentissa iteraatioissa parhaiksi avainsanakandidaateiksi. Jokaisella iteraatiolla ei siis valita samoja sanoja, sillä mallin alkuarvoista riippuva koulutus määrittelee, mitkä dokumentit saadaan ennustettua oikein ja myös mitkä sanat ovat vaikutuksen kannalta merkittävimpiä. Sanan valinnan lukumäärää kutsutaan valintafrekvenssiksi. Valintafrekvenssiltään ehdon täyttäviä avainsanoja nimitetään stabiileiksi avainsanoiksi. Lopulliset stabiilit avainsanat on esitelty taulukossa 4. Luokalle SP stabiileja avainsanoja ei saatu yhtään kappaletta, mikä johtuu käytetyn mallin ongelmista tunnistaa luokka SP.

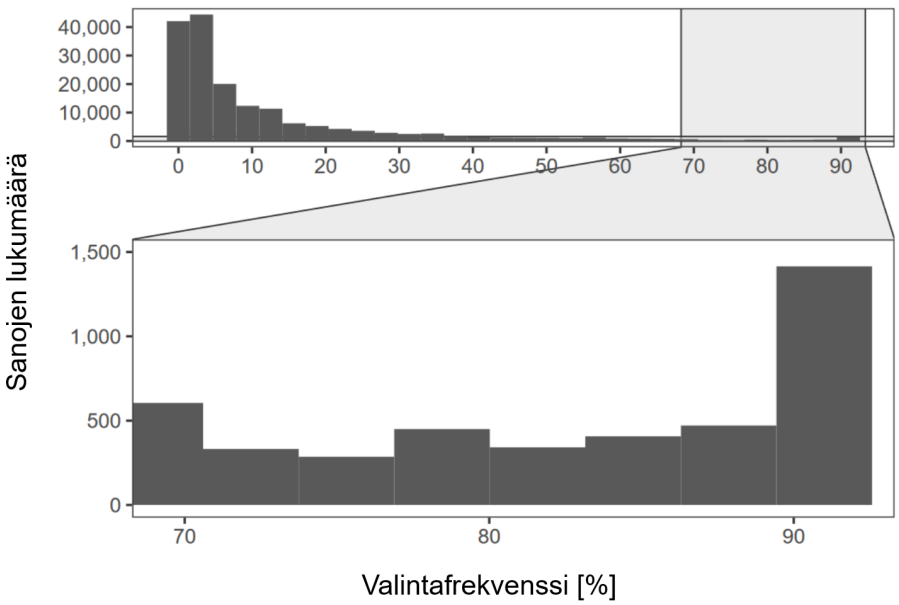
Valintafrekvenssin käyttäminen erottelemaan stabiilit ja epästabiilit avainsanat nojaa oletukseen, että mallien iteroiminen todella tuottaa sanoja, jotka valitaan toisia useammin avainsanakandidaateiksi. SACX-menetelmän tulisi siis arvon t avulla jakaa sanat kahteen ryhmään, epästabiileihin ja stabiileihin avainsanoihin. Tämän takia avainsanojen määrästä luotiin histogrammi prosentuaalisen valintafrekvenssin t funktiona. Histogrammi on esitetty kuvassa 16. Sanoja, joiden valintafrekvenssi on suuri, saadaan enemmän kuin sanoja, joiden valintafrekvenssi on keskitasoa. Suurin osa sanoista valitaan harvoin, mikä on SACX-menetelmän kannalta toivottua: harvoin valittujen sanojen poistaminen käyttämällä valintafrekvenssiarvoa t poistaa mallien alustukseen liittyvää satunnaisuutta.

Määritellään seuraavaksi tapoja tarkastella tulosten laatua. Merkitään luokalle c valittuja stabiileja avainsanoja merkinnällä K_c. Eriytyneisyys (eng.*distinctiveness*) määriteltiin olemaan suhde ainoastaan luokan c avainsanoissa esiintyvien ja kaikkien luokassa c esiintyvien avainsanojen välillä:

$$\mathrm{Dist}_{int}(c) = \frac{\{k \mid k \in K_c \setminus K_{\neg c}\}}{|K_c|}.$$

Kun luokkien joukkoa merkitään symbolilla C, kaikkien valittujen avainsanojen kuvailuun voidaan käyttää eriytyneisyyden luokittaista keskiarvoa (eng.*macro average*)

$$\mathrm{Dist}_{int} = \frac{1}{|C|} \sum_{c \in C} \frac{\{k \mid k \in K_c \setminus K_{\neg c}\}}{|K_c|}.$$



Kuva 16: Artikkelin [46] esitys avainsanojen valintafrekvenssistä histogrammina. Suurin osa sanoista valitaan avainsanakandidaateiksi vain pienessä osassa iteraatioita, ja sanojen määrä vähenee valintafrekvenssin kasvaessa. Suurilla valintafrekvenssin arvoilla (>85-90%) saadaan kuitenkin enemmän avainsanoja, jotka voidaan valita SACX-menetelmässä stabiileiksi.

Avainsanojen informatiivisuus perustuu eritteleviin ominaisuuksiin rekisterien välillä. Eriytyneisyyden suuret arvot kuvaavat sitä, että luokan avainsanat ovat muihin luokkiin nähden erillisiä. Jos taas eriytyneisyyden arvot ovat pieniä, voi olla, että avainsanoiksi on suurimmaksi osaksi valikoitunut aineistossa yleisiä sanoja, jotka on valittu samanaikaisesti useamman luokan avainsanoiksi.

Merkitään luokan c dokumenttien joukkoa merkinnällä D_c. Luokan dokumenteilla tarkoitetaan tekstejä, eli joukkoa sanoja, jotka on annotoitu luokalla c. Peittävyys (eng.*coverage*) määriteltiin sen mukaan, kuinka monta kyseisen luokan avainsanaa esiintyvät kyseisen luokan teksteissä keskimäärin. Mikäli peittävyys luokalle c on esimerkiksi 5, se tarkoittaa, että katsottaessa satunnaista luokan c dokumenttia se sisältää keskimäärin 5 luokan c avainsanaa. Samoin kuin yllä, ottamalla luokittainen keskiarvo saadaan kaikkia avainsanoja mittaava peittävyysarvo

$$\mathrm{Cov} = \frac{1}{|C|} \sum_{c \in C} \frac{1}{|D_c|} \sum_{D \in D_c} \frac{\{k \mid k \in K_c \cap D\}}{|K_c|}.$$

Viimeinen SACX-menetelmän arviointiin käytetty metriikka on ulkoinen eriytyneisyys (eng.*extrinsic distictiveness*),

$$\mathrm{Dist}_{ext} = \frac{\mathrm{Cov} - \mathrm{XCov}}{\mathrm{Cov}}.$$

Ulkoinen eriytyneisyys on koko avainsanajoukkoa kuvaava mitta, joka saa arvon 0, kun luokkien avainsanajoukkojen välillä ei ole eroja ja 1, kun kaikkien luokkien avainsanat ovat erilliset. Kaavassa esiintyvä ristipeittävyys XCov (eng.*Cross-coverage*) määriteltiin

$$\text{XCov} = \frac{1}{|C|} \sum_{c \in C} \frac{1}{|D_{\neg c}|} \sum_{D \in D_{\neg c}} \frac{\{k \mid k \in K_c \cap D\}}{|K_c|},$$

jonka voi ajatella mittaavan käänteistä peittävyyttä: se kuvaa, kuinka monta luokan c sanaa on keskimäärin dokumenteissa, jotka eivät kuulu luokkaan c, eli dokumenteissa $D_{\neg c}$.

Artikelissa [46] SACX-menetelmän yhtenä vertailukohtana käytetään TF-IDF – tilastollista arvoa. TF-IDF (eng.*term frequency–inverse document frequency*) on tapa esittää tekstidokumentit vektoreina, joissa vektorin kukin dimensio esittää yhtä sana-yksikköä ja vektorin arvo sitä, kuinka informatiivinen sana on kussakin dokumentissa TF-IDF metriikan perusteella [51]. TF-IDF toimii siis myös tapana ilmoittaa dokumenttien avainsanat valitsemalla kustakin dokumentista suurimman pisteytyksen saamat sanat. Toisena vertailukohtana käytettiin tukivektorikonemallia [51] käyttävää avainsanametodia. Tukivektorimalli (SVM) koulutettiin samoilla koulutus-testijaoilla kuin XLM-RoBERTa SACX-menetelmässä. Tulokset vertailukohtien kanssa on esitelty taulukossa 3. Tulosten mukaan SACX-menetelmä pystyy erittelemään rekisterien avainsanoja muihin malleihin verrattavalla tavalla ja saavuttaa parannuksia erityisesti TF-IDF-menetelmään verrattuna.

Vertailukohtien valitsemien avainsanojen ero SACX-menetelmän tuottamiin avainsanoihin on myös merkittävä: SACX-menetelmän avainsanat sisältävät tilastollisesti merkittävästi enemmän leksikaalisia sanoja (eng.*content words*) ja vähemmän funktiosanoja (eng.*function words*) kuin vertailukohtien avainsanat. Leksikaalisia sanoja ovat esimerkiksi verbit, adjektiivit sekä substantiivit, joita SACX-menetelmä tuottaa kaikkein eniten. Funktiosanoja ovat kieliopilliset sanat, kuten pronominit ja prepositiot. Vaikka SACX-menetelmä tuottaa funktiosanoja vähän, tuotetut sanat vastaavat käytetystä aineistosta aiemmin valittua funktiosanastoa [7].

Taulukosta 4 nähdään, että kaikkiin luokkiin (pl. SP) on saatu valittua sanoja, jotka selvästi kuvaavat luokan rekisteriä. Lisäksi kuten yllä todettiin, suuri osa valituista sanoista edustaa leksikaalisia sanoja. Esimerkiksi luokassa ID jokaista ensimmäisistä 15 sanasta voidaan kuvailla leksikaalikseksi sanaksi. Lisäksi kaikissa luokissa valintafrekvenssi pysyy vähintään kohtalaisen stabiilina. SACX-mentelmän tuottamia avainsanoja voidaan siis kuvailla informatiivisiksi sekä annettujen metriikkojen perusteella laadukkaiksi.

Metodi	Eriytyneisyys (%)	Ulk. eriytyneisyys (%)	Peittävyys (%)
SACX	82.57	44.27	10.08
SVM	91.43	39.94	10.53
TF-IDF	29.86	43.92	10.37

Taulukko 3: SACX-menetelmä verrattuna muihin avainsanamenetelmiin.

— How-to (HI) —

Word	Score	SF(%)
how	0.5206	97
howto	0.4024	87
diy	0.3538	77
recipe	0.3368	97
recipes	0.2965	97
to	0.2425	96
ingredients	0.2344	97
tutorial	0.2311	96
tutorials	0.2268	78
tips	0.2194	97
tip	0.2012	94
navigation	0.1910	77
remove	0.1870	97
build	0.1849	91
preheat	0.1831	87

— Inter. Discussion (ID) —

Word	Score	SF(%)
faq	0.5806	98
question	0.5514	98
answer	0.4815	98
forum	0.4733	98
answers	0.4554	98
thread	0.4202	98
forums	0.4007	98
re	0.3786	98
discuss	0.3723	98
answered	0.3645	93
replies	0.3554	98
threads	0.3490	98
resolved	0.3284	98
quote	0.3280	98
answerer	0.3188	98

— Inform. Description (IN) —

Word	Score	SF(%)
abstract	0.5633	98
storyline	0.4589	88
faqs	0.4412	95
faq	0.4198	95
aspect	0.3403	97
introduction	0.3057	98
summary	0.3023	98
contents	0.3016	98
abstracts	0.2838	90
bio	0.2635	92
disclaimer	0.2538	98
meta	0.2519	74
profiles	0.2500	86
downloads	0.2471	72
dictionary	0.2441	98

— Inform. Persuasion (IP) —

Word	Score	SF(%)
description	0.5049	96
isbn	0.4181	70
product	0.2804	96
book	0.2776	96
important	0.2603	93
shop	0.2431	73
details	0.2322	93
amazon	0.2131	96
reviews	0.2034	96
buy	0.1807	96
available	0.1787	96
review	0.1772	96
item	0.1732	76
package	0.1711	70
products	0.1681	96

— Lyrical (LY) —

Word	Score	SF(%)
lyrics	0.4117	97
poem	0.2839	75
written	0.1583	81
sorry	0.1471	70
lyricsmode	0.1462	91
truth	0.1351	91
songs	0.1343	73
yeah	0.1337	95
tired	0.1331	79
finally	0.1314	76
tonight	0.1312	87
something	0.1302	97
heaven	0.1300	77
lord	0.1299	74
fucking	0.1287	79

— Narrative (NA) —

Word	Score	SF(%)
bundesliga	0.3588	98
afp	0.3462	98
nba	0.3455	98
ufc	0.3327	98
blog	0.3307	98
playoffs	0.3283	98
nfl	0.3263	98
wordpress	0.3253	87
flickr	0.3248	95
playoff	0.3075	98
reuters	0.3073	98
uefa	0.3065	98
zlatan	0.3055	98
psg	0.3038	97
responses	0.3000	92

— Opinion (OP) —

Word	Score	SF(%)
review	0.5456	98
weblog	0.5028	72
psalm	0.4444	95
feminist	0.3376	94
tips	0.3292	92
blog	0.3279	98
bible	0.3250	98
thursday	0.3000	98
lgbt	0.2957	87
eucharistic	0.2925	71
monday	0.2899	97
tuesday	0.2873	98
wednesday	0.2861	98
testament	0.2780	98
post	0.2688	98

Taulukko 4: Parhaiten keskimäärin arvotetut (Score) stabiilit avainsanat. SF (valintafrekvenssi) ilmaisee, kuinka monessa iteraatiossa sana valittiin kandidaatiksi.

5 Loppupäätelmät

Tekoälyn kehitys on näkynyt pelkästään jo tämän tutkielman kirjoittamisen aikana. Yliopiston tiloissa oleskellessani olen huomannut, kuinka tekoälyä käytetään harjoitusten tekemisessä ja olen myös osallistunut kurssille, jossa opetuksessa käytetystä esimerkkikoodista osa on kirjoitettu tekoälyä käyttäen. Tämän ilmeisen tarpeen vuoksi Turun yliopisto julkaisi 23.3.2023 lausunnon tekoälyn käytöstä opiskelussa ja opetuksessa, jossa tekoälyyn opetus- ja opiskeluvälineenä kannustettiin suhtautumaan "uteliaan kriittisesti".

Etenkin yliopistolla tekoälyn käyttöön liittyy jonkinlaista ylemmyyden tunnetta. Koska suurin osa meistä vähintään kuvittelee tietävänsä tekoälystä paljon, on helppo ajatella, että kyllä me ymmärrämme, milloin tekoälyn käyttäminen on perusteltavaa ja tekoälyn käytön ongelmat johtuvat mielestämme kaupallistamisessa ja sen käyttämisessä sopimattomissa tilanteissa. Asia ei tietenkään näin ole, kuten luvussa 1 osoitettiin: vaikka koneoppimismallia sovellettaisiinkin oikeisiin tilanteisiin ja sen koulutukseen käytetty data olisi edustavaa ja harhatonta, on sen toiminta silti riippuvainen lopulta ihmisen antamasta kysymysasettelusta eikä meillä ihmisillä ole tehokasta tapaa tunnistaa omia sisäisiä ennakkoluulojamme.

Selitysmenetelmät auttavat meitä kumoamaan joitakin näistä harhoista. Ne auttavat meitä saavuttamaan parempia tuloksia kertomalla, missä menemme vikaan. Ne saavat meidät pohtimaan, miksi keräämämme aineisto näyttää vinoutuneelta tai mitä jätimme huomioimatta. Ne mahdollistavat sen, että tulevaisuudessa meillä on mahdollisuus ottaa selvää, miksi lainahakemustamme ei hyväksytty pankissa. Selitysmenetelmillä tällaisten tulosten saavuttaminen on realistista, mutta näiden hyötyjen saavuttaminen nojaa siihen oletukseen, että selitysmenetelmiä käytetään vastuullisesti, mikä ei eroa kovinkaan paljon nykyisestä toiveesta käyttää itse tekoälyä vastuullisesti. Tämä on todennäköisesti asia, johon on otettava tulevaisuudessa kantaa. Selitysmenetelmien vastuulliseen käyttöön kuuluu myös ympäristöaspekti, sillä esimerkiksi luvussa 4 esitelty avainsanamenetelmä kuluttaa huomattavan määrän laskentatehoa saavuttaessaan keskikokoisia parannuksia verrattuna vähemmän virtaa kuluttuviin menetelmiin. Tämä "vähenevien tuottojen laki" on pohdinnan arvoinen asia selitysmenetelmän käyttöönottoa harkittaessa.

Luvussa 3 esitellyt algoritmit, integroidut gradientit ja kerroksittainen relevanssi, ovat molemmat alan hyväksymiä esimerkkejä toimivista selitysmenetelmistä. Menetelmiä vertaillessa huomattiin, että ne korostavat erilaisia asioita tuottamissaan selityksissä, ainakin kuva-aineistolla. Lisäksi nähtiin, miten toisella menetelmistä pystyttiin tuottamaan esimerkki, josta ihmisen ja koneoppimismallin tekemät tulkinnat olivat ristiriitaiset. Kumpikin näistä seikoista avaa mahdollisuuden valita käytetty selitysmenetelmä omien intressien mukaisesti ja silti tarjota toiselle osapuolelle, esimerkiksi kuluttajalle, yleisesti päteväksi katsottua selitystä.

Pinnallisesti haluan paneutua vielä yhteen varsinkin kielimallien käyttöön liittyvään seikkaan, joka mielestäni hämärtää ihmisen kykyä tulkita mallia. Me olemme kehit-

täneet älykkäitä systeemejä jo vuosikymmeniä, mutta edelleen meidät innostuneimmiksi saavat mallit, joiden kanssa tulemme itse ymmärretyiksi. Kaikkein selvimpänä esimerkkinä tästä ovat kielimallit: mikäli tekoälyn älykkyyttä voitaisiin mitata objektiivisesti jollain mittarilla ja loisimme kaksi tismalleen yhtä älykästä tekoälyä, toinen tekemään data-analyyttisia tehtäviä ja toinen keskustelemaan ihmisen kanssa, olisimme varmasti enemmän innoissamme jälkimmäisestä. Kun systeemi vastaa meille, meidän on helpompi ohittaa sen ongelmia samoin kuin ohitamme toistemme pieniä lipsahduksia, vaikka kyse ei ole samanlaisesta ongelmasta.

Kuten luvun alussa jo viitattiin tekoälyn käytön räjähtämiseen tutkielman kirjoittamisen aikana, on mielestäni lopuksi sopivaa kysyä tällä hetkeltä merkittävimmältä tekoälyltä, mitä se on oppinut selitettävästä tekoälystä. ChatGPT [41] on koulutettu GPT-3.5 (*Generative Pre-trained Transformer*) pohjalta hienosäätämällä se erityisesti keskusteluun ja se on todennäköisesti altistunut jonkinlaiselle konsensukselle siitä, mitä selitettävä tekoäly on sen koulutusdatan ja kouluttajien mielestä. Lisäksi haluan vielä viimeiseksi tarjota kontekstia siihen, millaista tekoäly on ollut tutkielman kirjoittamisen aikoihin.

Tulevaisuuden selitettävän tekoälyn (XAI) suhteen olen optimistinen. XAI on yhä tärkeämpi ala, joka pyrkii tekemään tekoälystä ihmiskunnalle läpinäkyvämpää ja ymmärrettävämpää. Sen merkitys kasvaa, kun tekoälysovellukset yleistyvät päivittäisessä elämässämme. [...] Lisäksi useat organisaatiot, mukaan lukien Euroopan unioni ja Yhdysvaltain hallitus, ovat aloittaneet määräysten käyttöönoton, jotka vaativat tekoälyjärjestelmien olevan läpinäkyviä ja selitettäviä. Tämä sääntelypaine todennäköisesti kannustaa kehittämään yhä läpinäkyvämpiä tekoälyteknologioita. [...] Kaiken kaikkiaan tulevaisuus selitettävän tekoälyn suhteen näyttää lupaavalta, ja uskon, että tämä ala jatkaa kasvuaan ja kehitystään, kun pyrimme tekemään tekoälystä ihmiskunnalle ymmärrettävämpää ja luotettavampaa.

ChatGPT (julkaisu 3/5/2023)

Viitteet

[1] Amina Adadi ja Mohammed Berrada. "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)". *IEEE Access* 6 (2018), s. 52138–52160. DOI: 10.1109/ACCESS.2018.2870052.

[2] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts ja Peter Walter. "General Principles of Cell Communication". Teoksessa: 4. painos. New York: Garland Science, 2002. Luku "Molecular Biology of the Cell".

[3] Richard Ambrosino, Bruce Buchanan, Gregory Cooper ja Michael Fine. "The use of misclassification costs to learn rule-based decision support models for cost-effective hospital admission strategies". *Proceedings / the ... Annual Symposium on Computer Application [sic] in Medical Care. Symposium on Computer Applications in Medical Care* (helmikuu 1995), s. 304–8.

[4] Marco Ancona, Enea Ceolini, Cengiz Öztireli ja Markus Gross. "Gradient-Based Attribution Methods". Teoksessa: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Toim. Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen ja Klaus-Robert Müller. Cham: Springer International Publishing, 2019, s. 169–203. ISBN: 978-3-030-28954-6. DOI: 10.1007/978-3-030-28954-6_10. URL: https://doi.org/10.1007/978-3-030-28954-6_10.

[5] Dzmitry Bahdanau, Kyunghyun Cho ja Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

[6] Douglas Biber. *Corpus Linguistics and Linguistic Theory* 8.1 (2012), s. 9–37. DOI: doi:10.1515/cllt-2012-0002. URL: https://doi.org/10.1515/cllt-2012-0002.

[7] Douglas Biber ja Jesse Egber. "Using grammatical features for automatic register identification in an unrestricted corpus of documents from the open web". *Journal of Research Design and Statistics in Linguistics and Communication Science* (2016), s. 3–36.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. 1. painos. Springer, 2006. ISBN: 978-0387-31073-2.

[9] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle ja John Guttag. "What is the state of neural network pruning?" *Proceedings of machine learning and systems* 2 (2020), s. 129–146.

[10] Keith Brown ja Jim Miller. *Dictionary*. Cambridge University Press, 2013, s. 247–249, 370–391.

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray,

Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever ja Dario Amodei. "Language Models are Few-Shot Learners". Teoksessa: *Advances in Neural Information Processing Systems*. Toim. H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan ja H. Lin. Vol. 33. Curran Associates, Inc., 2020, s. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[12] Nadia Burkart ja Marco F. Huber. "A Survey on the Explainability of Supervised Machine Learning". *Journal of Artificial Intelligence Research* 70 (2021), s. 245–317. DOI: 10.1613/jair.1.12228. URL: https://doi.org/10.16132Fjair.1.12228.

[13] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm ja Noemie Elhadad. "Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-Day Readmission". Teoksessa: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15. Sydney, NSW, Australia: Association for Computing Machinery, 2015, s. 1721–1730. ISBN: 9781450336642. DOI: 10.1145/2783258.2788613. URL: https://doi.org/10.1145/2783258.2788613.

[14] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk ja Yoshua Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". Teoksessa: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, lokakuu 2014, s. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://aclanthology.org/D14-1179.

[15] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer ja Veselin Stoyanov. *Unsupervised Cross-lingual Representation Learning at Scale*. Online, heinäkuu 2020. DOI: 10.18653/v1/2020.acl-main.747. URL: https://aclanthology.org/2020.acl-main.747.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee ja Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". Teoksessa: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019, s. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[17] Jesse Egbert, Douglas Biber ja Mark Davies. "Developing a bottom-up, user-based method of web register classification". *Journal of the Association for Information Science and Technology* 66 (2015).

[18] Dumitru Erhan, Aaron Courville ja Y. Bengio. "Understanding Representations Learned in Deep Architectures" (toukokuu 2023).

[19] Francis Bach Ethem Alpaydin. *Introduction to Machine Learning*. 3. painos. MIT Press, 2014. ISBN: 978-0-2623-2574-5.

[20] Ruth Fong ja Andrea Vedaldi. "Explanations for Attributing Deep Neural Network Predictions". Teoksessa: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Toim. Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen ja Klaus-Robert Müller. Cham: Springer International Publishing, 2019, s. 169–203. ISBN: 978-3-030-28954-6. DOI: 10.1007/978-3-030-28954-6_10. URL: https://doi.org/10.1007/978-3-030-28954-6_10.

[21] Amirata Ghorbani, Abubakar Abid ja James Zou. "Interpretation of Neural Networks Is Fragile". *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (heinäkuu 2019), s. 3681–3688. DOI: 10.1609/aaai.v33i01.33013681. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4252.

[22] Leilani Gilpin, David Bau, Ben Yuan, Ayesha Bajwa, Michael Specter ja Lalana Kagal. "Explaining Explanations: An Overview of Interpretability of Machine Learning". Teoksessa: lokakuu 2018, s. 80–89. DOI: 10.1109/DSAA.2018.00018.

[23] Ian Goodfellow, Yoshua Bengio ja Aaron Courville. *Deep learning*. Adaptive computation and machine learning. London, England: The MIT Press, 2016. ISBN: 9780262035613.

[24] Bryce Goodman ja Seth Flaxman. "European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation"". *AI Magazine* 38.3 (lokakuu 2017), s. 50–57. DOI: 10.1609/aimag.v38i3.2741. URL: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2741.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren ja Jian Sun. "Deep Residual Learning for Image Recognition". Teoksessa: *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '16. Las Vegas, NV, USA: IEEE, kesäkuu 2016, s. 770–778. DOI: 10.1109/CVPR.2016.90. URL: http://ieeexplore.ieee.org/document/7780459.

[26] Bernease Herman. "The Promise and Peril of Human Evaluation for Model Interpretability" (marraskuu 2017).

[27] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal ja Heimo Müller. "Causability and explainabilty of artificial intelligence in medicine". *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9 (heinäkuu 2019), e1312. DOI: 10.1002/widm.1312.

[28] J. J. Hopfield. "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the National Academy of Sciences* 79.8 (1982), s. 2554–2558. DOI: 10.1073/pnas.79.8.2554. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.79.8.2554. URL: https://www.pnas.org/doi/abs/10.1073/pnas.79.8.2554.

[29] Sebastian Lapuschkin, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller ja Wojciech Samek. "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". *PLoS ONE* 10 (heinäkuu 2015), e0130140. DOI: 10.1371/journal.pone.0130140.

[30] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek ja Klaus-Robert Müller. "Unmasking Clever Hans predictors and assessing what machines really learn". *Nature Communications* 10.1 (maaliskuu 2019). DOI: 10.1038/s41467-019-08987-4.

[31] Jimmy Lei Ba, Jamie Ryan Kiros ja Geoffrey E. Hinton. "Layer Normalization". *arXiv e-prints*, arXiv:1607.06450 (heinäkuu 2016), arXiv:1607.06450. DOI: 10.48550/arXiv.1607.06450. arXiv: 1607.06450 [stat.ML].

[32] Seppo Linnainmaa. "Taylor expansion of the accumulated rounding error" (1976), s. 146–160. URL: https://doi.org/10.1007/BF01931367.

[33] Zachary C. Lipton. "The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability is Both Important and Slippery." *Queue* 16.3 (kesäkuu 2018), s. 31–57. ISSN: 1542-7730. DOI: 10.1145/3236386.3241340. URL: https://doi.org/10.1145/3236386.3241340.

[34] Yin Lou, Rich Caruana ja Johannes Gehrke. "Intelligible Models for Classification and Regression". Teoksessa: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '12. Beijing, China: Association for Computing Machinery, 2012, s. 150–158. ISBN: 9781450314626. DOI: 10.1145/2339530.2339556. URL: https://doi.org/10.1145/2339530.2339556.

[35] Thang Luong, Hieu Pham ja Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". Teoksessa: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, syyskuu 2015, s. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://aclanthology.org/D15-1166.

[36] Aravindh Mahendran ja Andrea Vedaldi. "Visualizing Deep Convolutional Neural Networks Using Natural Pre-images". *International Journal of Computer Vision* 120.3 (toukokuu 2016), s. 233–255. DOI: 10.1007/s11263-016-0911-8.

[37] Stephen Marsland. *Machine Learning: an Algorithmic Perspective*. 2. painos. CRC Press Inc., 2015. ISBN: 978-1-4665-8333-7.

[38] David Martens, Bart Baesens ja Tony Van Gestel. "Decompositional Rule Extraction from Support Vector Machines by Active Learning". *Knowledge and Data Engineering, IEEE Transactions on* 21 (maaliskuu 2009), s. 178–191. DOI: 10.1109/TKDE.2008.131.

[39] R. Thomas McCoy, Junghyun Min ja Tal Linzen. "BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance". Teoksessa: *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*. Online: Association for Computational Linguistics, marraskuu 2020, s. 217–227. DOI: 10.18653/v1/2020.blackboxnlp-1.21. URL: https://aclanthology.org/2020.blackboxnlp-1.21.

[40] Tomas Mikolov, Kai Chen, Greg Corrado ja Jeffrey Dean. ”Efficient Estimation of Word Representations in Vector Space”. *Proceedings of Workshop at ICLR* 2013 (tammikuu 2013).

[41] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[42] Jeffrey Pennington, Richard Socher ja Christopher D. Manning. ”GloVe: Global Vectors for Word Representation”. Teoksessa: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, s. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[43] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev ja Percy Liang. ”SQuAD: 100,000+ Questions for Machine Comprehension of Text”. Teoksessa: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, marraskuu 2016, s. 2383–2392. DOI: 10.18653/v1/D16-1264. URL: https://aclanthology.org/D16-1264.

[44] D. Rumelhart, G. Hinton ja R. Williams. ”Learning representations by back-propagating errors”. 323 (1986), s. 533–536. URL: https://doi.org/10.1038/323533a0.

[45] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg ja Li Fei-Fei. ”ImageNet Large Scale Visual Recognition Challenge”. *International Journal of Computer Vision (IJCV)* 115.3 (2015), s. 211–252. DOI: 10.1007/s11263-015-0816-y.

[46] Samuel Rönnqvist, Aki-Juhani Kyröläinen, Amanda Myntti, Filip Ginter ja Veronika Laippala. ”Explaining Classes through Stable Word Attributions”. Teoksessa: *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, toukokuu 2022, s. 1063–1074. DOI: 10.18653/v1/2022.findings-acl.85. URL: https://aclanthology.org/2022.findings-acl.85.

[47] Samuel Rönnqvist, Amanda Myntti, Aki-Juhani Kyröläinen, Sampo Pyysalo, Veronika Laippala ja Filip Ginter. *Explaining Classes through Word Attribution*. 2021. DOI: 10.48550/ARXIV.2108.13653. URL: https://arxiv.org/abs/2108.13653.

[48] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin ja Klaus-Robert Müller. *Evaluating the visualization of what a Deep Neural Network has learned*. Syyskuu 2015.

[49] Wojciech Samek, Gregoire Montavon, Sebastian Lapuschkin, Christopher J. Anders ja Klaus-Robert Müller. ”Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications”. *Proceedings of the IEEE* 109.3 (2021), s. 247–278. DOI: 10.1109/jproc.2021.3060483. URL: https://doi.org/10.11092Fjproc.2021.3060483.

[50] Wojciech Samek, Thomas Wiegand ja Klaus-Robert Müller. *Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models*. 2017. DOI: 10.48550/ARXIV.1708.08296. URL: https://arxiv.org/abs/1708.08296.

[51] Claude Sammut ja Geoffrey I. Webb. *Encyclopedia of Machine Learning and Data Mining*. Springer New York, NY, 2017.

[52] Kristof T. Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R. Müller ja Alexandre Tkatchenko. "Quantum-chemical insights from deep tensor neural networks". *Nature Communications* 8, 13890 (2017). DOI: https://doi.org/10.1038/ncomms13890.

[53] Karen Simonyan, Andrea Vedaldi ja Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". Teoksessa: *Workshop at International Conference on Learning Representations*. 2014.

[54] Karen Simonyan ja Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". Teoksessa: *International Conference on Learning Representations*. 2015.

[55] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas ja Martin Wattenberg. "SmoothGrad: removing noise by adding noise" (2017).

[56] Pascal Sturmfels, Scott Lundberg ja Su-In Lee. "Visualizing the Impact of Feature Attribution Baselines". *Distill* (2020). DOI: 10.23915/distill.00022. URL: https://distill.pub/2020/attribution-baselines.

[57] Mukund Sundararajan, Ankur Taly ja Qiqi Yan. "Axiomatic Attribution for Deep Networks". Teoksessa: *Proceedings of the 34th International Conference on Machine Learning*. Toim. Doina Precup ja Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, elokuu 2017, s. 3319–3328. URL: https://proceedings.mlr.press/v70/sundararajan17a.html.

[58] Ilya Sutskever, Oriol Vinyals ja Quoc V Le. "Sequence to Sequence Learning with Neural Networks". Teoksessa: *Advances in Neural Information Processing Systems*. Toim. Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence ja K.Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser ja Illia Polosukhin. "Attention is all you need". *Advances in neural information processing systems* 30 (2017).

[60] Paul Werbos ja Paul John. "Beyond regression : new tools for prediction and analysis in the behavioral sciences" (1974).

[61] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest ja Alexander Rush. "Transformers: State-of-the-Art Natural Language Processing". Teoksessa: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, lokakuu 2020, s. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. URL: https://aclanthology.org/2020.emnlp-demos.6.

[62] Matthew D. Zeiler ja Rob Fergus. "Visualizing and Understanding Convolutional Networks". Teoksessa: *Computer Vision – ECCV 2014*. Toim. Fleet D., Pajdla T., Schiele B. ja Tuytelaars T. Vol. 8689. Lecture Notes in Computer Science. Springer, Cham, 2014, s. 818–833.

[63] Yu Zhang, Peter Tiňo, Aleš Leonardis ja Ke Tang. "A Survey on Neural Network Interpretability". *IEEE Transactions on Emerging Topics in Computational Intelligence* 5 (2020), s. 726–742.