

Satu Johansson

SUOMEN KIELEN SÄÄNTÖ- JA TEKOÄLY- POHJAINEN TEKSTISTEGANOGRAFIA

Diplomityö
Informaatioteknologian ja viestinnän tiedekunta
Tarkastaja: Taneli Riihonen
Tarkastaja: Okko Räsänen
Joulukuu 2024

TIIVISTELMÄ

Satu Johansson: Suomen kielen sääntö- ja tekoälypohjainen tekstisteganografia
Diplomityö
Tampereen yliopisto
Tietotekniikan DI-ohjelma
Joulukuu 2024

Arkaluonteinen viestintä on usein salattua, mutta paras salainen viestintä on myös piilotettua. Tässä apuun tulee steganografia, jolla viestinnän olemassaolo voidaan piilottaa ja viestintä toteuttaa piilossa näkyvillä. Digitaalinen steganografia piilottaa tietoa bitteihin, kuviin ja tekstin aseteluun tai sanamuotoihin tavalla, joka on merkityksellinen viestinnän osapuolille mutta merkityksetön ulkopuolisille tarkkailijoille. Paras puolustus on viestinnän havaitsemattomuus, mutta toisaalla steganalyysi pyrkii havaitsemaan ja purkamaan steganografista viestintää. Digitalisaation myötä steganografian ja steganalyysin ikuinen kilpajuoksu on nopeutunut ja mukaan on tullut myös uusi arvaamaton osapuoli: tekoäly.

Tekoäly ja huomiota keränneet suuret kielimallit, joiden avulla tekoäly kykenee inhimillistä jäljittelevään tekstintuottoon, tuovat uudenlaisia mahdollisuuksia etenkin tekstisteganografiaan kuuluvan lingvistisen steganografian toteuttamiseen. Tekoällyn mahdollistavan neuroverkon opettaminen on kuitenkin vaativa ja pitkäkestoinen urakka, joka vaatii laskentatehon lisäksi myös kattavaa opetusaineistoa. Suomenkielinen tekoälykeskustelu on jäänyt englannin jalkoihin, ja toisaalta suomen kielioppi kielelle ominaisine taivutuksineen tekee suomen kielen käytöstä tekstiä tai puhetta käsittelevissä teknologioissa haastavaa.

Tutkielmassa laaditaan sääntöpohjainen, suomen kieltä taivuttava sanankorvausohjelma, jossa sanankorvaustaulukon avulla luonnollisesti taivutettuja sanoja voidaan korvata toisella lauseessa. Alkuperäinen lause analysoidaan sanantunnistusohjelmalla, ja saadun kieliopillisen tiedon perusteella ohjelma taivuttaa korvaavan sanan vastaavaan sijaan. Tämä toiminnallisuus on tuttu suomalaisesta taktisesta viestinnästä ja on sen päivitys tietotekniseen toteutukseen, mutta samalla se mahdollistaa tekoällyn tarvitseman opetusaineiston tuottamisen. Lisäksi käsitellään digitaalista steganografiaa, neuroverkkoja, suuria kielimalleja ja transformer-arkkitehtuuria sekä esitetään neljä tapaa opettaa kielimallia toteuttamaan vastaavaa sanankorvausta. Tämän lisäksi testataan kaupallisen, pitkälle opetetun kielimallin onnistumista suomen kielen sanankorvauksessa. Lopussa esitetään jatkotutkimushankkeita ja spekuloidaan tekoällyn suomia uusia mahdollisuuksia sekä steganografiaan että steganalyysiin.

Avainsanat: aktivaatiofunktio, alisovittaminen, ChatGPT, embedding, etuliitesäättö, generatiivinen steganografia, generatiivinen tekoäly, GPT, hallusinointi, hienosäättö, kehotesuunnittelu, kehotesäättö, kielimallin opettaminen, lingvistinen steganografia, luonnollisen kielen käsittely, Microsoft Copilot, neuroverkon opettaminen, neuroverkot, NLP, ohjaamaton oppiminen, ohjattu oppiminen, steganalyysi, steganografia, suuret kielimallit, syväoppiminen, taktinen viestintä, tarkkaavaisuus, tekoäly, tekstisteganografia, tiedon piilottaminen, tilastollinen kielimalli, token, tokenointi, transformer, vahvistusoppiminen, Voikko, ylisovittaminen.

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin Originality Check -ohjelmalla.

ABSTRACT

Satu Johansson: Finnish Language Rule- and AI-Based Text Steganography
Master's Thesis
Tampere University
Master's Programme in Information Technology
December 2024

Sensitive communication is often encrypted, but the best secret communication is also hidden. Here comes in steganography, which allows hiding the existence of communication and hiding communication in plain sight. Digital steganography hides information in bits, images, and text positioning and phrasing in a way that is meaningful to communicating parties but meaningless to outside observers. The best defense is the imperceptibility of communication, but elsewhere, steganalysis seeks to detect and recover steganographic communication. Digitalization has hastened the endless rivalry between steganography and steganalysis and a new, unpredictable party has entered the fray: artificial intelligence.

Artificial intelligence and attention-garnering large language models that enable artificial intelligence to emulate human text generation have provided new opportunities especially for linguistic steganography, a subcategory of text steganography. However, training the neural network that enables artificial intelligence is a demanding and time-consuming task that requires both processing power and extensive training data. Finnish artificial intelligence discussion has been overshadowed by English; likewise, Finnish grammar with its characteristic inflections makes it challenging to use Finnish language in technologies that deal with written or spoken language.

This thesis introduces a rule-based word substitution program that inflects Finnish language and allows replacing natural words with another with the help of a word substitution table. The original sentence is analyzed with a word recognition software, and the grammatical information obtained allows the program to inflect the replacement word accordingly. This functionality is familiar from Finnish tactical communication and is a digital update to it, but at the same time, it enables producing training data needed by artificial intelligence. In addition, digital steganography, neural networks, large language models, and transformer architecture as well as four ways to train a language model to perform corresponding word substitution are discussed. Additionally, the performance of an extensively trained, commercial language model is tested in Finnish word substitution. Further research opportunities are discussed in the end, along with speculations on new opportunities provided by artificial intelligence for both steganography and steganalysis.

Keywords: activation function, artificial intelligence, attention, ChatGPT, deep learning, embedding, fine tuning, generative AI, generative steganography, GPT, hallucination, information hiding, language model training, large language models, linguistic steganography, Microsoft Copilot, natural language processing, neural network training, neural networks, NLP, overfitting, prefix tuning, prompt engineering, prompt tuning, reinforcement learning, statistical language model, steganalysis, steganography, supervised learning, tactical communication, text steganography, token, tokenization, transformer, underfitting, unsupervised learning, Voikko.

The originality of this thesis has been checked using the Turnitin Originality Check service.

TEKOÄLYN KÄYTTÖ OPINNÄYTTEESSÄ

Opinnäytteessäni on käytetty tekoälysovelluksia:

- Ei
- Kyllä

Ilmoitukseni mukaan olen käyttänyt opinnäytteessäni tutkielmaprosessin aikana seuraavia tekoälysovelluksia:

Tekoälysovellusten nimet ja versiot: Microsoft Copilot (GPT-4 Turbo) ja ChatGPT (GPT-4 mini)

Käyttötarkoitus: Diplomityössä on testattu kaupallisen kielimallin Microsoft Copilot soveltumista työssä käsiteltyyn suomenkieliseen peittämiseen ja dokumentoitu testin tulokset. Tekoälyn generoimat vastaukset on annettu tutkielman liitteessä ja testin tulokset on referoitu omin sanoin luvussa 5.6. Lisäksi on lyhyesti käytetty sovellusta ChatGPT, jonka erot kaupalliseen versioon on niin ikään referoitu omin sanoin luvussa 5.6.

Itse työn kirjoittamisessa ei ole käytetty tekoälyä, vaan työssä referoidaan ja dokumentoidaan tekoälyllä tehdyn testin tulokset.

Osiot, joissa tekoälyä on käytetty: luku 5.6 ja liite

Olen tietoinen siitä, että olen täysin vastuussa koko opinnäytteeni sisällöstä, mukaan lukien osat, joissa on hyödynnetty tekoälyä, ja hyväksyn vastuun mahdollisista eettisten ohjeiden rikkomuksista.

ALKUSANAT

Vähän tekoälyä vai paljon tekoälyä, siinä pulma. Vuosikymmeniä kytenyt tutkimus tuntuu yhtäkkiä näyttäytyvän kaikkialla, ja 2020-luku on osoittautunut entisen ja uuden maailman vedenjakajaksi. Tutkimustiedon ja käytännön sovellusten päivittyminen on huimaa.

Suurkiitokset Taneli Riihoselle ja Okko Räsäselle väsymättömästä ohjauksesta ja tarkastustyöstä. Opastuksenne on ollut korvaamatonta ja vaivannäköänne arvostetaan kiitollisuudella.

Diplomityön tekeminen suomeksi saattaa vaikuttaa epätavalliselta valinnalta. Olkoon tämä viimeinen ristiretki suomen kielen puolesta aikana, jolloin englantia hallitsee suomalaisista tieteellistä diskurssia enemmän kuin koskaan.

Tampereella 16.12.2024

Satu Johansson

SISÄLLYSLUETTELO

1. JOHDANTO	1
1.1 Työn motivointi ja rajaukset	1
1.2 Työn tavoitteet.....	3
1.3 Työn tulokset.....	3
1.4 Työn rakenne	5
2. STEGANOGRAFIA JA TIEDON KÄTKEMINEN.....	6
2.1 Steganografian toiminta ja jaottelu	6
2.2 Tekstisteganografia.....	17
2.3 Steganalyysi ja steganografian heikkoudet.....	22
3. PEITTÄMINEN OHJELMALLISESTI	29
3.1 Peittäminen sanankorvauksella	29
3.2 Suomen kielen lyhyt kielioppi ja ominaispiirteitä	30
3.3 Ohjelman toiminta vaiheittain	32
3.4 Ohjelman toteutuksessa tehdyt valinnat	36
3.4.1 Käyttötarkoituksen mukaiset valinnat	36
3.4.2 Ohjelmallisen toteutuksen valinnat.....	37
3.5 Ohjelman toiminta	37
3.6 Ohjelman toiminnan arviointi	40
4. NEUROVERKOT JA TEKOÄLY	43
4.1 Tekoäly ja koneoppiminen.....	43
4.2 Syväoppiminen ja neuroverkot	46
4.2.1 Neuroverkon rakenne	49
4.2.2 Aktivaatiofunktio.....	50
4.3 Neuroverkon opettaminen	52
4.3.1 Vastavirta-algoritmi ja virhefunktio	52
4.3.2 Gradienttimenetelmä ja neuroverkon oppimishopeus.....	55
4.3.3 Mallin yli- ja alisovittaminen.....	55
4.4 Transformer-arkkitehtuuri	57
4.4.1 Transformerin syötteen muodostuminen	58
4.4.2 Transformerin rakenne.....	61
4.4.3 Transformerin sovellusalueet	65
5. PEITTÄMINEN KIELIMALLEILLA	67
5.1 Suuret kielimallit	67
5.2 Suurten kielimallien rakenne	71
5.2.1 Token.....	71
5.2.2 Embedding.....	73
5.3 Suurten kielimallien opettaminen.....	74

5.4	Suuren kielimallin opettaminen omalla aineistolla.....	75
5.4.1	Kehotesuunnittelu	77
5.4.2	Kehotesäätö.....	79
5.4.3	Etuliitesäätö	81
5.4.4	Hienosäätö.....	84
5.5	Suomen kielen tekoälypohjainen tekstisteganografia	85
5.6	Peittäminen Microsoft Copilotin ja kehotesuunnittelun avulla.....	87
6.	JOHTOPÄÄTÖKSET	93
	LÄHTEET	96
	LIITE: PEITTÄMINEN MICROSOFT COPILOTIN AVULLA.....	108

KUVALUETTELO

Kuva 1.	Tiedon kätkemisen ”taikakolmio”. Lähteestä Othman (2020, s. 17).	7
Kuva 2.	Steganografinen prosessi, jossa salaviesti upotetaan olemassa olevaan peiteobjektiin.	8
Kuva 3.	Generatiivinen steganografinen prosessi, jossa salaviestiä varten tuotetaan peiteobjekti.	8
Kuva 4.	Esimerkki LSB-salauksesta kuvasteganografiassa.	13
Kuva 5.	OSI-mallin mukaan jäsenneiltyjä keinoja toteuttaa verkkosteganografiaa. Lähteestä Lubacz et al. (2014, s. 227).	16
Kuva 6.	Peittämis- ja purkamisprosessin vaiheet. Peittäminen kuvattu vasemmalla ja purkaminen oikealla.	33
Kuva 7.	Selkotekstin muuntaminen peitetekstiksi.	34
Kuva 8.	Peitetekstin purkaminen takaisin selkotekstiksi.	35
Kuva 9.	Puretun selkotekstin ja alkuperäisen selkotekstin tarkkuusvertailu.	36
Kuva 10.	Ohjelmaan sisällytettyjen taivutusluokkien kattavuus kaikista suomen kielen sanoista. Kunkin taivutusluokan esiintymien lukumäärä on esitetty pystypalkeilla ja vasemmanpuoleisella asteikolla. Esiintymien kumulatiivinen summa prosenttiosuutena kokonaisuudesta on esitetty oranssilla viivalla ja oikeanpuoleisella asteikolla. Vihreät pylväät ja oranssi pisteiviiva kuvaavat ohjelman kattavuutta ja toteutukseen sisällytettyjä taivutusluokkia.	41
Kuva 11.	Yksinkertaistettu neuroverkko on musta laatikko.	46
Kuva 12.	Neuroverkko, jossa kahden neuronin syötekerros yhdistyy kolmen neuronin piilokerrokseen, joka edelleen yhdistyy neljän neuronin piilokerrokseen ja edelleen ulostulokerrokseen.	47
Kuva 13.	Neuroverkko, jossa kahden neuronin syötekerros yhdistyy neljän neuronin piilokerrokseen, joka edelleen yhdistyy yhden neuronin ulostulokerrokseen. Neuroneilla on lisäksi paino w ja vakiotermit b , jotka vaikuttavat neuronin laskutoimituksiin.	47
Kuva 14.	Eteenpäinkytketty ja täysin yhdistynyt neuroverkko. Tieto kulkee vain yhteen suuntaan ja kunkin kerroksen neuronit kytkeytyvät aina seuraavan kerroksen neuroneihin.	49
Kuva 15.	ReLU-funktion kuvaaja vasemmalla ja vuotava ReLU -funktion kuvaaja oikealla.	50
Kuva 16.	GELU-funktion kuvaaja.	51
Kuva 17.	Alisovitetun mallin kuvaaja vasemmalla ja ylisovitetun mallin kuvaaja oikealla. Alisovitettu malli ei huomioi opetusdataa riittävästi, kun taas ylisovitettu malli on liian yksityiskohtainen, eikä yleisty laajempaan ilmiöön. Ihanteellinen sovitus keskellä kuvaa mittauspisteitä riittävällä tarkkuudella.	57
Kuva 18.	Transformerin syötteen muodostuminen tekstisyötteelle.	61
Kuva 19.	Transformerin rakenne. Enkooderi vasemmalla ja dekooderi oikealla. Suomennettu lähteestä Vaswani et al. (2017, s. 3).	62
Kuva 20.	Transformerin tarkkaavaisuusmekanismin kerrosrakenteita. Skaalattu pistetulotarkkaavaisuus vasemmalla ja monipäinen tarkkaavaisuus oikealla. Suomennettu lähteestä Vaswani et al. (2017, s. 4).	64
Kuva 21.	Suurten kielimallien suhde muihin tekoälykentän osa-alueisiin.	67
Kuva 22.	Luonnollisen kielen käsittelyn osa-alueet.	68
Kuva 23.	Suuren kielimallin opettaminen opetusaineistosta sovellukseksi.	69
Kuva 24.	Englanninkielisen lauseen tokenointi.	72
Kuva 25.	Suomenkielisen lauseen tokenointi.	72

Kuva 26.	<i>Kielimallin säätämisen tekniikat helpoimmasta vaikeimpaan.....</i>	<i>76</i>
Kuva 27.	<i>Etuliitesäädön havainnollistus.....</i>	<i>82</i>

TAULUKKOLUETTELO

<i>Taulukko 1</i>	<i>Esimerkki LSB-salauksen toteutuksesta.....</i>	<i>12</i>
<i>Taulukko 2</i>	<i>Hyökkäystyypit steganografiaa vastaan helpoimmasta vaikeimpaan. Lähteestä Karampidis et al. (2018, s. 217–218).</i>	<i>23</i>
<i>Taulukko 3</i>	<i>Tekstisteganografian eri lajien ominaisuuksia ja kyky vastustaa steganalyysiä. Lähteestä Ahvanooey et al. (2019, s. 15).....</i>	<i>27</i>
<i>Taulukko 4</i>	<i>Sanankorvaustaulukko.....</i>	<i>29</i>
<i>Taulukko 5</i>	<i>Sanankorvaustaulukko, jossa selkosanaa vastaa usea peitesana.....</i>	<i>30</i>

LYHENTEET JA MERKINNÄT

AI	artificial intelligence, tekoäly
API	application programming interface, ohjelmarajapinta
ASCII	American standard code for information interchange, eräs merkistöstandardi
BERT	bidirectional encoder representations from transformers, Googlen kehittämä suuri kielimalli
BNN	Bayesian neural network, bayesilainen neuroverkko
BPE	byte-pair encoding, tokenoinnissa käytetty tavu-parikoodaus
cc	carbon copy, kopio
CNN	convolutional neural network, konvoluutioneuroverkko
CoT	chain of thought, ajatuksenkulkua jäljittelevä tekniikka kielimallin opettamiseen
DCT	discrete cosine transform, diskreetti kosinimuunnos
FFNN, FNN	feedforward neural network, eteenpäinkytketty neuroverkko
FFT	fast Fourier transform, nopea Fourier-muunnos
FSL	few-shot learning, muutaman näytteen oppiminen syväoppimisessa
GELU	Gaussian error linear unit, eräs aktivaatiofunktio
GPT	generative pretrained transformer, OpenAI:n kehittämä suuri kielimalli, generatiivinen esiopetettu transformer
HTML	hypertext markup language, eräs merkintäkieli
HTTP	hypertext transfer protocol, eräs tietoliikenneprotokolla
ID	identification, tunnistus
IP	Internet protocol, eräs tietoliikenneprotokolla
JPEG	Joint Photographic Experts Group, eräs kuvaformaatti
Kotus	Kotimaisten kielten keskus, aiemmin Kotimaisten kielten tutkimuskeskus
LLM	large language model, suuri kielimalli
LM	language model, kielimalli
LSB	least significant bit, binääriluvun vähiten merkitsevä bitti ja sen muokkaamista hyödyntävä tekniikka
LUMI	large unified modern infrastructure, suomalainen supertietokone
MP3	MPEG-1 audio layer 3, äänenpakkausformaatti
MSE	mean squared error, keskineliösumma
NLG	natural language generation, luonnollisen kielen tuottaminen
NLP	natural language processing, luonnollisen kielen käsittely
NLU	natural language understanding, luonnollisen kielen ymmärtäminen
OCR	optical character recognition, tekstintunnistus
OFDM	orthogonal frequency-division multiplexing, eräs modulointitekniikka
OSI	open systems interconnection, tietoliikenneprotokollien kerrosmalli
OSL	one-shot learning, yhden näytteen oppiminen syväoppimisessa
PDF	portable document format, eräs tiedostoformaatti
PLM	pretrained language model, esiopetettu kielimalli
RAG	retrieval-augmented generation, haulla parannettu luominen
ReLU	rectified linear unit, eräs aktivaatiofunktio
RGB	red, green, blue, eräs kuvissa käytetty värimalli, joka perustuu punaiseen, vihreään ja siniseen
RLHF	reinforcement learning from human feedback, vahvistusoppiminen ihmispalautteella
RNN	recurrent neural network, takaisinkytketty neuroverkko
RSA	Rivest-Shamir-Adleman, eräs julkisen avaimen salausmalli
SGD	stochastic gradient descent, stokastinen gradienttimenetelmä
SIP	session initiation protocol, eräs tietoliikenneprotokolla

SOTA	state-of-the-art, johtava tai yleisesti parhaaksi todettu alan käytänne
SSL	self-supervised learning, itseohjautuva oppiminen; semi-supervised learning, osittain ohjattu oppiminen
T5	text-to-text transfer transformer, Googlen kehittämä suuri kielimalli
TCP	transmission control protocol, eräs tietoliikenneprotokolla
ToS	type of service, IP-kehiksen kenttä
VoIP	voice over IP, Internet-puheluissa käytetty teknologia
WAV	waveform audio file format, eräs äänitiedostoformaatti
WLAN	wireless local area network, langaton lähiverkko
ZSL	zero-shot learning, näytteetön oppiminen syväoppimisessa

TEKOÄLYN, SYVÄOPPIMISEN JA STEGANOGRAFIAN SANASTO

activation function	aktivaatiofunktio
agent	kone
algorithm	algoritmi, toteutustapa
alignment	intentio
attention	tarkkaavaisuus
backpropagation	vastavirta-algoritmi
bias	vakiokerroin
capacity	kapasiteetti
chain of thought	ajatuksenkulku
cipher	salain
ciphertext	salateksti
classification	luokittelu
code book	koodikirja
complex, complexity	monimutkainen, monimutkaisuus
convolutional neural network	konvoluutioneuroverkko
cost function	kustannusfunktio
cover, carrier	peite(objekti)
coverless	peitteetön
deep learning	syväoppiminen
deep neural network	syväneuroverkko
downstream task	loppupään tehtävä
embedded	upotettu
embedding	embedding, upote (lopputulos), upottaminen (toteutus)
encoding scheme	koodausmalli
encrypt	salata
error function	virhefunktio
feedforward neural network	eteenpäinkytketty neuroverkko
few-shot learning	muutaman näytteen oppiminen
few-shot prompting	muutaman näytteen kehote
fine tuning	hienosäätö
foundation model	perustamalli
frozen	jäädytetty
fully-connected neural network	täysin yhdistynyt neuroverkko
generated, generative	generoitu, generatiivinen
gradient descent	gradienttimenetelmä
hallucination	hallusinointi
hard prompt	kova kehote
hidden layer	piilokerros
hidden message, embedded data	salaviesti
imperceptibility, undetectability	havaitsemattomuus
input	syöte
input layer	syötekerros
key-value pair	avain-hyötykuorma-pari
labeled	luokiteltu

large language model	suuri kielimalli
learning rate	oppimisnopeus
loss function	tappiofunktio, hukka-funktio
machine learning	koneoppiminen
multi-head self-attention	monipäinen itsetarkkaavaisuus
narrow AI, weak AI	heikko tekoäly
natural language processing	luonnollisen kielen käsittely
neural network	neuroverkko
noisy	kohinainen
null cipher	nollasalaus
one-shot learning	yhden näytteen oppiminen
one-shot prompting	yhden näytteen kehote
output	tuloste
overfitting	ylisovittaminen
parallel attention layer, head	rinnakkainen tarkkaavaisuuskerros, pää
positional encoding	sijainnin koodaus
prefix tuning	etuliitesäätö
pretrained	esiopetettu
prompt,	kehote, kehoitesuunnittelija, kehoitesuunnittelu
prompt engineer,	
prompt engineering	
prompt tuning	kehotesäätö
recurrent neural network	takaisinkytketty neuroverkko
reinforcement learning (from human feedback)	vahvistusoppiminen (ihmispalautteella)
robustness	kestävyys
scaled dot-product attention	skaalattu pistetulo-tarkkaavaisuus
self-attention, intra-attention	itsetarkkaavaisuus
self-supervised learning	itseohjautuva oppiminen
semi-supervised learning	osittain ohjattu oppiminen
soft prompt	pehmeä kehote
steganalysis	steganalyysi
stego key	stegoavain
stego object	stego-objekti
step size	askelpituus
strong AI	vahva tekoäly
superintelligence	supertekoäly
supervised learning	ohjattu oppiminen
text generation	tekstintuotto, tekstin generoiminen
token, tokenize, tokenizer	token, tokenoida, tokenoija
transfer learning	siirto-oppiminen
transformation	muunnos
transformer	transformer, muunnin
underfitting	alisovittaminen

universal detection, blind detection	universaali tai sokea tunnistus
unsupervised learning	ohjaamaton oppiminen
weight	painokerroin
weighted sum	painotettu summa
zero-shot learning	näytteetön oppiminen
zero-shot prompting, direct prompting	näytteetön kehote

1. JOHDANTO

Tekoäly on arkipäiväistynyt ja vaikuttaa olevan läsnä kaikkialla. Samalla siitä tuntuu tulleen jonkinlainen kattotermi kaikelle tietotekniselle, jota tehdään ilman merkittävää ihmisen työpanosta. Tämä ei luonnollisesti ole aivan totta, mutta jonkinlainen kiinnostuksen aalto ja pieniä, arkisia sovelluksia hakukoneen avuliaista vastauksista uutistiivistelmiin ilmestyy päivittäin. Samalla ovat yleistyneet huomiota herättäneet suuret kielimallit, joiden avulla tekoäly kykenee luonnollisen kielen käsittelyyn ja inhimilliseltä vaikuttavaan tekstintuottoon.

Erillisenä mutta ei kuitenkaan irrallisena osa-alueena viestintä on voimakkaassa murroksessa. Viestintää leimaa voimakas digitaalistuminen ja jako avoimissa kanavissa. Tämä pätee myös arkaluonteiseen ja ulkopuolisilta piilotettavaan tietoon, jota ei enää suojata säilyttämällä ja jakamalla tietoa fyysisessä formaatissa, johon pääsy on tarkkaan rajattu. Sen sijaan luottamuksellista tietoa jaetaan eri tavoin kryptografialla salattuna suojaamattomissa kanavissa. Tähän tilanteeseen lisäsuojauksen tuo kryptologiaan linkittyvä tieteenala steganografia, joka kätkee viestinnän ulkopuolisten tarkkailulta.

1.1 Työn motivointi ja rajaukset

Työ käsittelee toisaalta steganografiaa ja tiedon kätkemistä, toisaalta tekoälyä ja sen huomionarvoista saavutusta suurilla kielimalleilla: valtavia tilastollisia, supertietokoneiden laskemia, matriiseista ja vektoreista koostuvia malleja, joiden avulla tietokone kykenee tuottamaan ihmisen tasoista tekstiä, vastaamaan kysymyksiin ja tuottamaan uskottavia konekäännöksiä. Kielimallien avulla luonnollisen kielen käsittely on kokenut todellisen heräämisen ja nopeasti yleistynyt arkiseksi osaksi verkon palveluiden ja hakukoneiden käyttöä, yritysten toimintaa ja erilaisia helppokäyttötoimintoja. Samoin kielimallien käyttö herättää keskustelua omistajuudesta, kun generatiivista tekoälyä käytetään tekstintuottoon viestinnässä ja opiskelussa. Aihealue on kipinöivän ajankohtainen, mutta asiallisen tiedon saaminen on silti työlästä.

Toisaalla taas on steganografia, tiedon piilottamisen opinala. Steganografia pyrkii estämään salaisen tiedon paljastumista kätkemällä viestinnän olemassaolon kokonaan naamioimalla viestin joksikin yleiseksi ja harmittomaksi asiaksi, joka ei herätä ulkopuolisen tarkkailijan mielenkiintoa. Kun kryptografia salaa viestin sisällön, steganografia salaa

viestin olemassaolon. Steganografiaa ovat näkymättömät musteet, kuviin piilotetut kuvat ja tekstiin piilotetut salaviestit. Sitä ovat tietoliikenteen manipuloiminen ja redundanttien rakenteiden valjastaminen välittämään tietoa, joka sulautuu massaan. Steganografian sovellusalueet ovat loputtomia ja maailman mukana kehittyviä. Digitaaliseen viestintään kohdistuva tarkkailu on elvyttänyt kiinnostusta myös steganografiaa kohtaan, ja samalla neuroverkkojen ja tekoälyn kehitys on avannut uusia mahdollisuuksia sekä toteuttaa steganografiaa että havainnoida ja purkaa sillä toteutettua viestintää.

Samaan aikaan kun tekoäly yleistyy hieman kaikkialla, tulee myös kiinnostusta hyödyntää uuden teknologian mahdollisuuksia myös aiemmin puhtaasti ei-tietoteknisiin asioihin. Tässä mukaan tulee peittäminen, sanankorvaus, jolla etenkin suomalaisessa sotilaallisessa radioviestinnässä naamioidaan kriittistä tietoa ulkopuolisilta tarkkailijoilta. Peittämisessä taktisesti tärkeät sanat korvataan jollain toisella. Esimerkiksi viesti ”ammukset ovat lopussa ja evakuointi mahdotonta” voitaisiin ilmaista ”kalat ovat verkossa ja rapusaalis runsasta”. Nykyisin käytössä olevassa viestinnässä peittäminen toimii edelleen paperitulosteina jaetuilla sanankorvaustaulukoilla, eikä tämän toiminnan korvaamiseen ole ollut vaihtoehtoja.

Tästä tulee tarve suomen kieltä oikeakielisesti taivuttavan ohjelman laatimiselle, mistä syntyy tutkielman pääasiallinen käytännön osuus ja laadittu sanankorvausohjelma, joka hyödyntää sille annettuja sanastoja. Samalla tehdään tärkeää pohjatyötä tekoälyn hyödyntämistä varten, sillä tekoälyn opettaminen vaatii laajaa opetusaineistoa, jota ei ole olemassa valmiiksi. Sanankorvaustoiminnallisuuden avulla sellaista voidaan kuitenkin laatia, ja tarkkuutta vertailevan toiminnallisuuden avulla ohjelman toimintaa arvioida ja parantaa entisestään. Valmiin käyttäjäsovelluksen toteuttaminen on rajattu tutkielman ulkopuolelle, mutta arkkitehtuuri ja tarvittava toiminnallisuus on toteutettu.

Eräänä motivaationa, joka ansaitsee ehdottomasti tulla mainituksi ja kaikin puolin korostetuksi, on halu koota yhteen kunnollista suomenkielistä tutkimustietoa asiasta, josta ei suomeksi kirjoiteta päivälehtiä ja yksittäisiä blogeja lukuun ottamatta. Tekoälystä ja steganografiastakin löytyy englanniksi runsaudensarvi nimellisiä lähteitä, mutta uskottavan tieteellisen tutkimustiedon erottelu yksityisyritteliästä harrastuskirjoittelusta ei ole triviaali haaste. Suomen kielen sanastotyö ei ole työn itsetarkoitus, mutta se on ehdottomasti tärkein työstä saatava sivuvirta.

Työn ulkopuolelle jää monia kiinnostavia aihealueita, jotta käsittely pysyy järkevässä laajuudessa. Työssä ei käsitellä peitesanastojen laatimista eikä steganografian psykologiaa, kuten keinoja tehdä viestinnästä ihmistarkkailijalle huomaamatonta. Lukuisia tekoälyn mahdollistavia neuroverkkorakenteita ominaispiirteineen ei ole mahdollista käsitellä

kattavassa laajuudessa, vaan niihin on halutessaan tutustuttava muissa yhteyksissä. Kielimallien kehityksen historia ja monia tilastollisten kielimallien edistysaskeleita ja aiempia malleja on sivuutettu, vaikka ne sinällään ovat kiinnostavia ja niissä tehty tutkimus on osaltaan mahdollistanut tämänhetkisten suurten kielimallien uskottavan kyvyn jäljitellä ihmisen ilmaisuja. Rajaukset ovat kuitenkin olleet tarkoituksenmukaisia, tarkoituksellisia ja tarpeellisia, ja kaiken lyhyt historia on jätettävä tietokirjallisuuden huoleksi.

1.2 Työn tavoitteet

Työn tavoitteena on löytää tietotekninen toteutus aiemmin puhtaasti ihmisen varassa olleelle sanankorvauspeittämiselle, jossa arkaluonteisina pidetyt sanat korvataan avoimessa tai tarkkaillussa kanavassa viestiessä harmittoman oloisilla koodisanoilla. Suomen kieltä ja taivutuksia analyysoivia ohjelmistototeutuksia on saatavana useita, mutta ei ohjelmistoja, jotka taivuttaisivat suomea käyttäjän pyytämällä tavalla. Oikolukuohjelmat ja konekäännökset ovat asia erikseen, eivätkä ne korvaa tätä puutetta. Suomen kielen ominaispiirteenä sanat taipuvat kymmenillä tavoilla useiden tekijöiden samanaikaisesta vaikutuksesta, minkä kielen käyttäjä kyllä ymmärtää intuitiivisesti. Näiden tapojen dokumentointi on lähinnä suuntaa antavaa jopa kielen rakennetta käsittelevässä kirjallisuudessa, ja kielioopin sääntöjen sovellettavuus sääntöpohjaiseen toteutukseen loppujen lopuksi heikkoa.

Sen lisäksi työn tavoitteena on selvittää, millä tavoin peittämistä voidaan toteuttaa tekoälyn ja suomalaisten kielimallien avulla. Tämä tarkastelu pitäytyy teoreettisena lukuun ottamatta lyhyttä kokeellista osuutta, jolla pyritään vahvistamaan tai kumoamaan keveimmän menetelmän sovellettavuus peittämiseen.

Läpi työn tavoitteena on kattavan katsauksen luominen digitaalisen steganografian valituista osa-alueista, tekoälystä ja kielimalleista suomen kielellä, joista aiempi julkaistu tutkimustieto ja tavoitteellinen kielenhuolto on pirstaloitunutta, paikoin olematonta. Lisätiedon etsimisen helpottamiseksi ja käsittelyn pitämiseksi yksiselitteisenä termit annetaan myös englannin kielellä.

1.3 Työn tulokset

Käytännön osuutena esitellään ohjelmallisesti toteutettu ratkaisu, jossa luonnollista suomenkielistä tekstiä voidaan peittää sanankorvaustekniikalla ja peiteteksti purkaa takaisin alkuperäiseksi selkotekstiksi. Sanankorvaus on tyyppillistä sotilaskäytössä, jossa taktisesti tärkeää viestintää peitetään vaihtelevien sanankorvaustaulukoiden avulla. Myös arkisessa verkkoviestinnässä ilmiö on tuttu: keskustelupalstojen estettyjä tai kiellettyjä

sanoja kierretään ilmaisuilla, jotka ihmislukija tunnistaa tai päättelee helposti mutta koneellinen tarkastus ei havaitse. Tällainen sanankorvaus on ihmiselle hyvin luontevaa ja helppoa, mutta toiminnallisuuden toteuttaminen ohjelmallisesti on suomen kielen erityispiirteiden vuoksi haastavaa. Lisäksi toteutetaan alkuperäisen salatekstin ja puretun peiteviestin vertailu, jonka avulla toiminnallisuuden toteutusta voidaan arvioida nyt ja mahdollisen jatkokehityksen aikana.

Teoreettisena osuutena esitellään konsepti, jolla peittäminen olisi mahdollista toteuttaa opettamalla suurta kielimallia omalla aineistolla. Tässä yhteydessä tunnistetaan kolme säätötekniikkaa, joilla suurta kielimallia voidaan opettaa halutun toiminnallisuuden toteuttamiseen. Näiden lisäksi esitellään tekniikka laatia kehoitteita tavalla, jolla kielimalli todennäköisimmin antaa haluttuja tuloksia. Kunkin tekniikan soveltuvuutta suomen kielen sanankorvauspeittämiseen arvioidaan. Kielimallin säätötekniikoiden yhteydessä esitellään toteutetun sääntöpohjaisen peittämisohjelman käyttö opetusmateriaalin tuottoon, sillä kielimallin opettaminen vaatii laajaa opetusaineistoa, jota ei ole muutoin saatavilla. Lisäksi raportoidaan tulokset testauksesta, jossa peittämistä ja purkamista toteutettiin valmiilla kaupallisella ohjelmistolla kehoitteiden avulla.

Todetaan myös, että kielimallien hyödyntämistä hankaloittaa yleisesti vähäinen panostus suomenkieliseen käyttöön: siinä missä malleja on esiovetetty englanniksi jopa monikymmenkertaisella aineistolla ja tämän jälkeen mallin ulostuloa muokattu käyttäjän odotusten mukaiseksi, suomenkielistä aineistoa mallin opettamiseen on käytettävissä suhteellisen vähän ja valtavaa työpanostusta vaatinut työvaihe, joka mallista tekee näennäisen inhimillisen, on sivuutettu. Näin ollen myös mahdollisuudet kielimallien jatkokäyttöön lähtevät takamatkalta.

Tässä tutkielmassa käsittely keskittyy pitkälti GPT-kielimalliperheeseen, jonka teki tunnetuksi chattibotti ChatGPT, ja sen pohjalta suomalaisen tutkimusryhmän laatimaan kielimalliin FinGPT:hen. Tekoälyn sykli on nopeaa: yksin tutkielman kirjoittamisen aikana on julkaistu lisää erityisesti suomen kielialueelle suunnattuja kielimalleja, kuten Poro ja Viking. Samalla GPT-malleja on päivitetty entistä monipuolisempiin versioihin. Eri mallien tarkka vertailu on vaikeaa, sillä suomalainen kielimalli on tehty nimenomaisesti avoimeen käyttöön osin yliopiston tutkimuksena. Vertailukohtana käytetyn kaupallisen esikuvan tarkat tekniset tiedot ja prosessin kuvaus puolestaan ovat liikesalaisuuksia – herkeämättömästä julkisesta mielenkiinnosta ja satunnaisesti tihkuneesta tiedosta huolimatta.

Työssä on myös osaltaan tehty tekoälyn ja syväoppimisen kielenhuoltoa ja sanastotyötä. Tällainen työ olisi tärkeää, mutta ammatillinen sanastotyö tuntuu jääneen vähälle

huomiolle englanninkielisen tieteellisen diskurssin ja yleiskielisen tekoälydebatin hallitessa. Sanastotyötä on tutkielmassa tehty sivujuonteena keräämällä suomenkielistä verkkomateriaalia laajalti ja ottamalla parhaaksi arvioituja termejä käyttöön. Tehty työ ei ole normatiivista, eikä sellaiseen ole annettu mandaattia. Toivottavasti tutkielman ote kuitenkin osoittaisi, että tekoälystä ja syväoppimisesta on mahdollista keskustella myös suomeksi.

Merkittävästä panostuksesta huolimatta työ jätti paljon kysymyksiä ja mahdollisuuksia lisätutkimukselle. Tämän vuoksi esitetään kattavasti pohdintoja jatkotutkimuksesta ja suomenkielisen peittämisen modernisoinnista edelleen suuria kielimalleja hyödyntämällä.

1.4 Työn rakenne

Tutkielman rakenne jakautuu seuraavasti.

Luvussa 2 esitellään digitaalisen steganografian eri sovellusalueita ja toisaalta steganalyysiä, jolla steganografista viestintää pyritään havaitsemaan ja häiritsemään. Erityistä huomiota kiinnitetään tekstisteganografiaan ja tekstisteganalyysiin perinteisten tekniikoiden lisäksi myös neuroverkkojen avulla toteutettuna.

Luvussa 3 käsitellään suomen kielen erityispiirteitä, jotka tekevät luonnollista suomen kieltä käsittelevän ohjelman toteuttamisesta erityisen haastavaa, ja esitellään laadittu toiminnallisuus suomen kielen käsittelyn toteuttamiseen. Lisäksi esitellään suomalaisessa maanpuolustuskontekstissa tärkeä sanankorvauspeittäminen, jota sääntöpohjaisella ohjelmatoteutuksella on mahdollista toteuttaa koneellisesti ja automatisoidusti ilman käyttäjältä tavallisesti vaadittavaa sanataiturointia.

Luvussa 4 tutustutaan koneoppimisen kautta syväoppimiseen ja neuroverkkojen mahdollistamaan tekoälyyn. Lisäksi esitellään luonnollisen kielen käsittelyn mullistanut transformer-arkkitehtuuri.

Luvussa 5 tutustutaan lähemmin transformer-arkkitehtuurin pohjalle rakentuneisiin suuriin kielimalleihin ja niiden hyödyntämiseen kevyemmillä tekniikoilla kuin kielimallin tavanomainen, pitkäkestoinen ja vaativa opettaminen. Lisäksi spekuloidaan säätötekniikoiden soveltuvuutta peittämiseen ja testataan helpoimman tekniikan kehotesuunnittelun soveltuvuutta itse. Tässä yhteydessä esitetään kaupallisella kielimallilla tehdyn peittämistestin tulokset.

Luvussa 6 kootaan tutkielman keskeiset havainnot ja esitetään näkemyksiä jatkotutkimuksesta.

2. STEGANOGRAFIA JA TIEDON KÄTKEMINEN

Tässä luvussa tutustutaan steganografian toimintaperiaatteisiin ja erilaisiin digitaalisen steganografian lajeihin. Lähemmin käsitellään tekstisteganografiaa sekä steganografisen viestinnän havaitsemiseen ja purkamiseen käytettyä steganalyysiä.

2.1 Steganografian toiminta ja jaottelu

Steganografia on tiedon piilottamista, kätkemistä. Toisin kuin kryptografia, joka keskittyy viestin sisällön salaamiseen, steganografia pyrkii salaamaan viestinnän olemassaolon ylipäättään. Usein steganografinen viestintä kuitenkin hyödyntää kryptografista salausta. Toisin päin viestin kätkemisestä on suurta hyötyä tilanteissa, joissa jo salauksen käyttö herättää epäilyjä. (Karampidis et al. 2018, s. 217; Othman 2020, s. 1, 16) Steganografiassa vaikkapa kuvaan tai tekstiin kätetään jotain muuta, kuten toinen kuva, kirjoitettu salaviesti tai bittejä. Vastapuolella **steganalyysi** pyrkii havaitsemaan ja purkamaan steganografista viestintää. Steganalyysiä voivat toteuttaa kone- ja ihmistarkkailija muun muassa tilastollisella analyysillä ja havainnoimalla. (Karampidis et al. 2018, s. 218)

Steganografia on yllättävän arkipäiväistä, vaikka siihen käsitteenä törmää harvoin: sitä hyödynnetään setelirahan autentikoinnissa, tekijänoikeuksia suojaavassa näkymättömässä digitaalisessa vesileimauksessa ja luonnollisesti kaikenlaisen salattavan tiedon kätkemisessä siten, että tietoa tarkasteleva ulkopuolinen ei havaitse eikä edes epäile piiloviestin olevan olemassakaan. Myös näkymättömät musteet (*invisible ink*) ovat tunnettu esimerkki steganografiasta. Steganografian idea on siten olla piilossa näkyvillä (*hiding in plain sight*). (Karampidis et al. 2018, s. 217; Othman 2020, s. 1–2, 8–9, 15; Euroopan keskuspankki 2024)

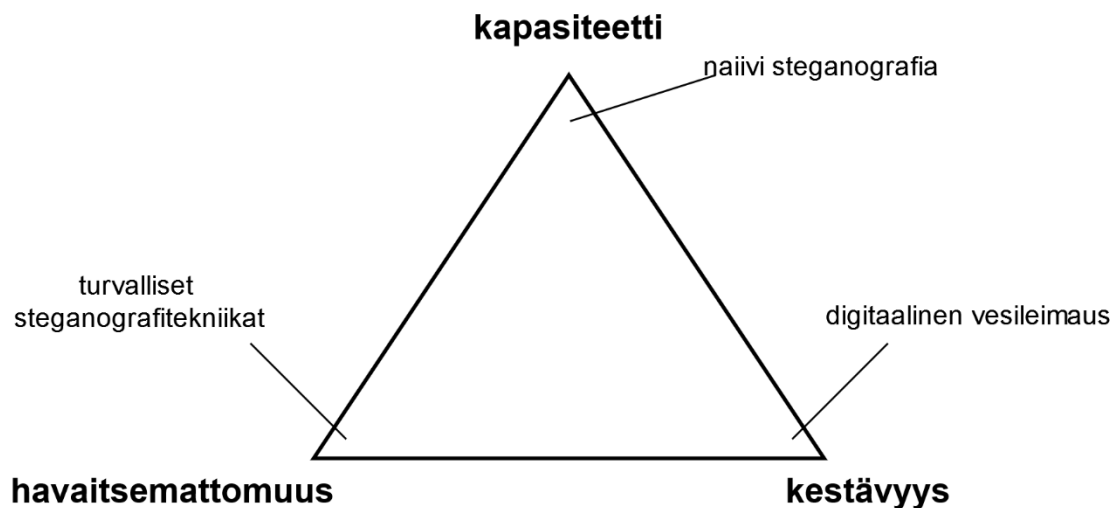
Steganografiaa voidaan havainnollistaa niin sanotulla ”taikakolmiolla” (*“magic triangle”*) (kuva 1), jossa kuvataan tiedon kätkemiseen käytetyn tekniikan kolme kriteeriä (Othman 2020, s. 17):

1. **Kapasiteetti** (*capacity*), toisin sanoen tekniikan kyky kätkeä tietoa suhteessa alkuperäisen viestin (kuten teksti, ääni, kuva) kokoon. Esimerkki suuren kapasiteetin tekniikasta on naiivi steganografia (*naïve steganography*), jolla viitataan mahdollisimman yksinkertaisesti toteutettuun salaukseen, kuten nollasalauus (*null cipher*). Nollasalauksessa piiloviesti yksinkertaisesti muodostuu vaikkapa viestin kunkin sanan ensimmäisestä kirjaimesta (Othman 2020, s. 2)
2. **Havaitsemattomuus** (*undetectability* tai *imperceptibility*), toisin sanoen havaitsemisen vaikeus: kuinka vaikeaa tai helppoa ihmis- tai konetarkkailijan on huomata, että viestiin on kätkeyty piiloviesti. Esimerkki korkeasta havaitsemattomuudesta ovat turvalliset steganografiatekniikat, toisin sanoen tekniikat, joita ei ole

mahdollista havaita ihmisen tarkastelulla tai tilastollisella analyysillä (Hamid et al. 2012, s. 180)

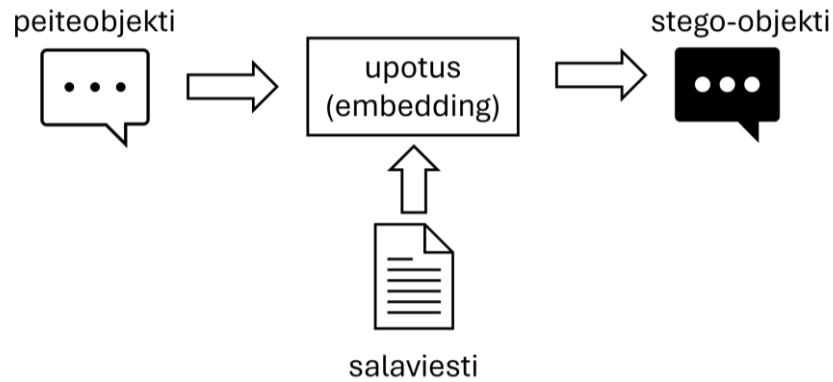
3. **Kestävyys** (*robustness*), toisin sanoen kyky säilyttää kätkeyty viesti ennallaan huolimatta kuvan tai äänen häviöllisestä kompressoinnista, kopioinnista, suodattamisesta tai muusta digitaalisen median tyypillisestä toimenpiteestä. Esimerkki korkeasta kestävyydestä on digitaalinen vesileimaus

Näiden lisäksi tulee huomioida tiedon kätkeymisen ja purkamisen **monimutkaisuus** (*complexity*), joka vaikuttaa paitsi tekniikan helppokäyttöisyyteen myös käyttökustannuksiin, sekä tekniikan **turvallisuus** (*security*). Käytetty steganografinen tekniikka on aina kompromissi kapasiteetin, havaitsemattomuuden ja kestävyuden välillä. (Othman 2020, s. 17) Osana harkintaa on myös steganografian resistenssi steganalyysiä vastaan. Erittäin korkeaan havaitsemattomuuteen ja kestävyteen panostava steganografiatekniikka todennäköisesti vastustaa steganalyysiä onnistuneesti (viestiä ei voi ulkopuolinen tarkkailija havaita, purkaa tai korruptoida), mutta vain näihin kahteen osa-alueeseen panostavan tekniikan kapasiteetti on luultavasti niin rajoittunut, että käytännön sovelluksissa sen käyttökelpoisuus jää vähäiseksi. Taikakolmio on esitetty kuvassa 1.



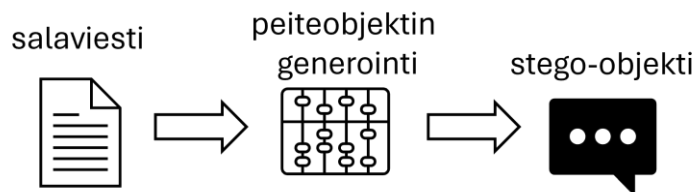
Kuva 1. Tiedon kätkeymisen "taikakolmio". Lähteestä Othman (2020, s. 17).

Steganografinen prosessi eli piiloviestin tuottaminen steganografian avulla voidaan esittää yksinkertaisesti näin: **peiteobjektiin** (*cover* tai *carrier*) upotetaan **salaviesti** (*hidden message* tai *embedded data*) joltain steganografista **algoritmia** eli keinoa käyttäen, jolloin saadaan piiloviestin sisältävä **stego-objekti** (*stego object*). Steganografinen prosessi on esitetty kuvassa 2.



Kuva 2. Steganografinen prosessi, jossa salaviesti upotetaan olemassa olevaan peiteobjektiin.

Nouseva steganografian laji on **generatiivinen steganografia**, jossa salaviestin sisältävä stego-objekti generoidaan eli tuotetaan sen sijaan, että salaviesti upotettaisiin olemassa olevaan peiteobjektiin. Generatiivista tekoälyä on lyhyessä ajassa käytetty tuottamaan kaikenlaista uutta sisältöä tekstistä kuvaan ja äänestä videoon. (Wei et al. 2022a, s. 1). On luultavaa, että generatiivisen tekoälyn edelleen yleistyessä myös kaikenlainen generatiivinen steganografia yleistyy. Generatiivinen steganografinen prosessi on esitetty kuvassa 3.



Kuva 3. Generatiivinen steganografinen prosessi, jossa salaviestiä varten tuotetaan peiteobjekti.

Peiteobjekti voi olla tekstiä, kuvaa, ääntä tai se voi olla mikä vaan alusta, johon on mahdollista upottaa (*embed*) informaatiota. Hyvä peite paitsi huomioi taikakolmion kriteerit mahdollisimman hyvin myös on yleinen, eikä piiloviestin upottamisesta aiheutuva pieni laadun alenema herätä huomiota. Näin vaalitaan steganografisen viestin yleistä ajatusta olla huomiota herättämätön ja piilossa näkyvillä. (Zielińska et al. 2013, s. 3–4, 11)

Steganografinen prosessi voi myös sisältää salaviestin koodauksen kryptografian ja salaimen (*cipher*) avulla salatekstiksi (*ciphertext*) sekä salauksessa stegoavaimen (*stego key*) käytön, jolla varmistetaan, että purettu ja peitteestä irrotettu salaviesti ei kuitenkaan ole ulkopuolisen luettavissa (Agarwal 2013, s. 1). Mahdollinen stegoavain voi olla yksityinen tai julkinen (*private key*, *public key*), ja salaus siten symmetristä tai epäsymmetristä eli julkista (Anderson & Petitcolas 1998; Sharp 2001). Tällainen julkista salausta hyödyntävä steganografia on varsin lähellä kryptografiaa, sillä Kerchoffsin periaatteen

mukaisesti vain avain on salaista tietoa: ei käytetty steganografinen algoritmi, ei peite eikä edes viestinnän olemassaolo (Ke et al. 2019).

Peitteiden yleisyys ei ole muuttumatonta. Tekstiviesti olisi ollut huomiota herättävä ja harvinainen viestintämuoto vuonna 1993, mutta kymmenen vuotta myöhemmin se oli jo täysin arkinen: Suomessa lähetettiin 2003 keskimäärin 31 tekstiviestiä liittymää kohti (Liikenne- ja viestintäministeriö 2004). Vuonna 2010 otettiin maailmanlaajuisesti noin 0,35 biljoonaa (10^{12}) kuvaa päivässä, joista 40 % matkapuhelimella, ja vuonna 2024 kuvia otettiin 1,94 biljoonaa, joista 96 % matkapuhelimella. (Heyman 2015; Broz 2024) Samaan aikaan kuvien ottaminen ja jakaminen on suorastaan muuttunut arkipäiväisestä tyhjämpäiväiseksi samalla, kun kuluttajalaitteiden kuvanlaatu on parantunut. Näkemykset peitteen yleisyydestä ja vaivihkaisuudesta ovat siis monin tavoin sidoksissa aikaan.

Siinä missä matkapuhelinkuvalta kesti kymmenen vuotta arkipäiväistyä ja toiset kymmenen syrjäyttää kilpailijat, generatiivinen tekoäly toteutti vastaavan harppauksen kuukausissa: vain kymmenen kuukauden aikana vuosien 2022–2023 aikana generatiivisella tekoälyllä tuotettiin arviolta 15 miljardia kuvaa, ja toukokuussa 2023 tekoälyn arvioitiin tuotavan yli 34 biljoonaa kuvaa päivässä (Valyaeva 2023). Jos kuvien tekoälyboomin katsotaan alkaneen DALL-E 2 -kuvageneraattorimallin julkaisusta vuonna 2022 (ibid.) ja kehitys on noin nopeaa, tämänhetkistä julkaisutahtia tai tekoälykuvien kokonaismäärän suuruusluokkaa on mahdotonta arvioida.

Vastaava generatiivisen tekoälyn kehitystahti toteutuu muussakin digitaalisessa mediassa: nopeasti yleistyvät syvävääreännökset (*deepfake*) videolla, äänellä ja kuvalla eivät välttämättä ole edes vaikuttamisen väline, vaikka ne ovatkin profiloituneet sellaiseksi (Chauhan 2024, s. 141–142), vaan ne ovat YouTubessa lukuisten fanivideoiden toteutustapa. Mihin tämä kehitys johtaakaan, steganografiaan meneillään oleva murros linkittyy ainakin kahdella merkittävällä tavalla:

- Generatiivinen steganografia ja sen myötä steganalyysi kehittyy generatiivisen tekoälyn rinnalla. Kehitys mahdollistaa entistä monipuolisempien stego-objektien tuottamisen.
- Valtava tekoälypohjaisen materiaalin saatavuus verkossa tarjoaa tilaisuuksia myös stego-objekteille sulautua huomaamatta massaan ja olla piilossa näkyvillä. Generatiivisen steganografian ohella myös upottamista hyödyntävä steganografia, joka käyttää tekoälyn luomaa kuvaa, täyttää helposti vaatimukset peitteen yleisyydestä (Zielińska et al. 2013, s. 3).

Tekoälyyn kohdistuvat sääntelytoimet, kuten tekoälyn luomaa puheääntä käyttävien automatoitujen puheluiden (*robocall*) kieltö Yhdysvalloissa (Chauhan 2024, s. 142) ja

Euroopan Unionin tuleva tekoälyasetus (Hallamaa 2023a), voivat vaikuttaa epäsuorasti myös steganografian toteuttamismahdollisuuksiin. Samoin kryptografiaan kohdistuva sääntely linkittyy steganografiaan, vaikka steganografia sinänsä vaikuttaa toistaiseksi säästyneen nimenomaisesti siihen kohdistuneilta lainsäädännöllisiltä rajoituksilta (Warkentin et al. 2008, s. 30; Wood 2023). Tämä toki ei ole uutta: steganografia on aina hakenut paikkaansa muutosten keskellä muun yhteiskunnan hakiessa tasapainoaan.

On myös olemassa **peitteetöntä steganografiaa** (*coverless steganography*) tai upotuksetonta steganografiaa (*steganography without embedding*). Peitteettömässä steganografiassa peiteobjekti (*cover* tai *carrier*) säilyy muuttumattomana, ja salaviesti välitetään sen sijaan kuvaamalla (*mapping*) salaviesti osissa useiden peiteobjektien ja hajautustaulujen (*hash tables*) avulla (Meng et al. 2024, s. 1–2). Erityispiirteenä peitteetön steganografia voi toisinaan olla erityisen vastustuskykyinen passiiviselle steganalyysille eli steganografian käytön havaitsemiselle (ibid., s. 20). Peitteetöntä generatiivista kuvasteganografiaa yhdistettynä mainittuun Kerchoffsin periaatteeseen ovat tutkineet esimerkiksi Ke et al. (2019).

Digitaalinen steganografia – toisin sanoen digitaalisessa muodossa viestittävän tiedon kätkeminen – voidaan jakaa viiteen pääasialliseen osa-alueeseen (Kaur & Rani 2016, s. 2; Tidmarsh 2023):

1. **Tekstisteganografia** (*text steganography*)
2. **Kuvasteganografia** (*image steganography*)
3. **Audiosteganografia** (*audio steganography*)
4. **Videosteganografia** (*video steganography*)
5. **Verkkosteganografia** (*network steganography*)

Eri steganografian lajeja voidaan myös yhdistellä multimediateganografiaksi (*multimedia steganography*). Luonnollisesti vaikkapa tekstisteganografia ei suinkaan rajoitu digitaaliseen toimintaympäristöön, vaan salattua viestintää esiintyy ja se syntyi bittimaailman ulottumattomissa. Esitellään seuraavassa näitä steganografian lajeja.

Tekstisteganografia

Tekstisteganografiassa tekstiin piilotetaan salaista tietoa. Tyypillisiä tapoja tiedon piilottamiseen ovat tekstin muotoilu, sanojen korvaus tai viestin piilottaminen toisen viestin sisään, kuten oikean viestin koostuminen tekstin sanojen ensimmäisistä kirjaimista (Agarwal 2013, s. 2; Othman 2022, s. 2–6). Tutkielmalle keskeistä tekstisteganografiaa käsitellään lähemmin omassa luvussaan 2.2.

Kuvasteganografia

Kuvasteganografiassa kuvaan piilotetaan salaista tietoa, kuten toinen kuva tai tekstiä. Kuvasteganografian onnistuminen perustuu pitkälti siihen, että silmä ei tyypillisesti havaitse vähäisiä vivahde-eroja väreissä, ja harmaasävyillä (*grayscale*) voidaan edelleen häivyttää sävyeroja ihmiskatsojan ulottumattomiin. Vähäisetkin vivahde-erot ovat kuitenkin edelleen tietokoneen tulkittavissa, ja salattu tieto siten purettavissa. Tämä tarkoittaa sitä, että ihmisen merkityksettömänä ohittama kuva on edelleen koneellisen tarkastuksen havaittavissa. (Othman 2020, s. 4, 9)

Kuvasteganografiaa voidaan toteuttaa eri tavoin (Othman 2020, s. 10, 13–15):

- **LSB** (*least significant bit*), binääriluvun vähiten merkitsevän bitin käyttäminen tiedon kätkemiseen
- **Algoritmit ja muunnokset** (*transformations*), kuten diskreetin kosinimuunnoksen (DCT, *discrete cosine transformation*), nopean Fourier-muunnoksen (FFT, *fast Fourier transform*) tai aallokemuunnoksen (*wavelet transform*) käyttö
- **Peittäminen ja suodattaminen** (*masking and filtering*). Näitä käytetään 24-bittiin kuviin ja harmaasävykuviin piilotettuihin vesileimoihin, joilla suojellaan tekijänoikeuksia ja merkitään teosten lisätietoja

Näistä tekniikoista erityisen kiinnostava on LSB, sillä vähiten merkitsevään bittiin perustuvaa salausta hyödynnetään laajalti myös muissa steganografian lajeissa. Vaikka käsittely on kuvien kontekstista, esitetty periaate pätee yleisesti binäärimaailmaan – toisin sanoen kaikkeen digitaaliseen ja tietokoneilla tapahtuvaan toimintaan.

Digitaalisissa kuvissa käytetty värisyvyys (*color depth*) on tyypillisesti 8-, 16- tai 24-bittinen, tosin värisyvyys ei rajoitu näihin vaihtoehtoihin. Värisyvyys tai bittisyvyys (*bit depth*) kuvaa värien määrää ja siten tarkkuutta, joilla yksittäinen pikseli esitetään. Tällainen 8-bittinen kuva voi koostua 256 värisävystä (2^8) ja 24-bittinen puolestaan 16,7 miljonnasta värisävystä (2^{24}).

Havainnollisuuden vuoksi ajatellaan vanhaa 8-bittistä JPEG-kuvaa (.jpg), jonka 256 värisävyä esitetään RGB-värimallilla punaisen (**R**ed), vihreän (**G**reen) ja sinisen (**B**lue) yhdistelminä. Puhdas punainen saa siten arvon (255,0,0), puhdas vihreä (0,255,0), puhdas sininen (0,0,255), musta (0,0,0) ja valkoinen (255,255,255). Muut värisävyt esitetään erilaisina yhdistelminä kutakin komponenttia. Värisävy voidaan edelleen esittää desimaalilukujen sijaan binäärilukuina tai heksadesimaalilukuina. Esimerkiksi puhdas punainen pikseli voidaan ilmaista




- desimaalilukuna (255,0,0)

- binäärilukuna (11111111,00000000,00000000)
- heksadesimaalilukuna (#FF0000)

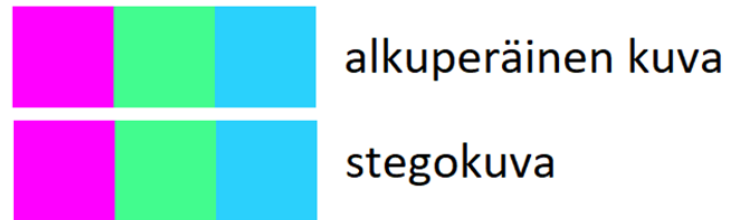
Binääriluvun oikeanpuolimmaista bittiä (11111111**1**) kutsutaan vähiten merkitseväksi bittiksi, ja kuvasteganografia muiden steganografisten sovellusten tavoin hyödyntää tätä bittiä tiedon piilottamiseen, sillä bitin muuttaminen ei yleensä merkittävästi vaikuta alkuperäisen kuvan tai tiedoston ulkonäköön tai sisältöön. Otetaan lyhyt esimerkki Shakyaa (2020) mukaillen. Salainen viesti, joka halutaan välittää, on 104 eli numerot 1, 0 ja 4, Suomen pelastustoimen säteilyhälytyskoodi (Salonen & Okkolin 2008, s. 26). Ilmaistuna 3-bittisenä binäärilukuna nämä luvut ovat 001, 000 ja 100. Alkuperäinen kuva on kolme pikseliä sisältävä kuva, joka on luotu osoitteesta <https://make8bitart.com> löytyvällä kuvatyökalulla.

Salauks toteutetaan siten, että kukin binääriluku upotetaan merkki kerrallaan kuvan värin R-, G- tai B-kanavaan (*channel*). Siten siis joko punaisen, vihreän tai sinisen värisävyn vähiten merkitsevää bittiä käytetään ilmaisemaan uuden luvun (001,000,100) osia. Salauks kokonaisuudessaan on esitetty taulukossa 1. Taulukossa vähiten merkittävä bitti on lihavoitu ja salauksessa muutetut arvot on merkitty punaisella värillä.

Taulukko 1 Esimerkki LSB-salauksen toteutuksesta.

Alkuperäinen kuva		Stegokuva		
magenta 	RGB(255 ,0,255) R 1111111 1 G 0000000 0 B 1111111 1 väri #ff00ff	RGB(254 ,0,255) R 1111111 0 G 0000000 0 B 1111111 1 väri #fe00ff		Luvun 1 (001) salaus
mintunvihreä 	RGB(66,252, 143) R 0100001 0 G 1111110 0 B 1000111 1 väri #42fc8f	RGB(66,252, 142) R 0100001 0 G 1111110 0 B 1000111 0 väri #42fc8E		Luvun 0 (000) salaus
turkoosi 	RGB(42,209,252) R 0010101 0 G 1101000 1 B 1111110 0 väri #2ad1fc	RGB(43,208,252) R 0010101 1 G 1101000 0 B 1111110 0 väri #2BD0FC		Luvun 4 (100) salaus

Kuten esimerkistä havaitaan, LSB-salauksen käyttö ei välttämättä edellytä, että vähiten merkitsevää bittiä muutettaisiin aina. Othman (2020, s. 12) arvioikin, että LSB-salausta hyödyntävässä kuvasteganografiassa noin puolet biteistä vaihdetaan toiseksi. Kuvassa 4 on esitetty alkuperäinen peitekuva ja edellä kuvatun salaviestin sisältävä stegokuva allekkain.



Kuva 4. Esimerkki LSB-salauksesta kuvasteganografiassa.

Tuotettu salaus tai salaviesti ei itsessään ole erityisen vaikuttava, mutta se havainnollistaa LSB-tekniikan toteuttamista. Kuvien välillä on erittäin vaikea havaita eroja silmämääräisesti, ja käytännön toteutuksessa muutetut pikselit olisivat vain pieniä pisteitä lopullisessa stegokuvassa. Sopivia sijainteja muutettujen pikseleiden piilottamiseen ovat valokuvan kohinaiset, värikylläiset kohdat (Othman 2020, s. 8).

Salausta ei tarvitse toteuttaa bitti kerrallaan, vaan useita bittejä voidaan korvata kerralla suurempia viestejä välitettäessä. Viestin purkamisen ei myöskään tarvitse olla erityisen monimutkaista tai monivaiheista, vaan tulkintaa voidaan helpottaa tekemällä päätelmiä binääriluvun parillisuudesta tai parittomuudesta. Yksityiskohtiin voi tutustua lähteessä Shakya (2020).

Samaa tekniikkaa hyödyntäen kuvaan voidaan piilottaa haluttuja yksityiskohtia, kuten tekstiä, koodia tai toinen kuva. Kuvan piilottamisesta kuvaan ja useamman bitin muokkaamisesta binääriluvussa Shakya (2020) antaa edelleen esimerkkejä, samoin varoituksen siitä, että useita bittejä kerrallaan muokkaavassa salauksessa myös purettu salaviesti saattaa korruptoitua. Tämä on yleinen huoli kuvasteganografiassa: erilaiset kuvan kompressoinnissa tapahtuneet muutokset ja langattoman tiedonsiirron epävarmuus yleensäkin altistavat steganografian aina tahattomille muutoksille. Muutoksista huolimatta viestin välittyminen ei välttämättä esty, kuten Shakya toteaa, vaan lopputulos toimii kyllin hyvin. Samaa ajattelua korostetaan läpi tekoälyä käsittelevän luvun 4.

Mielenkiintoinen strategia piilottaa kuvasteganografiaa on satunnaisten pikseleiden valinta tavalla, jossa kuva vääristyy mahdollisimman paljon ja tilastolliset oletukset eivät enää päde; tällä pyritään etenkin vastustamaan universaalia steganalyysiä eli steganografian havainnointia ylipäätään. Universaalia steganalyysiä käsitellään luvussa 2.3. Myös satunnaisten pikseleiden valinta tavalla, joka ottaa huomioon JPEG-kuvan

häviöllisen pakkauksen, DCT-muunnokset ja lisäksi hyödyntää virheenkorjauskoodeja (*error correcting codes*) vaikuttaa piilottavan kuvasteganografian ja LSB-tekniikan käytön hyvin. (Solanki et al. 2007, s. 4–7) Tutkimus on tosin toteutettu ennen neuroverkkojen analysointikykyjen esiinnousua, joten vanhat oletukset eivät välttämättä päde, aivan kuten myös steganografian toteuttamiselle on avautunut koneoppimista laajempia mahdollisuuksia syväoppimisessa. Tekoälyn ja koneoppimisen kehitystä käsitellään luvussa 4.1.

Audiosteganografia

Audiosteganografiassa salainen viesti piilotetaan äänitiedostoon. Salainen viesti voi olla lähes mitä vaan digitaalista viestintää: kuvaa, tekstiä, ääntä tai jopa tiedostoja, kuten PDF:iä. Kiinnostavana yksityiskohtana myös populaarikulttuurista tuttu takaisinpeittäminen (*backmasking*), jossa äänitallennetta taaksepäin kelatessa (*rewind*) paljastuu salattu viesti, on audiosteganografiaa. (Domsch 2013, s. 8; Tidmarsh 2023)

Äänitiedoston formaatille, johon piiloviesti kätketään, on useita vaihtoehtoja: erityisen käytettyjä ovat WAV- ja MP3-tiedostot sekä VoIP-tekniikka (AISabhany et al. 2020, s. 8, 10, 14, 20–22). Tutkimuskartoituksessa WAV-tiedostoa hyödynsi peräti 84 % audiosteganografiaa kehittäneistä tutkimuksista ja VoIP-tekniikkaa 7 % (ibid., s. 20). Voi olettaa, että myös käytännön toteutuksissa WAV-tiedosto dominoi kenttää.

Audiosteganografian havaitsemattomuutta ihmistarkkailussa auttaa kuuloaistin heikko vaiheentunnistus ja toisaalta haittaa kuuloaistin herkkyys satunnaiselle kohinalle, jos stegotiedostoa verrataan alkuperäiseen lähteeseen (Othman 2020, s. 4). Audiosteganografiassa hyödynnetään LSB-tekniikan ohella vaiheeseen (*phase*), kaikuun (*echo*) ja hajasppektritekniikkaan (*spread spectrum*) perustuvia salauksia (ibid., s. 6). Audiosteganografian heikkous on digitaalisen kanavan epävarmuudessa: pienetkin virheet lähetyksessä voivat muuttaa hienovaraisesti piilotetun viestin bittejä, jolloin piilotettu viesti korruptoituu. Toisin sanoen useiden audiosteganografisten tekniikoiden kestävyys (*robustness*) on heikko. (AISabhany et al. 2020, s. 8, 10, 14, 20–22)

Videosteganografia

Videosteganografiassa salainen viesti piilotetaan videotiedostoon. Viesti voidaan piilottaa muun muassa videon pakkaamisvaiheessa (*compression*), upottamalla bittivirtaan (*bit stream*), manipuloimalla raakavideon pikseleitä tai käyttäen muunnoksia, kuten DCT:tä (Liu et al. 2019, s. 238). Viestin kätkemiseen käytetty tekniikka suunnitellaan tyypillisesti hyödyntämään jotain nimenomaista videoformaattia, kuten Flash-videota

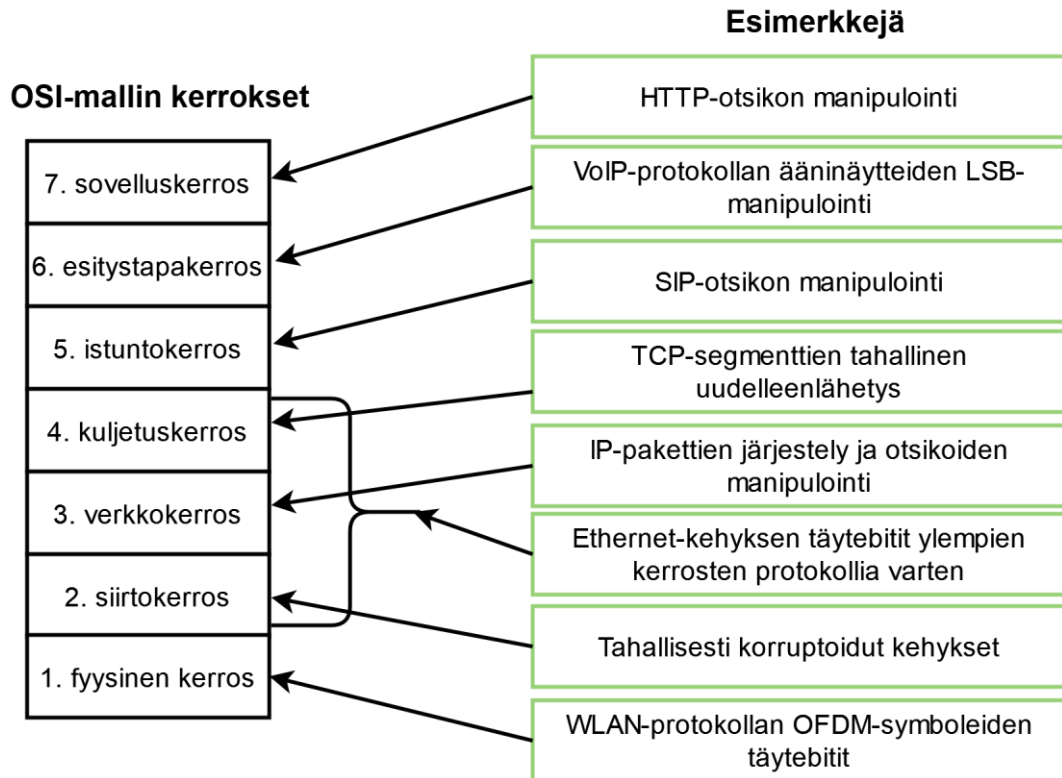
(.flv), Windowsin .avi-formaattia tai videonpakkausstandardeja H.264 (MPEG-4 AVC) ja H.265 (HEVC) (Singh & Singhal 2017, s. 691).

Audiosteganografian tapaan myös videosteganografia on herkkä viestintäkanavan epätäydellisyyksille ja piiloviestin korruptoitumiselle, joskin videosteganografiaa toteuttavien tekniikoiden kestävyudessa (*robustness*) on huomattavia eroja (Jenifer et al. 2018, s. 628–630; Liu et al. 2019, s. 238–239, 244). Viestin piilottaminen heikentää herkästi videon laatua ja altistaa steganografian käytön ulkopuolisen tarkkailijan havaitsemiselle. Parhaan videolaadun lopullisessa viestissä säilyttävät tekniikat puolestaan tyypillisesti mahdollistavat vain suhteellisen pienten viestien kätkemisen (Jenifer et al. 2018, s. 628–629). Videosteganografia kuitenkin steganografian lajina hyötyy sekä näkö- että kuuloaistin vajavaisuuksista: pienet ja vähittäiset muutokset jäävät helposti ihmiskatsojalta havaitsematta, aivan kuten kuva- ja audiosteganografiassa tapahtuvat muutoksetkin. Kuvasteganografiaa voidaan viestin upottamisen jälkeen parannella tekoälyn avulla (Liu et al. 2019, s. 246), joten on luultavaa, että vastaavat edistysaskeleet tulevat käyttöön myös videosteganografiassa, jos näin välitetty viesti muuten kärsisi liian suuresta havaittavuudesta.

Verkkosteganografia

Verkkosteganografia hyödyntää tietoliikenneprotokollien rakenteita salatun tiedon välittämiseen. Erityisen käyttökelpoinen on TCP/IP-pinoprotokolla, jonka käyttämättömiin ja varattuihin otsikkokenttiin (*header field*) on mahdollista piilottaa bittejä ja siten salattua tietoa. Tällaiset lisäykset eivät estä tavanomaisen viestin kulkua tai herätä huomiota verkkoliikennettä tarkkailevassa tahossa, sillä näiden kenttien sisältöä ei tarkasteta vastaanottimessa. Mahdollisia kohteita salatun viestin piilottamiselle ovat TCP-protokollan Flags-kenttä ja IP-protokollan Type of Service (ToS) -kentän ylimääräiset bitit. Myös IP-pakettien järjestysnumerointia on mahdollista käyttää steganografisiin tarkoituksiin. (Lubacz et al. 2014, s. 228)

Verkkosteganografiaa voidaan ajatella myös OSI-mallin eli tietoliikenneprotokollien kerrosmallin pohjalta. Lubacz et al. (2014, s. 227) ovat koonneet kullekin kerrokselle esitetyjä steganografisia metodeja, joiden voi laveasti sanoen katsoa hyödyntävän joko kerrosten protokollien otsikkorakenteita tai kehysten täytebittejä (*padding*), tiedonsiirtotapaa ja pakettien uudelleenlähetyksiä. Kuvassa 5 on esitetty kooste keinoista toteuttaa steganografiaa ja tiedon kätkemistä eri OSI-mallin kerroksilla ja niihin liittyvillä tietoliikenneprotokollilla.



Kuva 5. OSI-mallin mukaan jäsennellyjä keinoja toteuttaa verkkosteganografiaa. Lähteestä Lubacz et al. (2014, s. 227).

Erikseen jaoteltujen digitaalisen steganografian lajien lisäksi steganografialla on muitakin kiinnostavia käyttökohteita. Käsitellään niitä lyhyesti seuraavassa.

Muita steganografian sovellusalueita

Vaikka edellä steganografia on lajiteltu melko tarkkarajaisiin osa-alueisiin, usein tiedon kätkeminen yhdistelee eri elementtejä. Esimerkkinä tästä on sellainen multimediasteganografia, jossa viestintää piilotetaan Microsoft PowerPointin tiedostojen animaatioihin ja äänielementteihin. Yang ja Chen (2008) ovat tutkimuksessaan ehdottaneet, että PowerPointin eri elementteihin – kuviin, tekstiruutuihin, muotoihin – lisätyt animaatiot ja äänet merkitsevät eri asioita. Yhteensä tutkimuksessa löydettiin häkellyttävät 13 miljardia tapaa piilottaa tietoa ohjelmiston elementteihin ja niiden variaatioihin (Yang & Chen 2008, s. 3104). Esimerkiksi tietyllä leikkauksella esiin tuotu, määrätyn värinen viivaelementti merkitsee tiettyä asiaa, kirjainta tai merkkiä. Ei-tekstimuotoinen viestintä edellyttää jaettua koodikirjaa (*code book*) viestin lähettäjän ja vastaanottajan välillä, mutta arkipäiväinen sovellus voi avata odottamattomia tilaisuuksia piilotetulle viestinnälle.

Steganografian soveltuminen laittomiin tarkoituksiin on ilmeistä. Varoittava esimerkki steganografian sovelluksista on *stegware*, steganografiaa hyödyntävä haittaohjelma (*malware*), jossa koodiin, HTML-tiedostoihin, kuviin tai muuhun mediaan on piilotettu

haittaohjelma, jota perinteiset tietoturvaohjelmistot eivät havaitse. Stegware voidaan myös piilottaa tavanomaisen verkkoliikenteen tai tiedostorakenteiden sekaan osana verkkosteganografiaa, jolloin päästyään läpi tietoliikenne- ja tietoturvatarkistuksista haittakoodi purkautuu ja pääsee toteuttamaan tarkoitustaan. (SIMARGL 2020; Monika & Eswari 2023, s. 4401–4403)

Tiedustelutoiminta ja yritysvalvonta ovat niin ikään ilmeisiä steganografian sovellusalueita: vuonna 2023 Yhdysvalloissa annettiin tuomio General Electricin entiselle työntekijälle, joka oli lähettänyt yrityksen liikesalaisuuksia auringonlaskua esittävään kuvaan kätkeytynä. Liittovaltiotasolla huomiota herättänyt teko paljastui sattumalta, kun työntekijän havaittiin käyttäneen yhtiön ulkopuolista työkalua huomattavan suuren kuvamäärän salaamiseen (*encrypt*). (Yhdysvaltain oikeusministeriö 2023)

Toistaiseksi tietoturvapalvelut eivät nimenomaisesti tunnista steganografian käyttöä hyvässä tai pahassa, vaan steganalyysiä toteuttavat tietoturvatyökalut on laadittu lähinnä tutkimuskäyttöön. Keinot stegwaren tunnistamiseen rajoittuvat steganalyysin yleisiin keinoihin, joita käsitellään luvussa 2.3, ja yleiseen tietoturvakehitykseen, tosin on mahdollista, että nimenomaan steganografian tarkastelu nousee merkittävämmäksi myös kyberturvallisuudessa ja tietoturvassa yleensä. Vaikka steganografiaa hyödyntävät hyökkäykset ovat lisääntyneet vuosi vuodelta, on steganografian käyttö edelleen pääosin – olematta liian hauska – kätkeytä. (Oza 2019; SIMARGL 2020; Monika & Eswari 2023, s. 4402–4403)

Steganografisia tekniikoita voisi esitellä hyvinkin laveasti, sillä tutkimusalue on toisaalta poikkeuksellisen laaja ja toisaalta mielenkiintoinen juuri vähäisen tunnettuuden ja arkielämän toteutuksen hienovaraisuuden vuoksi. Tämä tutkielma kuitenkin keskittyy tekstisteganografiaan, jota käsitellään tarkemmin seuraavassa.

2.2 Tekstisteganografia

Agarwal (2013, s. 92) pitää tekstisteganografiaa vaativimpana steganografian lajina. Teksti ei samalla tavoin sisällä ylimääräistä (*redundant*) tietoa kuin muut digitaalisen steganografian alustat, joissa mediaan kuuluu lähtökohtaisesti epätäydellisyys: äänentoiston säröt, pienet epätarkkuudet kuvissa, häiriöt videon toistossa tai muutokset langattomassa tiedonsiirrossa. Sen sijaan tekstin muotoiluun tai sisältöön piilotettu viesti on tarkkailijan herkästi havaittavissa (*ibid.*).

Huomautettakoon kuitenkin, että ihmisen aivot toimintaan kuuluu leimallisesti ilmiö pareidolia – taipumus nähdä merkitystä ja säännönmukaisuutta satunnaisuudessa, kuten eläinhahmoja pilvissä, kasvonkuvia paahtoleivässä tai viestintää kohinassa (Camenzind

et al. 2024, s. 1–2) – joten tarkankaan katsojan havainnot eivät välttämättä paljasta steganografisen viestinnän todellista käyttöä. Satunnainen lukija saattaa havaita omituisia asettelua tai kirjoitusvirheitä tekstissä mutta ohittaa ne huolimattomuusvirheinä. Steganografia kuitenkin pyrkii häivyttämään epäilyksetkin viestinnän olemassaolosta, joten huoli tekstisteganografian liian helpposta havaittavuudesta ei ole aiheeton.

Tekstisteganografia voidaan jaotella kolmeen päätyyppiin (Por & Delina 2008, s. 690; Agarwal 2013, s. 92–93; Majeed et al. 2021, s. 8):

1. **Muotoon perustuvat tekniikat** (*format-based methods*)
2. **Satunnaisuuteen ja tilastollisuuteen perustuvat tekniikat** (*random and statistical generation*)
3. **Lingvistiset tekniikat** (*linguistic methods*)

Usein tekstisteganografian käyttäminen ei kuitenkaan rajoitu tarkasti määriteltyyn kategoriaan. Tutkimusjulkaisut eivät välttämättä kategorisoi esittämiään tutkimustuloksia tekstisteganografiaa kummemmin, ja myös toisenlaisia jaotteluja on esitetty sen kummemmin perustelematta. Esitellään kuitenkin lyhyesti esitetyn jaottelun mukaisia tapoja toteuttaa tekstisteganografiaa.

Muotoon perustuvat tekniikat hyödyntävät tekstin tai merkkien asettelua ja erilaisia fontteja salaisten viestien välittämiseen. Osa tekniikoista soveltuu vain määrätyn aakkoston tai merkistön kanssa hyödynnettäväksi, kuten arabiankieliseen kirjoitukseen tai Unicode- ja ASCII-merkistöön piilotettu viestintä. (Majeed et al. 2021, s. 17–18) Yleisiä tekniikoita ovat myös ylimääräisten välilyöntien (*whitespace*) hyödyntäminen ja sanojen tai rivien siirtäminen (*word shift, line shift*) (Agarwal 2013, s. 93).

Satunnaisuuteen ja tilastollisuuteen perustuvat tekniikat ovat sekalainen kokoelma steganografista viestintää. Niillä voidaan tuottaa piiloviestin sisältävää mutta silti satunnaiselta vaikuttavaa tekstiä tilastollisiin sanojen esiintymis- ja järjestystodennäköisyyksiin perustuen. Toisaalta voidaan myös luoda pituudeltaan ja kirjainten esiintymistiheydeltään uskottavia ”sanoja” tai pikemminkin merkkijonoja, joilla ei ole todellista leksiikaalista merkitystä. Viesti voidaan piilottaa tällä tavoin luotuun peitetekstiin joko bittitasolla tai sitten se voidaan piilottaa itse sanoihin. (Por & Delina 2008, s. 690–691)

Majeed et al. (2021, s. 20–21) katsauksessa satunnaisuuteen ja tilastollisuuteen perustuvat tekniikat hyödyntävät monipuolisesti erilaisia alustoja. Näihin lukeutuvat muun muassa RSA-salausta eli julkisen avaimen salausta hyödyntävä steganografia, sähköpostin cc-kenttää eli kopiokenttää hyödyntävä salaus ja Markovin ketjuja (*Markov chain*) hyödyntävät salaukset. Myös nämä tekniikat voivat olla sidoksissa tietynlaiseen merkistöön,

kuten kiinalaisten kirjoitusmerkkien pariteettia eli binääriluvun parillisuutta tai parittomuutta hyödyntävä salaus.

Lingvistiset tekniikat kiinnittävät huomion peitetekstin ja stegotekstin kielellisiin ominaisuuksiin. Peiteviesti voidaan piilottaa syntaksiin, ja peiteteksti yleensä noudattaa kohdekielen rakenteita. (Por & Delina 2008, s. 691) Salaviesti voidaan upottaa olemassa olevaan tekstiin (*embedded*) tai salaviestin välittämiseen käytetty teksti voidaan generoida eli tuottaa steganografiaa varten (*generated*). (Ziegler et al. 2019, s. 1) Lingvistinen steganografia voidaan siten jaotella muokkaukseen perustuviin (*edit-based*) tai generoimiseen perustuviin (*generation-based*) tekniikoihin. (ibid.)

Huolimatta semantiikan ja kieliopillisuuden erityisestä merkityksellisyydestä myös lingvistisessä steganografiassa salaviesti ilmaistaan tyypillisesti bitteinä varsinaisten sanojen sijaan. Tällöin on sovittava merkistöstä, jota viestijät käyttävät. Usein käytettyjä vaihtoehtoja ovat ASCII ja Unicode. Esimerkiksi isoilla kirjaimilla kirjoitettu hätäviesti ”SOS” olisi 8-bittisellä ASCII-merkistöllä ilmaistuna 01010011 01001111 01010011 (välit luettavuuden vuoksi) (ASCII 2024), mutta nolista ja ykkösistä koostuvan binäärikoodin merkitys on vapaasti viestijöiden sovittavissa.

Generatiivinen lingvistinen steganografia eli generoitua tekstiä hyödyntävä tekstisteganografia on noussut viime vuosien trendiksi. Vaikka erillisen peitetekstin tuottaminen on esitetty mahdollisuutena jo kymmeniä vuosia – esimerkkejä tästä ovat Waynerin (1992) yhteydettömät matkintafunktiot (*context-free mimic functions*) ja Bennettin (2004, s. 16–21) selonteko generatiivisesta lingvistisestä steganografiasta – todellinen mahdollisuus generatiiviseen lingvistiseen steganografiaan alkoi kuitenkin neuroverkkojen ja kielimallien kehityksen myötä vuodesta 2017 alkaen (Ziegler 2019, s. 1). Neuroverkkoihin tutustutaan luvussa 4 ja niitä hyödyntäviin suuriin kielimalleihin luvussa 5.

Viime vuosien lingvistisen steganografian kehitystyö onkin pitkälti keskittynyt nimenomaan generatiivisen lähestymistavan ympärille, kuten Majeed et al. (2021, s. 12–21) katsauksessa ilmenee. Katsauksessa pääosin mainittuja lingvistisen steganografian tutkimuskohteita ovat olleet nimenomaan neuroverkkojen avulla tuotetut steganografisen viestin sisältävät tekstit, jotka aiempaa paremmin piiloutuvat ihmis- ja konetarkkailijalta muun tekstin sekaan. Lingvistisen steganografian kapasiteettia (*capacity*) on parannettu erilaisilla tilastollisilla analyyseillä ja edelleen neuroverkkojen suomilla mahdollisuuksia. Myös Huffmanin koodauksella (*Huffman coding*) toteutettuun häviöttömään pakkaamiseen ja tiettyjen kielten (arabia, kiina) ominaisuuksiin perustuvaa lingvististä steganografiaa on kehitetty edelleen.

Kiinnostava generatiivinen sovellus on myös viestinnän naamioiminen roskapostiksi (*spam*). Waynerin (1992) matkintafunktioihin perustuvaa roskapostikonetta voi kokeilla osoitteessa www.spammimic.com. Kone tuottaa erehdyttävästi roskapostia muistuttavaa tekstiä, johon on piilotettu bittejä ja bittijonoja, tosin tuotettu stegoteksti on lyhyelläkin salaviestillä huomattavan pitkä ja voisi siten käytössä herättää huomiota muiden roskapostien joukossa. (Bennett 2004, s. 8–9)

Ei-generatiivisista lingvistisen steganografian sovelluksista Majeed et al. (2021) mainitsevat synonyymien hyödyntämisen viestin piilottamisessa. Synonyymien käyttöä lingvistisessä steganografiassa ovat tutkineet muun muassa Lockwood ja Curran (2017), joiden sovelluksessa sana voidaan korvata synonyymillään tai saman aihepiirin sanalla steganografisen viestin lähettäjän ja vastaanottajan jakamaa tesaurusta (*thesaurus*) käyttäen. Tesaurus antaa hakusanaa vastaan saman aihepiirin sanoja: esimerkiksi sanalle *kissa* voidaan käyttää useita synonyymeja, mukaan lukien osuvat *kissaeläin*, *kolli* ja *kotikissa* tai vähemmän osuvat *leijona* ja *pantteri*. Tutkimuksessa käytetty englanninkielinen hakuteos antoi sanalle *kissa* 16 vastaavaa synonyymiä.

Hakuteoksen sisältämien synonyymien lukumäärän mukaan sanoille annettiin ulkopuoliselle tarkkailijalle tuntemattomia bittiarvoja, joiden sisältämä viesti oli edelleen vain viestijöiden tiedossa. Tällainen synonyymistä sanankorvausta hyödyntävä viestintä voi olla täysin luonnollista: kissan vieminen eläinlääkäriin virkkeillä ”Vein kissan eläinlääkäriin” ja ”Vein kollin eläinlääkäriin” voi tarkoittaa samaa asiaa, mutta eri muodon käyttäminen viestii silti jotain salattua. (Lockwood & Curran 2017, s. 7–8)

Majeed et al. (2021) katsauksessa mainitaan myös Microsoft Wordin kommenttityökalun tarjoamat mahdollisuudet salattujen viestien välittämiseen. Samoin on tutkittu bittimuotoisten viestien piilottamista englanninkielisten tekstiviestien yleisiin slangilyhenteisiin (Shirali-Shahreza & Shirali-Shahreza 2007). Lingvististä steganografiaa toteutetaan siis monipuolisilla ja kekseliäilläkin työkaluilla, niin generatiivisena kuin ei-generatiivisenakin.

Vaikka binäärikoodia eri tavoin tekstin sekaan piilottava tekstisteganografia hallitsee tutkimuskenttää, se ei missään nimessä ole koko totuus tekstisteganografian käyttöta-voista. Etenkin toisen maailmansodan aikaan hyödynnettiin muiden steganografisten viestintätapojen, kuten mikropisteiden (*microdot*) ja näkymättömien musteiden, ohella nollasalausta (*null cipher*), jossa salaviestit on piilotettu tekstinä tekstin lomaan. Nimensä mukaisesti tällaista viestiä ei ole kryptologian keinoin salattu tai kryptattu (*encrypt*) ennen sen välittämistä ja kätkemistä steganografian avulla; useinhan näitä kahta salaustiedettä käytetään samanaikaisesti, ja niiden erottelu voi toisinaan tuntua keinotekoiselta.

Othman (2020, s. 2–3) esittää tunnettuja esimerkkejä tekstin piilottamisesta tekstiin nol-lasalauksen avulla. Ensimmäisessä, merihenkisessä mainosviestissä todellinen viesti saadaan ottamalla kunkin sanan kolmas kirjain:

Fishing freshwater bends and saltwater coasts rewards anyone
feeling stressed. Resourceful anglers usually find masterful leapers
fun and admit swordfish rank overwhelming anyday.

Näin piilotettu viesti olikin suora avunpyyntö ja lista pyydettävistä asioista:

Send lawyers, guns, and money.

Toinen viesti – väitetysti toisen maailmansodan aikainen akselivaltojen tiedusteluviesti – sisältää todellisen viestin piilotettuna kunkin sanan joka toiseen kirjaimen:

Apparently neutral's protest is thoroughly discounted and
ignored. Isman hard hit. Blockade issue affects pretext for embargo
on by products, ejecting suets and vegetable oils.

Näin piilotettu viesti sisälsikin tiedon liittoutuneiden upseeri Pershingin liikkeistä:

Pershing sails from NY June 1.

Tämän tutkielman käytännön osuus perustuu nimenomaisesti tällaiseen sanallisesti toteutettuun lingvistiseen steganografiaan ja sen erityistapauksena suomalaisessa sotilas-käytössä tärkeään sanankorvauksella toteutettuun peittämiseen, jota suomen kielen rakenteen ohella käsitellään kautta luvun 3. Peittäminen muistuttaa ehkä lähimmin Chapmanin ja Davidan (1997) esittelemää NICETEXT-toteutusta, jossa sanat on jaoteltu semanttisen merkityksen ja käyttötarkoituksen mukaisesti kategorioihin, joiden sisällä sanoja voidaan korvata toisella. Näin tuotetaan harmittoman oloista mutta oikean tyylistä 'kivaa tekstiä', *nice text*. Samantyylinen toiminnallisuus sisältyy peittämisessä sanastojen käyttöön, kun substantiivit korvataan substantiiveilla ja adjektiivit adjektiiveilla, joskaan peitesanastojen laatimista ei käsitellä tässä tutkielmassa.

Jatkotutkimuksessaan Chapman et al. (2001) kehittivät NICETEXT-toteutusta edelleen. Muutoksia haluttiin etenkin sen vuoksi, että huomiota herättämättömän pidemmän tekstin laatiminen lausemalleilla ja semanttisilla synonyymeilla on haastavaa, vaikka lyhyt teksti onkin melko helppo tuottaa (Chapman et al. 2001, s. 1). Jatkotutkimuksessa mallille annetusta samasta syötteestä saatiin useita syntaktisesti uskottavia stegotekstejä, kun lauseen syntaksianalyysin pohjana käytettiin halutun tyylistä esimerkkitekstiä (ibid., s. 4–6). Lähes vuosikymmen myöhemmin Yang ja Cao (2010) ratkoivat tutkimuksessaan edelleen samaa ongelmaa: pitkä generoitu stegoteksti paljastuu todennäköisemmin kuin

lyhyt. Tuossa tutkimuksessa englanninkielistä peitetekstiä analysoitiin yksityiskohtaisesti, ja näin selvitettyjä kielen metapiirteitä (*meta features*) sovellettiin erilaisilla matemaattisilla malleilla uskottavan stegotekstin laatimiseen. Tunnistettuihin metapiirteisiin lukeutuivat erot stegotekstien ja tavallisten tekstien sanojen pituudessa ja sanojen sisältämien kirjainten yleisyydessä. (Yang & Cao 2010, s. 662–663)

Itse asiassa Chapmanin ja Davidan toteutuksessa esitetty sanojen koneellinen kategorisointi muistuttaa yksinkertaisuudessaankin yllättävän paljon 20 vuotta myöhemmin kehitettyjä suurten kielimallien *embedding*-vektoreita, jossa sanan ominaisuudet tallennetaan matemaattisesti monisata- tai monituhatulotteiseen vektoriin. Samoin toteutuksen kyky tuottaa tekstiä mallina annettua tekstityyliä matkien on juuri samanlaista generatiivista tekstintuottoa, jota tämän hetken suuria kielimalleja hyödyntävät tekoälysovellukset tekevät monipuolisesti. Suuria kielimalleja ja *embedding*-vektoreita käsitellään luvussa 5. Tässä yhteydessä lienee kuitenkin mielekkäintä ajatella peittämistä eräänlaisena lingvistisen steganografian erikoistapauksena ilman lähempiä yhtäläisyyspohdintoja.

2.3 Steganalyysi ja steganografian heikkoudet

Steganografian vastinpari on steganalyysi (*steganalysis*), jolla steganografista viestintää pyritään tunnistamaan, purkamaan ja häiritsemään. Passiivinen steganalyysi tähtää ainoastaan steganografian käytön tunnistamiseen tai toisinaan häiritsemiseen, kun taas vaativampi aktiivinen steganalyysi tähtää tunnistamisen lisäksi viestin purkamiseen. (Cheddad et al. 2010, s. 745; Karampidis et al. 2018, s. 218) Passiivista tunnistamista voidaan kutsua myös nimellä universaali tai sokea tunnistus (*universal detection, blind detection*) (Karampidis et al. 2018, s. 218) mukailien ajatusta, että steganografian toteutustapa (algoritmi) ei ole tunnettu. Oikeastaan on hedelmällistä ajatella, että steganalyysi on myös steganografisen algoritmin validointia: nerokkainkin algoritmi on tarkistettava ja tarvittaessa vahvistettava steganalyysiä vastaan.

Nimitystä ”universaali” käytetään kuitenkin lähinnä steganografian tunnistamisesta tietyn median sisällä: kuvasteganalyysille kehitetty universaali tunnistus ei auta tekstisteganalyysissä. Harvinaisena poikkeuksena LSB-tekniikan käytölle digitaalisessa mediassa yleisesti on ehdotettu matemaattista steganalyysimenetelmää (Dumitrescu & Wu 2005). Tapaustutkimuksessa menetelmää kuitenkin sovellettiin yksinomaan kuvasteganografiaan, vaikka sen ilmoitettiin soveltuvan myös äänelle ja videolle.

Hyökkäykset steganografiaa vastaan – tai kääntäen steganalyysin keinot – voidaan jakotella sen mukaan, tunnetaanko kuvan 2 mukaiset stego-objekti, peiteobjekti, salaviesti ja steganografinen algoritmi tai ohjelmisto (*tool*). Esimerkkejä algoritmeista eli tavoista

toteuttaa steganografiaa ovat LSB ja DCT-muunnos. Helpoin tekniikoista on **tunnettu stego -hyökkäys** (*known stego attack*), jossa tunnetaan kaikki osat salaviestiä lukuun ottamatta. Astetta vaikeampi toteutettava on **valittu viesti -hyökkäys** (*chosen message attack*), jossa luodaan stego-objekti käyttäen tunnettua algoritmia ja salaviestiä. Saadusta stego-objektista pyritään tunnistamaan kyseisen algoritmin piirteitä (*signature*), joiden avulla havaita muita stego-objekteja viestinnästä. Tätä vaativammassa **valittu stego -hyökkäyksessä** (*chosen stego attack*) puolestaan saatavilla ovat stego-objekti ja sen luomiseen käytetty algoritmi. Puolestaan **tunnettu viesti -hyökkäyksessä** (*known message attack*) saatavilla ovat stego-objekti ja sen sisältämä salaviesti. Tästä vaikeammassa **tunnettu peite -hyökkäyksessä** (*known cover attack*) vertaillaan tunnettua peite- ja stego-objektia. Vaikein tekniikoista on **vain stego -hyökkäys** (*stego only attack*), jossa tunnetaan ainoastaan stego-objekti, mutta ei mitään muuta. (Karampidis et al. 2018, s. 217–218) Hyökkäystyypit on esitetty taulukossa 2.

Taulukko 2 Hyökkäystyypit steganografiaa vastaan helpoimmasta vaikeimpaan. Lähteestä Karampidis et al. (2018, s. 217–218).

	Tunnettu stego	Valittu viesti	Valittu stego	Tunnettu viesti	Tunnettu peite	Vain stego
Stego-objekti	✓	✗	✓	✓	✓	✓
Peiteobjekti	✓	✗	✗	✗	✓	✗
Salaviesti	✗	✓	✗	✓	✗	✗
Algoritmi	✓	✓	✓	✗	✗	✗

Steganografisen viestin sisältävän median laatu määrittelee siihen soveltuvia steganalyysin keinoja. Kuitenkin voidaan tehdä seuraava yleisluontoinen jaottelu steganalyysin tekniikoista (Karampidis et al. 2018, s. 218):

- **Tilastollinen steganalyysi**, jossa alkuperäisen median ja stegomedian ominaisuuksia analysoidaan tilastollisesti
- **Koneoppimiseen ja neuroverkkoihin perustuva steganalyysi**, jossa neuroverkolle annetaan esimerkkejä stegomediasta. Näin koulutetaan koneellinen luokittelija (*classifier*), joka tunnistaa syötteenä saamastaan mediasta steganografialle tyypillisiä piirteitä ja käyttää oppimaansa tietoa steganografian tunnistamiseen toisaalla. Ohjattua oppimista käsitellään neuroverkkojen yhteydessä luvussa 4.1

Näiden lisäksi on oleellisenä huomioitava **ihmistarkkailijan havainnointi**. Mikäli teksti vaikuttaa liian oudolta tai kuva liian vääristyneeltä, steganografia uhkaa paljastua, ja

avainasemassa steganografiassa on salaisen viestinnän olemassaolon kätkeyminen. Ihmistarkkailija ei luultavasti ole keskeisessä roolissa reaaliaikaisessa verkkosteganografiassa tai sitä havainnoivassa verkkosteganalyysissä, jos viestinnän välineenä käytetään datapaketteja. Mikäli verkkosteganografia taas muuttaa VoIP-puhelun laatua liiaksi, tarkkailijan mielenkiinto voi herätä.

Steganalyysin keinot vaihtelevat siis stegomedian mukaan. Kuvasteganalyysi hyödyntää kuvan manipulointia suodatuksella (*filtering*), kääntämisellä (*rotating*), leikkauksella (*cropping*) ja kuvakääntämisellä (*translating*). Kuvakääntämisessä kuva konekäännetään tekstintunnistusohjelman (*Optical Character Recognition*, OCR) avulla, jolloin kuvasta voidaan irrottaa siihen upotettua tekstiä. Tilastollisessa steganalyysissä voidaan tarkastella kuvan histogrammia (ensimmäisen kertaluvun tilastot) tai pikselien ominaisuuksia, kuten korrelaatiota, etäisyyttä ja suuntaa (toisen kertaluvun tilastot). Myös DCT-muunnosta ja muita muunnoksia hyödyntävän steganografian kertoimet (*coefficients*) voivat osaltaan auttaa kuvasteganalyysissä. (Cheddad et al. 2010, s. 745)

Tekstisteganografiassa erilaisiin tilastollisiin kielimalleihin perustuvaa steganalyysiä on hyödynnetty muun tilastollisen steganalyysin rinnalla jo 2000-luvun alkupuolelta saakka. Katsaukseen kielimalleja ja monenlaisia neuroverkkoja hyödyntävistä universaaleista tekstisteganalyysimenetelmistä voi tutustua lähteessä Yang et al. (2022).

NICETEXT-tekniikkaa ja Markovin ketjuja hyödyntävät salausmenetelmät tuottavat tilastollisilta ominaisuuksiltaan uskottavaa stegotekstiä, mutta tuotettu teksti ei ole luonnollista, jolloin ihmistarkkailija erottaa sen helposti. Tekniikoita hyödyntävä tekstisteganografia sopiikin vain tilanteisiin, joissa steganalyysiä suorittaa konetarkkailija. (Chen et al. 2008, s. 225) Näihin tekniikoihin perustuva stegoteksti, vaikka kieliopillisesti ja tilastollisella tarkastelullaan korrekta, eroaa etenkin korrelaatioiltaan: lauseen sisällä sanat eivät liity toisiinsa luontevasti toisiinsa. Generoitu teksti yhdistelee sanoja väärin, kuten lauseessa "She is an actor", kun luonnollisessa nykyenglannissa edelleen käytettäisiin ilmaisua "She is an actress". Steganalyysin keino generoidun tekstin erottamiseen olisikin tässä tapauksessa epätyypillisten korrelaatioiden havainnointi tekstikorpuksen ja luokitelijan avulla.

Chen et al. (2008, s. 229–234) tutkimuksessa stegotekstin tunnistamisessa päästiin 94–98 % tarkkuuteen, tosin luotettava havainnointi edellytti varsin suurta tekstiaineistoa, kooltaan vähintään 20 kB. Suurin resistenssi havainnointia vastaan oli Markovin ketjuihin perustuneella stegotekstillä kuitenkin siitä huvittavasta syystä, että tekniikka toisinaan lisää ylimääräisiä välilyöntejä sanojen väliin, jolloin generoidun stegotekstin ominaisuudet muistuttavat enemmän tavanomaista vertailuaineistoa.

Tätä seuranneessa Meng et al. (2009) jatkotutkimuksessa, jossa niin ikään hyödynnetään tilastollista kielimallia, Chen et al. (2008) tutkimuksen löydöksiä ja tutkimusasetelmaa sekä lisäksi tavallisen ja stegotekstin perpleksiteettiä (*perplexity*), päästiin jo lähes 94 % havainnointitarkkuuteen kymmenesosalla aineiston koosta, vain 2 kB kokoisella tekstiaineistolla. Havainnointitarkkuus ylitti 96 %, kun aineiston koko oli 5 kB. Perpleksiteetti viittaa kielimallin kykyyn ennustaa tekstin rakentumista. Yleistajuisesti aiheeseen voi tutustua Chiusano (2022).

On syytä huomata, että Chen et al. (2008) ja Meng et al. (2009) tutkimukset on tehty ennen laajamittaista tekoälyn läpimurtoa vuonna 2017. Sitä seurannut suurten kielimallien ja transformer-neuroverkkojen yleistymisen on aiheuttanut sen, että kyvykkyys generatiiviseen kielentuottoon on kasvanut ja monipuolistunut ilmiömäisesti. Ei ole kohtuutonta spekuloida, että myös kyvykkyys toteuttaa steganografiaa on samalla edistynyt harppauksin. Koska steganalyysi seuraa väistämättä uuden tekniikan jäljessä, parhaasakin tapauksessa vanavedessä, kiintoisa kysymys onkin, onko tilastollisen steganalyysin kyky tunnistaa stegomediaa suhteessa parantunut vai heikentynyt. Samoin voidaan pohtia, pätevätkö aiemmat oletukset lainkaan, kun tavallista tekstiä ja kuvaa laatiikin nyt heikosti ennustettava ja uudella tavalla arvaamaton tekoäly ihmisen sijaan.

Wang et al. (2023) toteuttavat lingvististä steganalyysiä samoilla periaatteilla kuin generatiivisten suurten kielimallien opettaminen. Itseohjautuvalla opettamisella (*self-supervised learning*) esiopetettu kielimalli saadaan pienellä määrällä opetusnäytteitä tunnistamaan eri algoritmeilla toteutettua generatiivista steganografiaa erityyppisistä teksteistä: elokuva-arvosteluista, lyhyistä sosiaalisen median päivityksistä ja uutissivustoilta.

Erona kielimalleihin tämä Wang et al. (2023) kehittämä muutaman näytteen steganalyysi (*few-shot steganalysis*) hyödyntää bayesilaista neuroverkkoa (*Bayesian neural network*, BNN), jossa neuroverkon opettamisessa säädettävät painokertoimet ovat lukuarvojen sijaan todennäköisyysjakaumia. Tutkimuksessa saatiin lupaavia tuloksia pienellä aineistolla eri algoritmeja tunnistavasta lingvistisestä steganalyysistä, ja samalla kierrettiin niin kielimallien kuin steganalyysinkin tyypillinen ongelma: koneoppimisen tarvitsema suuri, mielellään myös luokiteltu opetusaineisto. Neuroverkkojen rakenteita käsitellään luvussa 4.2, neuroverkkojen opettamista luvussa 4.3 ja kielimallien opettamista luvussa 5.3.

Wang et al. (2023) tutkimuksessa testattiin kuitenkin vain kahta neuroverkoista kumpuaavaa lingvistisen steganografian algoritmia. Nämä olivat generatiiviseen tekstisteganografiaan takaisinkytketyillä neuroverkoilla perustuva Huffmanin koodaus (Yang et al. 2019; myös Shen et al. 2020) ja datan pakkausta todennäköisyysjakaumien avulla hyödyntävä aritmeettinen koodaus (Ziegler et al. 2020). Siten edelleen on kiinnostava kysymys,

kuinka tutkimuksessa esitetty lupaava, joskin pienimuotoinen muutaman näytteen steganalyysi yleistyisi muihin algoritmeihin tai kenties jopa ilman neuroverkkoja tuotettuun tekstisteganografiaan. Kohtuudella voi joka tapauksessa ennustaa, että kielimallien ja alati monipuolistuvien neuroverkkojen rooli on tulevaisuuden steganografiassa ja steganalyysissä keskeinen – todennäköisesti tavalla, jota ei vielä voida ennakoida.

Tekstisteganografia on aiemmin luvussa 2.2 nimetty haastavimmaksi steganografian lajiksi. Toisaalta tätä heikkoutta saattaa kompensoida tekstisteganografialle ominainen resistenssi steganalyysiä vastaan. Digitaalisiin kuviin ja multimediaan verrattuna digitaalisen tekstin piirteet ovat moninaisia ja heikosti yksilöitävissä juuri steganografian läsnäoloon. Salaviestin upottamisessa on myös huomattavan paljon variaatiota, ja upottaminen vääristää alkuperäistä mediaa vain vähän (*low embedding distortion*). (Ahvanooey et al. 2019, s. 10) Tarkka tunnistus (*specific detection*), jossa käytetty algoritmi tai steganografinen keino on tunnettu, on odotetusti yleisluontoista universaalia tunnistusta tarkempi keino havainnoida tekstisteganografiaa. Universaalista lähestymistavasta houkuttelevan tekee kuitenkin sen potentiaali havainnoida myös aiemmin täysin tuntemattomia salaviestin upottamistapoja. (ibid., s. 10–11)

Ahvanooey et al. (2019, s. 11) jaottelevat tekstisteganalyysin hyökkäykset kolmeen kategoriaan:

1. **Visuaaliset hyökkäykset** (*visual attacks*), joissa yleensä ihmistarkkailija havaitsee tekstin syntaksissa, ilmaisussa tai kieliasussa jotain poikkeavaa. Satunnaisuuteen ja tilastollisuuteen perustuva tekstisteganografia on erityisen altis paljastamaan ihmistarkkailijan tarkastelussa
2. **Rakenteelliset hyökkäykset** (*structural attacks*), joissa salaviestit tuhoetaan muuttamalla tekstin ulkoasua tai muotoilua. Muotoon perustuva tekstisteganografia on erityisen herkkä tälle, tosin samalla se on myös kestävä vääristelyä (*distortion*) vastaan. Toisin sanoen muotoon perustuvalla tekniikalla laadittu salaviestit on helppo tuhota, mutta hyökkääjän on vaikea muuttaa sen sisältöä
3. **Satunnaisuuteen ja tilastollisuuteen perustuvat hyökkäykset** (*random and statistical attacks*), joissa tarkastellaan sanojen esiintymisen todennäköisyysjakamaa ja haetaan tilastollisia eroavaisuuksia tunnetun tekstin sekä tarkasteltavan tekstin välillä steganografian paljastamiseksi

Koontitutkimuksensa perusteella Ahvanooey et al. (2019) tarkastelevat eri tekstisteganografian lajien (luku 2.2) ominaisuuksia viestinnässä (kuva 1) ja vastustuskykyä steganalyysiä vastaan. Vääristelynkäytävyydellä (*distortion robustness*) tarkoitetaan tekniikan kykyä vastustaa salaviestin muuntelua yrittävää hyökkäystä ilman, että muutokset näkyisivät viestijöille. Sillä voidaan viitata myös tietyllä tekniikalla toteutetun tekstisteganografian yleiseen alttiuteen vääristyä viestintäkanavassa. (ibid., s. 9) Näkökulma on siis hieman eri kuin steganografiassa kestävyydellä (*robustness*) tavanomaisesti, sillä

korkea vääristelykestävyys ei sulje pois viestin herkkyyttä tuhoutumiselle, ainoastaan muuntelulle. Tutkimuksen havainnot on koottu taulukkoon 3.

Taulukko 3 Tekstisteganografian eri lajien ominaisuuksia ja kyky vastustaa steganalyysiä. Lähteestä Ahvanooy et al. (2019, s. 15).

Tekstisteganografian laji	Havaitsemattomuus	Kapasiteetti	Vääristelykestävyys	Huomioita
Lingvistiset tekniikat	● ● ●	●	● ●	- Kielikohtainen (englanti, arabia) - Korkea kompleksiteetti
Muotoon perustuvat tekniikat	● ● ●	● ● ●	● ● ●	- Salaviestin upottaminen ei muuta alkuperäisen tekstin sisältöä
Satunnaisuuteen ja tilastollisuuteen perustuvat tekniikat	●	● ●	● ● ●	- Sidoksissa alkuperäisen tekstin kieleen - Korkea kompleksiteetti

Selitykset: ● ● ● = Korkea, ● ● = Keskitasoinen, ● = Matala.

Ilman tarkoituksellistakin ulkopuolista häirintää steganografista viestintää on mahdollista häiritä tahattomasti. Stegokuvan tallentaminen häviöllistä pakkaamista hyödyntävään JPEG-muotoon tuhoaa herkästi etenkin LSB-tekniikkaa hyödyntävän salaviestin, tosin vaikkapa DCT-muunnoksia hyödyntävät tekniikat voivat olla resistentimpiä peukaloinnille (Cheddad et al. 2010, s. 734; Othman 2010, s. 11). Monet sisällönjako- ja viestintäpalvelut, kuten Instagram, Facebook ja Outlook, pakkaavat automaattisesti niillä jaettuja tiedostoja (Christian 2022a, Christian 2022b, Microsoft 2024), näkyvimmin kuvatiedostoja. Vaikka viestintävälineenä ne ovat erinomaisen tavallisia ja täyttävät siten hyvin onnistuneen steganografisen viestinnän kriteerin kyllin yleisen viestintäkanavan käytöstä (luku 2.1), niiden käyttökelpoisuus steganografiseen viestintään voi olla heikko tai sellaiseksi muuttua viestijän toiminnasta riippumatta.

Sama ongelma koskee luonnollisesti ääni- ja videotiedostojen pakkaamista. Asetteluun perustuvalla tekstisteganografialla tuotettu salaviesti on helppo tuhota yksinkertaisesti kopiaamalla ja liittämällä (*copy-paste*) viesti uuteen pohjaan. Hieman epäonnistunut valokopiointi tai tulostus on niin ikään hyvin yksinkertainen tapa häiritä tällaista viestintää. (Ahvanooy et al. 2019, s. 11) Digitaalisen kanavan bittivirheet tai epätäydellisyydet, vaikka luultavasti merkityksettömiä tavanomaisessa viestinnässä, voivat olla tuhoisia hienovaraisessa steganografisessa viestinnässä.

Kohtuudella voi spekuloida, että etenkin tekstisteganografiassa tietynlaista outoutta siedetään tiettyyn pisteeseen asti; tämän voi todeta kuka tahansa yleisillä keskustelupalstoilla kävijä, sillä vaikeaselkoisia ja jokseenkin järjettömiä viestejä on ollut jo silloin, kun minkäänlaista varteenotettavaa, julkisesti hyödynnettävää generatiivista tekoälyä ei ole ollut olemassakaan. Samoin epätäydellisyydet kuvasteganografiassa ja muussa median käytössä voidaan yksinkertaisesti sivuuttaa asiaankuuluvina. Kaiken lopulta ratkaisee käyttötarkoitus: jos tekstisteganografiaa hyödynnetään pikaviestinnässä tai sosiaalisessa mediassa, epätäydellisyys on sallittua. Jos sitä hyödynnetään julkisvirallisissa materiaaleissa, steganografiselta materiaaliilta edellytetty eheys muuttuu täysin, samoin kuin materiaaliin kohdistuva kriittinen tarkastelu.

Etenkin muotoon perustuva tekstisteganografia on altis havainnoinnille. Ylimääräiset välilyönnit tai erilaisten fonttien käyttö tai muu poikkeava asettelu näkyvät herkästi tekstinkäsittelyohjelmassa, kuten Agarwal toteaa (2013, s. 92), tosin Agarwal ei perustele havainnoinnin vahingollisuutta. Tekstinkäsittelyohjelmissa on arkihavaintojen perusteella enemmän poikkeus kuin sääntö nähdä toivotusta formaatista poikkeavaa asettelua tai fonttien käyttöä eri lähteistä siirretyn tekstin leikkaamisen ja liimaamisen (*copy-paste*) seurauksena, eikä tällaista havainnointia voi itsessään pitää indikaationa steganografiasta toiminnasta.

Vaille huomiota jää myös analyysi siitä, kuinka halukas ulkopuolinen tarkkailija on todella koskemaan tuntemattomiin tiedostoihin, jotka voivat ilman steganografista viestintääkin sisältää monenlaisia haittaohjelmia. Tilanne luonnollisesti muuttuu, mikäli salatun viestinnän käytöstä on jo valmiiksi epäilyksiä: silloin tavanomainen tuntemattomien tiedostojen tietoturvallinen käyttö oletettavasti jää taka-alalle. Jos tarkempi pohdinta tiedostojen sisällöstä sivuutetaan, mainittu muotoon perustuva steganografia on havaittavissa paitsi tekstinkäsittelyohjelman tyyl- ja muotomäärityksiä tarkkailemalla myös yleisesti tekstintunnistusta (OCR) hyödyntämällä. Vaarana tunnistetuksi tulemisen lisäksi on myös salaviestin tuhoutuminen. (Agarwal 2013, s. 93)

Tässä on tutustuttu steganografian eri lajeihin ja toteutustapoihin, etenkin tekstisteganografiaan ja sen erikoisalueena lingvistiseen steganografiaan, joka pyrkii säilyttämään viestinnän kielellisen merkityksen piiloviestin välittämisestä huolimatta. Seuraavassa tutustutaan etenkin suomalaisessa armeijassa merkitykselliseen peittämiseen, joka toteutetaan sanankorvauksena. Samalla tutustutaan suomen kielen ominaispiirteisiin, jotka tekevät sanankorvauksesta monin verroin haastavampaa kuin mitä se olisi vaikkapa englanninkielisessä kontekstissa. Samalla esitetään diplomityössä toteutettu ohjelma, jolla suomenkielinen sanankorvaus toteutetaan. Hieman sivutun ja paljon vihjaillun tekoälyn aika tulee myöhemmissä luvuissa.

3. PEITTÄMINEN OHJELMALLISESTI

Tässä luvussa käsitellään diplomityössä ohjelmallisesti toteutettua sanankorvausta, jossa sovelluksen avulla tuotetaan selkotekstin kanssa kieliopillisesti yhtenevää peite tekstiä, joka puolestaan puretaan takaisin selkotekstiksi. Tällainen sanankorvaus on kielien puhujalle luontevaa ja yksinkertaista, mutta saman toiminnallisuuden toteuttaminen ohjelmallisesti on haastavaa. Luvussa käsitellään ohjelmoinnissa tehtyjä valintoja ja arvioidaan lopullisen ohjelman toimintaa. Valmista ohjelmaa voidaan käyttää sanankorvausta hyödyntävään suomenkielisen tekstin peittämiseen sellaisenaan, ja toisaalta sillä voidaan luoda lähdeaineistoa suuren kielimallin opettamiseen.

3.1 Peittäminen sanankorvauksella

Yksinkertainen mutta toimiva tapa peittää viestintää on sanankorvaus. Tällöin viestissä salattavat sanat korvataan toisilla salaustaulukon mukaisesti, jolloin lopullinen viesti on kieliopillisesti samanlainen mutta merkitykseltään erilainen. Peittämisessä salaustaulukona käytetään sanankorvaustaulukkoa: listaa salattavista sanoista ja sanoista, joita niiden tilalla käytetään. Sanankorvaustaulukko on toisin sanoen viestinnän osapuolten keskenään jakama koodikirja. Sanankorvauksessa viestinnän olemassaolon salaaminen ei ole niinkään olennaista, mutta viestinnän sisältö voidaan kätkeä tehokkaasti. Havainnollistetaan sanankorvauksen toimintaa yksinkertaisella esimerkillä, jossa käytetään taulukossa 4 esitettyä sanankorvaustaulukkoa.

Taulukko 4 Sanankorvaustaulukko.

Selkosana	Peitesana
ajoneuvo	käpylintu
drooni	hippiäinen
kenttäsairaala	kanto
sotilas	käki
tukikohta	juurakko
tuntematon	tulipunainen

Viesti ”tuntematon ajoneuvo matkalla kenttäsairaalaan, lähettäkää droonit” olisi peitetty ”tulipunainen käpylintu matkalla kantaan, lähettäkää hippiaiset”. Tällaisesta viestistä voi edelleen tunnistaa, että jotain on menossa jonnekin, mutta sisältö ei yksittäisestä viestistä ole silti pääteltävissä. Sopivasti muotoiltuna ja sopivassa kanavassa viestittynä tuollainen viesti voisi yksinkertaisesti olla lintubongareiden havaintojen raportointia.

Sanankorvaus ei ole murtovarma peittämiskeino. Peitettyä viestintää kylliksi havainnoidulla tällainen salaus on luonnollisesti mahdollista murtaa. Helppo keino salauksen parantamiseksi on useiden peitesanojen antaminen samalle selkosanalle, ja luotettujen viestijöiden keskenään sopima järjestys siitä, milloin mitäkin salausta käytetään. Esimerkki tällaisesta taulukosta on annettu taulukossa 5.

Taulukko 5 Sanankorvaustaulukko, jossa selkosanaa vastaa usea peitesana.

Selkosana	Peitesana 1	Peitesana 2	Peitesana 3
ajoneuvo	käpylintu	hippiäinen	käki
drooni	hippiäinen	juurakko	kanto
kenttäsairaala	kanto	kanto	käpylintu
sotilas	käki	käki	hippiäinen
tukikohta	juurakko	käpylintu	juurakko
tuntematon	tulipunainen	taivaansininen	ruohonvihreä

Tällaisessa salauksessa voidaan käyttää samoja peitesanoja, kuten taulukon 5 substantiiveilla, mutta niihin yhdistyvät selkosanat vaihtuvat ulkopuoliselle lähettäjälle tuntemattomalla tavalla. Peitesanat voivat myös vaihdella, kuten taulukon adjektiiveilla. Aiempi viesti ”tuntematon ajoneuvo matkalla kenttäsairaalaan, lähettäkää droonit” voisi nyt olla ”taivaansininen hippiäinen matkalla kantaan, lähettäkää juurakot” tai ”ruohonvihreä käki matkalla käpylintuun, lähettäkää kannot”. Koska näissä salausmuodoissa muuttui ainoastaan käytetty peitesana, viesteistä on edelleen pääteltävissä, että jotain on menossa jonnekin, kuten ensimmäisestäkin salauksesta. Kuitenkin salauksen murtaminen on vaikeutunut jo tässäkin yksinkertaisessa esimerkissä. Peitesanaston laatiminen ja peittämisen turvallinen toteuttaminen eivät kuitenkaan kuulu tämän tutkielman käsittelyyn.

Sanankorvaus on yleisesti käytössä armeijan radioviestinnässä. Peittäminen sanankorvaustaulukoita hyödyntämällä on ihmiskäyttäjälle luontevaa, mutta tilanne muuttuukin, kun sama toiminnallisuus halutaan toteuttaa ohjelmallisesti. Ongelman avaamiseksi on syytä ensin tarkastella suomen kielen erityispiirteitä ja rakenteita.

3.2 Suomen kielen lyhyt kielioppi ja ominaispiirteitä

Suomen kielessä on lukuisia leimallisia erityispiirteitä, joiden huomioiminen on välttämättöntä ja toisaalta ohjelmallisesti haastavaa, kun halutaan toteuttaa luonnollista kieltä tuottava ohjelmisto. Monikollisuuden ja taivutuksen ohella kirjoitettua suomen kieltä hallitsee astevaihtelu.

Kirjoitetussa suomen kielessä tärkeimmät näistä ovat **astevaihtelu**, **taivutus** (deklinaatio tai konjugaatio) ja **monikollisuus**, jotka kukin voivat vaikuttaa sanan lopulliseen muotoon yhtäaikaaisesti. Verbeillä on lisäksi huomioitava **aikamuoto** (tempus) ja **tapaluokka**

(modus), adjektiiveilla puolestaan **vertailuluokka** (komparaatio). Koska valittu ohjelmallinen lähestymistapa on sanankorvausta – sana korvataan eli peitetään toisella saman sanaluokan sanalla, joka on taivutettu samaan sijaan – sellaisista seikoista kuten rektiosta (sanojen taipuminen määrätysssä sijassa), kongruenssista (predikaatin taipuminen subjektin mukaan) tai oikeakielisyydestä ei huolehdita ohjelmallisesti, vaan näistä kielilooppiseikoista huolehtiminen on yksinomaan käyttäjän vastuulla. Tarkastellaan mainittuja piirteitä lyhyesti. Luettelomaisen esitystavan on tarkoitus korostaa sitä, kuinka yksittäisellä sanalla voi olla kymmeniäkin taivutusmuotoja, joita ei voi toteuttaa yksioikaisesti huomioimatta sanaan samanaikaisesti vaikuttavia muitakin erityspiirteitä.

Nomineilla (substantiivi, adjektiivi, numeraali, pronomini) on suomen kielessä 15 **sijamuotoa**, jotka harvinaisimpia lukuun ottamatta esiintyvät sekä yksikössä että monikossa. Nämä sijamuodot ovat nominatiivi, genetiivi, partitiivi, essiivi, translatiivi, inessiivi, elatiivi, illatiivi, adessiivi, ablatiivi ja allatiivi. Näiden lisäksi esiintyy useimmiten nominatiivi- tai genetiivimuotoinen akkusatiivi ja harvinaisemmat abessiivi, komitatiivi ja instruktiivi. (Hakulinen et al. 2004, s. 1174–1175)

Verbit taipuvat aktiivissa persoonan (yksikön ja monikon 1.–3. persoona) mukaan tai passiivissa. Verbeillä on lisäksi **aikamuoto** eli tempus (preesens, imperfekti, perfekti, pluskvamperfekti) ja **tapaluokka** eli modus (indikatiivi, potentiaali, konditionaali, imperatiivi). (Hakulinen et al. 2004, s. 134, 437, 1452, 1510) Sekä verbien että nominien sijapäätteisiin vaikuttaa sanan vokaalien koostumus eli vaikkapa nominin inessiivin päätte voi olla *-lla* tai *-llä* (merkitään yhteisesti *-//A*) riippuen siitä, sisältääkö sana tai yhdysverbin loppuosa pelkästään etuvokaaleja *i, e, y, ä, ö* vai myös takavokaaleja *a, o, u* (ibid., s. 37).

Kuitenkaan suomen sanojen taivutus ei ole yksioikaisesti sijapäätteiden lisäämistä sanavartaloon ja tätä varten oikean sanavartalon tunnistamista. Sanojen taipumista oikean sijamuodon tai aikamuodon lisäksi säätelevät myös sanan taivutustyyppi ja astevaihtelu.

Taivutustyyppi tarkoittaa kokoelmaa sanan taipumisen ominaisuuksia, jotka yhdistävät samaan taivutustyyppiin kuuluvia sanoja. Kotimaisten kielten keskus määrittelee suomen kielelle taivutustyyppit 1–78 ja 99, yhteensä 79 muotoa. (Kotus 2024b) Tästä huolimatta taivutustyyppin toimintaa tai reunaehtoja ei ole Kotuksen jaottelussa juurikaan määriteltä, vaan kyseessä on enemmänkin kielenkäyttäjän sisäsyntyisesti ymmärtämä säännöstö ja jaottelu. Tämä jaottelu toimii ihmiskäyttäjälle varsin ymmärrettävästi ja riittävällä tarkkuudella, mutta saman toiminnallisuuden toteuttaminen ohjelmallisesti edellyttää aivan erilaista järjestelmällisyyttä. Taivutustyyppien esiintymien määrää tarkastelemalla ohjelman toteutukseen voitiin tehdä tarkoituksenmukaisia valintoja, kun toteutus keskittyi

yleisimpiin taivutustyyppihin. Näihin palataan myöhemmin luvussa 3.6. Taivutustyyppin lisäksi sanan rakenteeseen vaikuttaa oleellisesti sanan mahdollinen astevaihtelu.

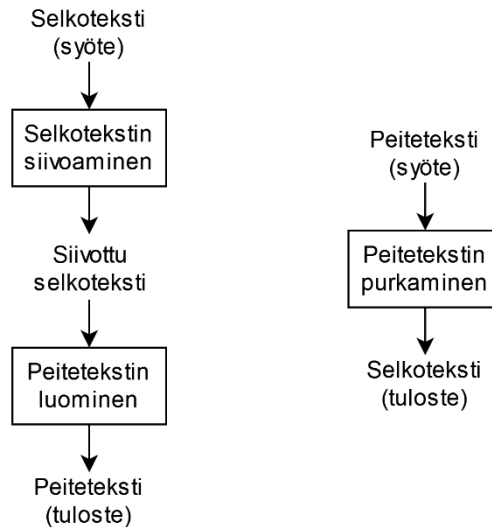
Astevaihtelu tarkoittaa sanansisäistä konsonantin kahdentumista tai yksinkertaistumista (pakata -> pakkaan, takki -> takin) tai konsonantin vaihtumista (luuta -> luudan) sanan eri taivutusmuodoissa. Astevaihtelu voi olla vaikuttaa myös useampaan konsonantiin tai vokaaliin samanaikaisesti (po/ka -> pojan) tai vaihdella eri sijamuotojen välillä huomattavasti (käsi -> käden -> kättä -> käsien/kätten, kynsi -> kynnen -> kynttä -> kynsien/kyntten). Kotimaisten kielten keskus määrittelee astevaihtelulle tyytit A–M, yhteensä 13 muotoa. (Kotus 2024a)

Samalla sanalla voi olla käytössä useita hyväksyttäviä taivutusmuotoja, joiden esiintyminen vaihtelee käyttäjän mieltymyksen tai totumuksen mukaan (vanhempien – vanhempain, omenoiden – omenoitten – omenien). Lisäksi sanat voivat olla toistensa homonyymejä: perusmuodoltaan samanlaisia mutta erimerkityksisiä sanoja, kuten kuusi (puu) ja kuusi (luku). Suomen kielen homonyymit voivat taipua samalla tavalla tai kuulua kokonaan eri taivutustyyppihin, kuten aika (substantiivi) ja aika (taipumaton adjektiivi). Homonyymien lukumäärä on kokonaisuudessaan vähäinen, vain alle 0,9 % suomen kielen sanalistaan (*kotus.xml*) sanoista, ja suurin osa sanalistaan homonyymeistä kuuluu samaan taivutustyyppiin, joten näiden olemassaoloa ei ole ollut tarpeen erikseen huomioida. Ohjelman toteutuksessa tehtyjä valintoja käsitellään tarkemmin luvussa 3.4.

Seuraavassa käsitellään laaditun peittämishjelman toimintaa ja ulkopuolisia komponentteja, joita ohjelman toteutus hyödyntää.

3.3 Ohjelman toiminta vaiheittain

Osana tutkimusta koodattiin Python-kielinen peittämishjelma, jolle annetaan syötteenä selkokielineen, todellisen merkityksen sisältävä lause. Ennen peittämistä ohjelma siistii syötteen: välimerkit ja erikoismerkit poistetaan, ja isot kirjaimet korvataan pienillä. Näin saadusta selkokielisestä lauseesta tuotetaan selko- ja peitesanat sisältävän sanaston avulla peitelause. Purkamisvaiheessa prosessi käännetään toiseen suuntaan: peitelause puretaan käännteissanaston avulla takaisin selkolauseeksi. Koska peitelause on tehty siistitystä syötteestä, peitelauseetta ei ennen purkamista tarvitse siistiä. Ohjelman toiminta on esitetty kuvassa 1.



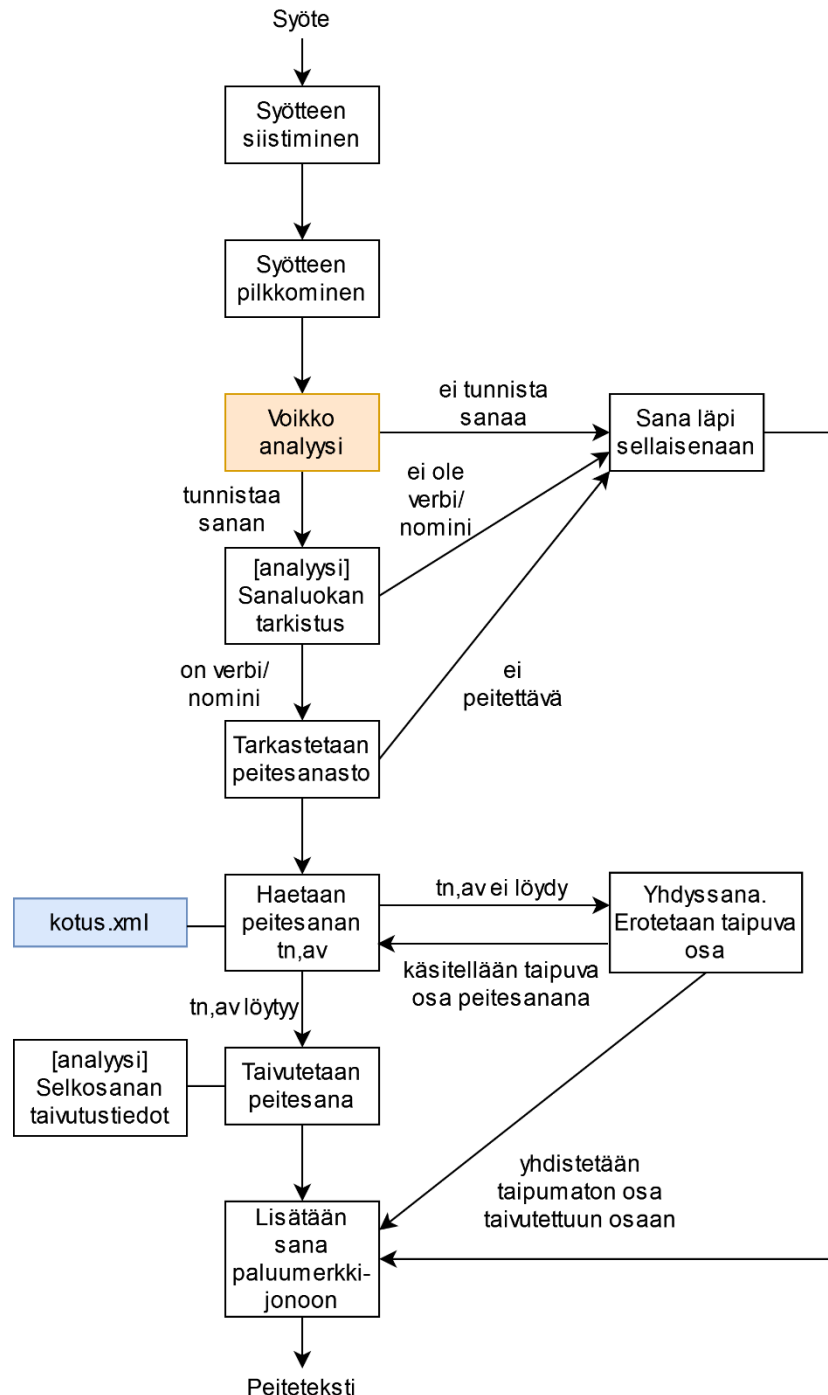
Kuva 6. Peittämis- ja purkamisprosessin vaiheet. Peittäminen kuvattu vasemmalla ja purkaminen oikealla.

Ohjelmallisessa sanankorvauksessa tärkeimmät ulkopuoliset osat ovat ilmainen, avoimen lähdekoodin sanantunnistusohjelmisto Voikko ja Kotimaisten kielten keskuksen (Kotus) tuottama lista (*kotus.xml*) suomen kielen sanoista taivutusmuotoineen ja astevaihteluineen, joita ohjelma hyödyntää peittämis- ja purkamisprosessin eri vaiheissa. Voikko ja Kotuksen sanasto tarjoavat analyysin selko- ja peitesanan perusmuodosta ja taivutetun sanan kieliopillisista ominaisuuksista, joiden mukaan ohjelmisto taivuttaa peitesanan samaan kieliopilliseen sijaan.

Peittämisprosessi toimii siten, että siistimisen jälkeen syötettä käsitellään sana (merkkijono) kerrallaan. Kukin syötteen sana analysoidaan yksitellen Voikon *analyze*-funktion avulla. Peittäminen tehdään ainoastaan nomineille (tarkemmin ottaen substantiiveille ja positiivimuotoisille adjektiiveille) ja verbeille, joten muun muassa partikkelit, konjunktiot ja pronominit ovat automaattisesti tarkastelun ulkopuolella. Tällaisiin sanaluokkiin kuuluvat sanat, kuten myös sanat, joita Voikko ei tunnista ollenkaan, lisätään sellaisenaan ohjelman palauttamaan merkkijonoon. Samoin tehdään sanoille, joiden perusmuotoa ei löydy peitesanastosta ja joita ei siten ole tarkoituskaan peittää.

Kun selkosana on tunnistettu peitettäväksi, vastaavan peitesanan taivutustyyppi (tn) ja astevaihtelu (av) haetaan Kotuksen sanalistan tiedoista. Mikäli taivutustyyppiä ei löydy, oletetaan, että sana on yhdyssana, jolloin ohjelma Voikon *analyze*-funktion tietoja käyttäen erottelee yhdyssanan loppuosan eli perussanan, jonka mukaan taivutus tapahtuu. Esimerkiksi yhdyssana ”yhdyssana” taipuu perussanan ”sana” mukaan, ja vain tälle itsenäiselle osalle *kotus.xml* sisältää tarkat tiedot. Ohjelma kutsuu omaa taivutusfunktiota joko substantiiveille tai verbeille käyttäen kutsussa tietoa muun muassa peitesanan taivutustyyppistä, astevaihtelusta ja selkosanan ominaisuuksia, kuten aikamuotoa, sijaa ja

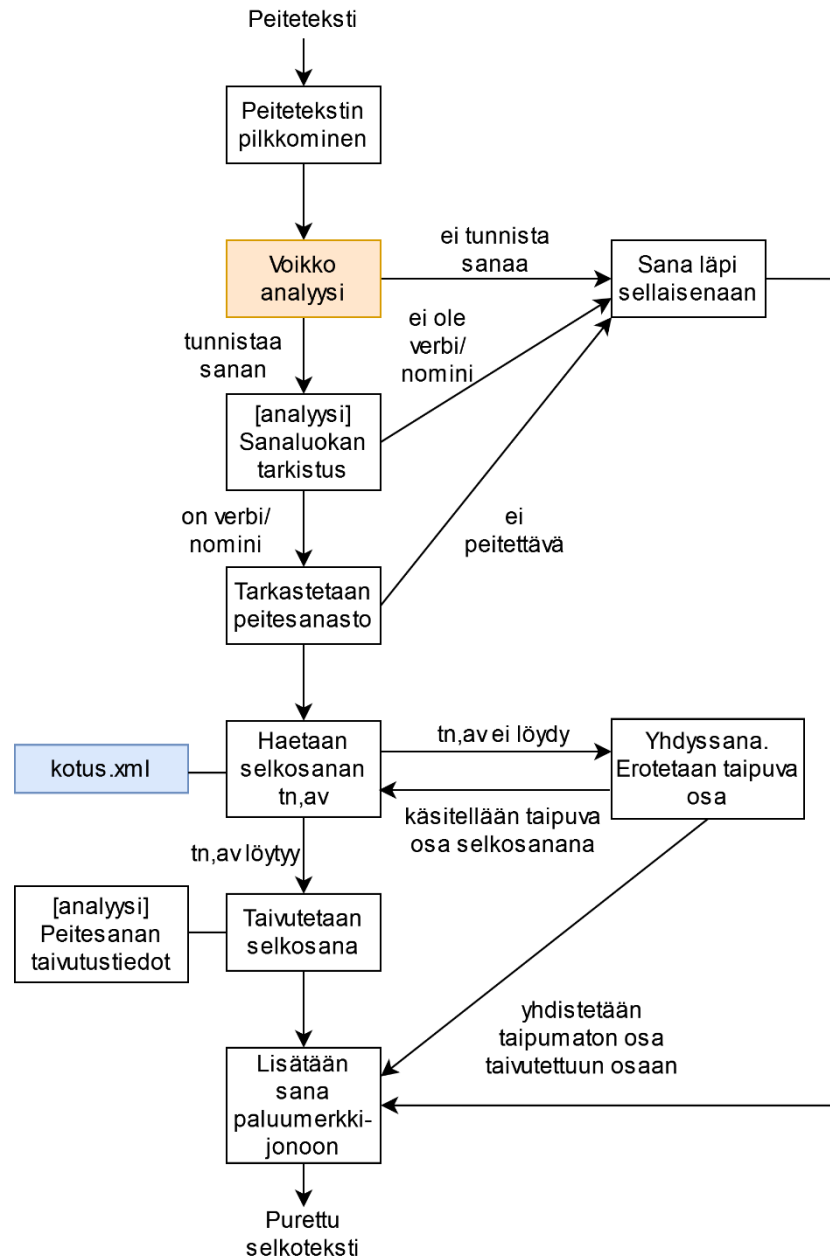
monikollisuutta. Saatu peitesana lisätään tämän jälkeen paluumerkkijonoon. Jos peitesana on harvinaista sijamuotoa, jota ohjelma ei osaa taivuttaa, lisätään paluumerkkijonoon pelkästään perusmuotoinen peitesana. Toteutuksen valintoja on käsitelty tarkemmin kappaleessa 3.4. Tässä kuvattu peittämisprosessi on esitetty kuvassa 7.



Kuva 7. Selkotekstin muuntaminen peitetekstiksi.

Peitetekstin purkaminen takaisin selkotekstiksi on yksinkertaisuudessaan käänteinen peittämisprosessi. Peiteteksti analysoidaan Voikon *analyze*-funktion avulla sana sanalta, ja käänteisestä peitesanastosta tarkastetaan, kuuluuko sana peitesanojen joukkoon.

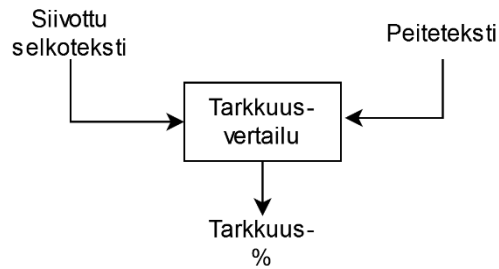
Peitetekstin kielipiilliset, Voikon avulla selvitetty ominaisuudet yhdistetään selkosanan *kotus.xml*-tiedostosta saatuun taivutustyyppiin ja astevaihteluun, ja selkosana taivutetaan haluttuun kielipiilliseen sijaan verbin ja substantiivin taivutusfunktioita käyttäen. Myös purkamisvaiheessa ja lopullisessa sanantaivutuksessa yhdyssanat käsitellään erottelemalla ainoana taipuva perussana yhdyssanan alkuosasta, kunnes lopulta saadaan purettu selkoteksti paluumerkkijonona. Purkamisprosessi on esitetty kuvassa 8.



Kuva 8. Peitetekstin purkaminen takaisin selkotekstiksi.

Lisäksi ohjelmaan on toteutettu toiminnallisuus, joka vertailee siistittyä selkotekstiä ja purkamisessa saatua selkotekstiä toisiinsa merkkijono kerrallaan. Vertailu tuottaa prosentteina ilmaistun kokonaislukuarvon, joka kertoo, kuinka hyvin purettu peiteteksti vastaa alkuperäistä selkotekstiä. Merkkijonokohtaisella vertailulla saadaan numeerinen

arvio peiteprosessin tarkkuudesta, jota voidaan hyödyntää ohjelman testaamisessa. Tarkkuuden arviointia on havainnollistettu kuvassa 9.



Kuva 9. *Puretun selkotekstin ja alkuperäisen selkotekstin tarkkuusvertailu.*

Ohjelman suunnittelussa ja toteutuksessa tehtyjä valintoja käsitellään seuraavassa.

3.4 Ohjelman toteutuksessa tehdyt valinnat

Jotta ohjelman toteutus on pysynyt järkevässä laajuudessa, toteutuksen suunnittelussa ja itse toteutuksessa on tehty priorisointia. Kuten kappaleessa 3.1 on esitetty, suomen kielen sanoille tunnusomaista taipumista säätelee suuri määrä samanaikaisesti vaikuttavia tekijöitä, joiden huomioiminen on kielen käyttäjälle melko luontevaa mutta luotettava tietotekninen toteuttaminen vaativaa. Ohjelman käyttökontekstiksi on ensisijaisesti tarkoitettu sotilaallinen viestintä: tämän mukaisesti toteutus on keskitetty sellaisiin taivutusmuotoihin ja tapaluokkiin, jotka arvioitiin taktisessa viestinnässä yleisiksi. Mikäli ohjelmaa myöhemmin laajennetaan ja kehitetään edelleen, voidaan mukaan tarvittaessa sisällyttää harvinaisempia taivutusmuotoja tai sanaluokkia. Nyt tehdyt valinnat kattavien ydintoiminnallisuuksien toteuttamiseksi käsitellään seuraavassa.

3.4.1 Käyttötarkoituksen mukaiset valinnat

Käyttötarkoituksen mukaisia valintoja olivat päätökset toteutettavista taivutusluokista ja verbien tapaluokista. Suomen kielessä on substantiiveille 14 itsenäistä sijamuotoa, jollei nominatiivi- tai genetiivimuotoista akkusatiivia huomioida (Hakulinen et al. 2004, s. 1174–1175). Harvinaisia sijamuotoja lukuun ottamatta kukin taipuu eriävästi yksikössä ja monikossa. Toteutus huomioi näistä yksiköllisenä ja monikollisena 11 ja jättää huomiotta harvinaiset abessiivin (aseitta), komitatiivin (teitse) ja instruktiivin (sotilaineen). Näiden merkitys on helppo ja nykykielessä luontevaakin ilmaista yksinkertaisemmillä rakenteilla (ilman aseita, tietä/teitä pitkin, sotilaiden kanssa).

Verbien aikamuodoista huomioitiin preesensin ja imperfektin ohessa perfektin ja pluskvamperfektin muodostamiseen tarvittava II partisiippi (tehnyt, laulanut), mutta ei I partisiippia (laulava) tai agenttipartisiippia (laulama), jotka osin esiintyvät myös

itsenäisinä sanakirjasanoina Kotuksen suomen kielen sanalistassa. Verbeistä huomioitiin indikatiivi ja kieltomuodon ohella imperatiivi eli käskymuoto mutta ei harvinaisempaa ja varsinkin sotilaskäyttöön vierasta potentiaalia (tehnee, saanee) tai konditionaalia (tekisi, ehtisi). Adjektiivien komparaatiota ei toteutettu, sillä komparatiivin ja superlatiivin toteuttaminen positiivin oheen kaikille sijamuodoille olisi tehnyt toteutuksesta kohtuuttoman laajan. Myös omistusliite (possessiivisuffiksi) jätettiin ensimmäisen toteutuksen ulkopuolelle.

Toteutuksessa oletetaan, että peitesana ei voi esiintyä selkosanana. Lisäksi oletetaan, että Voikko tunnistaa käytetyn peitesanan oikein ja kyseinen sana tai peitesanana toimivan yhdyssanan taipuva osa löytyvät *kotus.xml*-tiedostosta. Selkosanoille ja siten käyttäjän viestien sisällölle ei ole rajoituksia.

3.4.2 Ohjelmallisen toteutuksen valinnat

Käyttäjän syötteitä ei rajoiteta tai oikolueta, lukuun ottamatta erikoismerkkien poistoa syötteestä. Mikäli haluttaisiin, että käyttäjäsovelluksissa yleinen ennakoiva tekstinsyöttö otettaisiin käyttöön myös peittämisohjelmassa, edellyttäisi tämä kokonaan uusien ohjelmistokomponenttien lisäämistä. Tässä vaiheessa tällaisen toteutuksen edellytyksiä tai mahdollista toteutusta ei ole tarkasteltu. Tehtyä päätöstä luultavasti tarkasteltaisiin uudelleen, mikäli ohjelmaa päätetään kehittää jatkossa, ja samalla toteutettaisiin ohjelmalle graafinen käyttöliittymä. Samalla mahdollisesti haluttaisiin muutoksia nykyiseen sana sanalta toimivaan peittämislogiikkaan, joka nykyisellään ei mahdollista puhtaiden erisnimien tai useasta sanasta koostuvien sanaliittojen peittämistä. Samoin välimerkkien ja isojen kirjainten poistamista tarkasteltaneen uudelleen.

Liitepartikkeleita (-pA, -hAn) ei huomioitu, sillä Voikon *analyze*-funktio ei tunnista niiden olemassaoloa. Liitepartikkelin sisältävä selkosana käsitellään peittämisessä ja peittämisen purkamisessa kuin sanassa ei olisi kyseistä loppuliitettä. Tätä ei kuitenkaan voi pitää merkittävänä puutteena, sillä liitepartikkelien käyttö ei sanottavasti sisällä informaatiota, vaan tuo viestintään lähinnä sävyeroja, jopa tahatonta monitulkintaisuutta (Kielitoimiston ohjepankki 2023).

3.5 Ohjelman toiminta

Sanojen taivutuksen perustana on käytetty Kotuksen julkaisemaa listaa suomen kielen sanoista (*kotus.xml*). Tässä listassa on noin 96 000 sanaa, joista perussanoja on noin 39 000. Näistä toteutus huomioi noin 20 000 kappaletta ja joita näin ollen voidaan käyttää peitesanoina siten, että ohjelma osaa taivuttaa niitä oikein. Tähän lukuun sisältyy

myös jonkin verran sanalistassa olleita yhdyssanoja tai liitteitä. Käytettävissä olevien sanojen lukumäärää pienentää se, että osaa sanalistan sanoista ei Voikon *analyze*-funktio tunnista (esimerkiksi koirarotu "bernhardilainen" tai sen taivutusmuodot), joten lopullisten peitesanojen toimivuus on luonnollisesti tarkistettava ennen käyttöönottoa. Toisaalta ylivoimaisesti suurinta osaa suomen kielen yhdyssanoista ei ole sisällytetty tähän 20 000 sanan joukkoon. Jokerihaku osoitteessa www.sanapu.org tuottaa 84 -kivi-päätteistä yhdyssanaa, joita kaikkia voitaneen hyödyntää peitesanoina, sillä sanan "kivi" taivutusluokka on huomioitu toteutuksessa. Näin ollen todellista peitesanojen määrää on mahdotonta arvioida, mutta se voidaan todeta vähintäänkin riittäväksi.

Selkosanoille ei ole asetettu rajoituksia. Peittäminen oikeakielisesti onnistuu, sillä peitesanojen lista on rajattu, joskin erittäin laaja, mutta viestin purkamisen oikeakielisesti riippuu siitä, onko selkosanojen taivutusluokkia huomioitu. Peitesanojen tulkitseminen perusmuotoisiksi, taivuttamattomiksi selkosanoiksi onnistuu, mutta kieliopin mukainen taivutus ei välttämättä ole mahdollista.

Jos puretun sanan eli käyttäjän alun perin syötteenä antaman sanan taivutusta ei ole sisällytetty ohjelmaan, ohjelman ensimmäinen toteutus mahdollistaa puretun selkosanan esittämiseen kolme vaihtoehtoa:

- Vaihtoehto 1: Selkosana esitetään sellaisenaan perusmuodossa (nominatiivi).
- Vaihtoehto 2: Selkosanalle käytetään kullekin sijalle tyypillistä taivutusmuotoa, joka liitetään perusmuotoiseen sanaan kaksoispisteellä. Lisäksi sana merkitään kaarisulkeilla, (). Tätä vaihtoehtoa käytetään vain substantiiveille, joilla sijapäätteet ovat tunnistettavia, toisistaan erottuvia ja oletettavasti luonnollisen kielenkäyttäjän ymmärrettävissä. Esimerkiksi "sanasta" merkitään (sana:sta).
- Vaihtoehto 3: Selkosanalle käytetään kieliopillista taivutusta, jossa perusmuotoiseen sanaan liitetään kieliopillisia määreitä plusmerkillä, +. Lisäksi sana merkitään kaarisulkeilla, (). Tätä vaihtoehtoa käytetään sekä nomineille että verbeille. Liitetty kieliopillinen analyysi ei ole täydellinen. Esimerkiksi "kenttätykistö" merkitään (kenttätykistö+MON+INESS) ja "tiedustelivat" merkitään (tiedustella+MENNYT+MON.3.PERS.), kun täydellinen kieliopillinen kuvaus sanalle olisi "aktiivin indikatiivin monikon kolmannen persoonan imperfekti".

Seuraavassa on esimerkki viestistä, jossa selkosanat kuuluvat ohjelmaan sisältyneisiin taivutusluokkiin. Alkuperäinen ja purettu viesti ovat yhtenevät.

Selkoteksti: baretit havaittu sadan metrin päästä haupitseista hal-
lista etelään

Saatu peiteteksti: hyasintit havaittu sadan metrin päästä pyökeistä hon-
gasta etelään

Purettu selkoteksti: baretit havaittu sadan metrin päästä haupitseista hal-
lista etelään

Selkoteksti ja purettu teksti olivat yhtäläisiä 100 % tarkkuudella.

Annettu viesti on esimerkinomainen. Sotilasviestinnässä suunnat ilmastaan koordinaat-
tien avulla (Taistelijan opas 2013, s. 211), ja käytetty peitteistö vaihtelee tilannekohtai-
sesti (tulenjohto, tiedustelutiedon välittäminen, tilanneviestintä). Numeroita, aakkosia ja
erisnimiä, kuten osastojen tai yksikköjen nimiä, sisältävien ilmausten välittämiseen on
omat sääntönsä (Sotilaan käsikirja 2019, s. 218).

Seuraavassa on esimerkkejä viesteistä, joissa viestin selkosanat eivät kuulu ohjelmaan
sisältyneisiin taivutusluokkiin. Ensimmäisessä tapauksessa peitettyt sanat on viestiä pu-
rettaessa annettu perusmuodossa (vaihtoehto 1).

Selkoteksti: ilmatorjuntavaunut ovat tappaneet ja sotilaat ovat te-
loittaneet vaarallisia vihollisia

Saatu peiteteksti: hallitsijaparit ovat eksyttäneet ja kobrat ovat alis-
taneet tynkiä gepardeja

Purettu selkoteksti: ilmatorjuntavaunu ovat tappaa ja sotilas ovat teloit-
taneet vaarallisia vihollisia

Selkoteksti ja purettu teksti olivat yhtäläisiä 66 % tarkkuudella

Toisessa tapauksessa sanoille, joiden taivutusta ei ole sisällytetty ohjelmaan, on lisätty
tyypillinen sijapäätte (substantiivit) tai annettu kielipillinen kuvaus (verbit) (vaihtoehto 2).

Selkoteksti: ilmatorjuntavaunut ovat tappaneet ja sotilaat ovat
teloittaneet vaarallisia vihollisia

Saatu peiteteksti: hallitsijaparit ovat eksyttäneet ja kobrat ovat
alistaneet tynkiä gepardeja

Purettu selkoteksti: (ilmatorjuntavaunu:t) ovat (tappaa+PERF.) ja
(sotilas:t) ovat teloittaneet vaarallisia vihollisia

Selkoteksti ja purettu teksti olivat yhtäläisiä 66 % tarkkuudella.

Kolmannessa tapauksessa kaikille sanoille, joiden taivutusta ei ole sisällytetty
ohjelmaan, on annettu kielipillinen kuvaus (vaihtoehto 3).

Selkoteksti: ilmatorjuntavaunut ovat tappaneet ja sotilaat ovat
teloittaneet vaarallisia vihollisia

Saatu peiteteksti: hallitsijaparit ovat eksyttäneet ja kobrat ovat
alistaneet tynkiä gepardeja

Purettu selkoteksti: (ilmatorjuntavaunu+MON.NOM.) ovat (tappaa+PERF.) ja
(sotilas+MON.NOM.) ovat teloittaneet vaarallisia
vihollisia

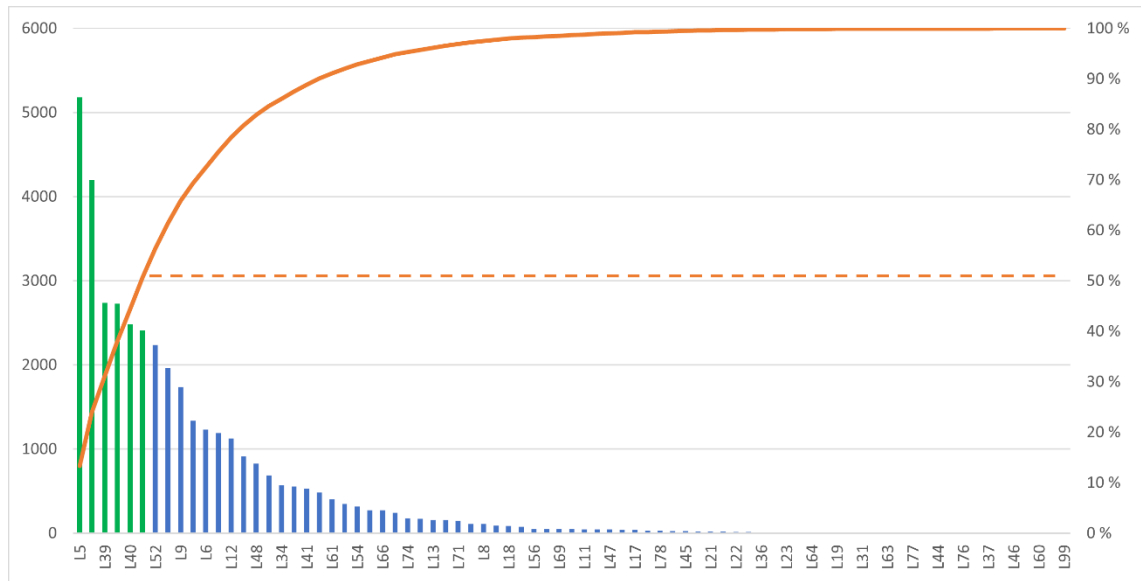
Selkoteksti ja purettu teksti olivat yhtäläisiä 66 % tarkkuudella.

3.6 Ohjelman toiminnan arviointi

Ohjelman perustana on käytetty Kotuksen listaa suomen kielen sanoista (*kotus.xml*). Sittemmin Kotus on julkaissut uuden version sanalistaista, Nykysuomen sanalistan (*nykysuomensanalista2022.csv*, päivitetty 20.4.2023) (Nykysuomen sanalista 2023), johon on lisätty uudissanastoa, kuten ”drooni”, ja yhdyssanoja. Sanojen lukumäärä on yli 104 000. Yksittäisiä sanoja lukuun ottamatta ei ole varmaa, olisiko uuden listan käyttöön ottamisella saavutettavissa merkittävää hyötyä, sillä lisäyksistä suurin osa koostuu yhdyssanoista, joiden käsittely ja taivutus perustuu joka tapauksessa yhdyssanan jälkimmäiseen osaan eli perussanaan, kuten luvussa 3.3 on esitetty. Lisäksi yhdyssanoja voidaan suomen kielessä muodostaa lähes rajattomasti (Lehtinen 2013), ja päivitettykin sanalista kattaa arkipäiväisistä sanoista vain pienen osan.

Näin ollen peiteratkaisujen kattavuutta arvioidessa hyödyllisintä on tarkastella perussanoja – toisin sanoen itsenäisiä sanoja, jotka eivät ole yhdyssanoja – joita *kotus.xml*-lista sisälsi noin 39 000 kappaletta. Taivutusluokkia on Kotuksen luokittelun mukaan yhteensä 79 (luokat 1–78 ja 99). Ohjelmaan valittiin yleisimmät taivutusluokat, joiden yhteenlaskettu esiintymien lukumäärä oli tästä määrästä 51 %.

Ohjelman mahdollista jatkokehitystä suunnitellessa huomionarvoista on se, että suhteellisen pieni määrä luokkia, 14 kappaletta, kattaa yli 80 % perussanojen lukumäärästä. Peräti 18 taivutusluokkaa sisältää alle 5 esiintymää eli taivutusluokkaan kuuluvaa perussanaa. Ohjelman käyttötarkoitus laaditun prototyypin ulkopuolella määrittää lopulta, mitkä luokat ohjelman mahdollisissa tulevilla versioissa toteutukseen halutaan sisällyttää. Käyttötarkoituksen mukaan voi olla hyvin mielekästä sisällyttää toteutukseen sanat ”vasen” ja ”mies” ainoina esiintyminään sisältävät taivutusluokat 37 ja 42. Nykyisen toteutuksen voidaan arvioida kuitenkin olevan lukumääräisesti hyvin kattava. Luokkiin kuuluvien sanojen taivutuksessa on luonnollisesti huomioitu myös astevaihtelu, jota edustavat astevaihtelutyypit A–M. Kuva 10 havainnollistaa taivutusluokkien yleisyyttä ja pienen esiintymämäärän sisältävien luokkien suhteessa korkeaa määrää, mikä tekee kaikkea suomen kieltä taivuttavasta ohjelmasta haasteellisen ja erittäin työlään toteuttaa.



Kuva 10. Ohjelmaan sisällytettyjen taivutusluokkien kattavuus kaikista suomen kielen sanoista. Kunkin taivutusluokan esiintymien lukumäärä on esitetty pystypalkeilla ja vasemmanpuoleisella asteikolla. Esiintymien kumulatiivinen summa prosentiosuutena kokonaisuudesta on esitetty oranssilla viivalla ja oikeanpuoleisella asteikolla. Vihreät pylväät ja oranssi pisteiviiva kuvaavat ohjelman kattavuutta ja toteutukseen sisällytettyjä taivutusluokkia.

Luvussa 3.5 on esitetty esimerkkitulosteita ohjelman toiminnasta. Yleisesti voidaan arvioida, että ohjelma toimii odotetusti huomattavan kattavalle määrälle taivutusmuotojen, tapaluokkien, astevaihtelun, aikavaihtelun ja monikollisuuden yhdistelmiä. Mahdollisia ongelmia on suurelta osin ennakoitu sopivien peitesanojen valinnoilla ja lähes huomattomilla, käyttäjälle todennäköisesti merkityksettömillä rajauksilla toteutukseen, kuten luvussa 3.4 on kuvattu.

Ongelmia, joita toteutuksessa ei pystytty täysin estämään, voi tulla sanoista, joille Voikon *analyze*-funktio antaa odottamattomia tuloksia. Voikko kykenee analysoimaan kieltä monipuolisesti huomioiden jopa taivutusmuodot, joita ihminen ei tule ajatelleeksi, kuten ”minä” analysoidaan sanan ”mikä” taivutusmuodoksi persoonapronominin nominatiivin sijaan tai sana ”kasteen” analysoidaan harvinaisen sukunimen Kaste genetiiviksi yleissanan ”kaste” genetiivin sijaan. Jos analyysi sisältää useita mahdollisia tuloksia, juuri oikean, kulloinkin tarkoitetun sanan perusmuodon poimiminen peittämistä varten tuloslistasta on ohjelmallisesti haastavaa, sillä sääntöpohjaiselta toteutukselta puuttuu ihmisen luontainen ymmärrys. Jo syötteen analysointi sana kerrallaan estää tämänkaltaisen semanttista kontekstia jäljittelevän ratkaisun toteuttamisen sääntöpohjaisesti.

Tässäkään asiassa sanankorvauslistoja käyttävä ihminen tuskin tulisi edes ajatelleeksi ongelmaa, jonka ratkaiseminen ohjelmallisesti on haastavaa. Kuvattu tilanne ei ohjelmaa testatessa tullut merkittävästi vastaan, mutta mahdollisuus ongelmille on silti hyvä tiedostaa ja toteutuksen yksityiskohtia pohtia jatkossakin. Kuitenkin kun arvioidaan

ohjelman osaamien sanojen kattavuutta ja toteutuksen toimintaa yleisesti, voidaan arvioida, että luotu prototyyppi on vähintäänkin riittävä.

Tässä luvussa on esitelty ohjelmallinen toteutus sille, kuinka sanankorvauslistojen avulla ihmisen toteuttama tekstisteganografia voidaan toteuttaa tietotekniikkaa hyödyntäen. Suomen kielen ominaispiirteiden vuoksi ihmiskäyttäjälle luontainen sanankäyttö ohjelmallisesti toteutettuna on vaativa ja monimutkainen kokonaisuus, vaikka kaikkein harvinaisimmat sanamuodot jätettäisiinkin toteutuksen ulkopuolelle.

Olemassa olevan peittämisratkaisun modernisoidulle versiolle on kuitenkin olemassa toinenkin, erittäin ajankohtaista teknologiaa hyödyntävä vaihtoehto: tekoälyn ja suurten kielimallien käyttäminen peittämiseen. Seuraavassa luvussa perehdytään tarkemmin voimakkaasti kehittyvään alaan ja myöhemmin esitetään tapoja, joilla suurten kielimallien luomisessa tehtyjä edistysaskeleita voidaan hyödyntää myös tekstisteganografiassa.

4. NEUROVERKOT JA TEKOÄLY

Neuroverkot ovat jo vuosia hallinneet tekoälyn kehitystä ja mahdollistaneet sen suurimmat edistysaskeleet. Tässä luvussa perehdytään lyhyesti koneoppimiseen ennen kuin siirrytään neuroverkkojen syväoppimiseen ja lopulta transformer-arkkitehtuuriin, joihin suuret kielimallit perustuvat.

4.1 Tekoäly ja koneoppiminen

Tekoälyllä (*artificial intelligence*, AI) tarkoitetaan laajasti määritellen tietokoneiden ja ohjelmistojen kykyä toteuttaa ihmisille ominaisia, ajattelua ja älykkyyttä vaativia toimintoja (Goodfellow et al. 2016, s. 1; IBM 2024a). Käytössä oleva tekoäly on kapeaa tai heikkoa (*narrow, weak AI*): se toteuttaa tarkasti rajattua tehtäväjoukkoa, kuten ajoneuvojen automatisoitua ajamista tai virtuaaliavustajien Siri ja Alexa toimintoja (Vartiainen et al. 2021, s. 106; IBM 2024a). Tämän näennäisen älykkään mutta todellisuudessa silti sääntöjä noudattavan tekoälyn vastapari on vahva tekoäly (*strong AI*), jossa koneella saavutetaan ihmistä vastaava, itsestään tietoinen älykkyys. Vahvan tekoälyn viimeinen askel on supertekoäly (*superintelligence*), jossa koneen älykkyys ja kyvykkyys ylittävät ihmisen. Tekoälyn huikeista edistysaskeleista ja hätkähdyttävän nopeasta arkipäiväistymisestä huolimatta vahva tekoäly on puhtaasti teoreettista ja toistaiseksi olemassa vain tieteisviitteessä. (Vartiainen et al. 2021, s. 106; IBM 2024a)

Tekoälyn tärkeä osa-alue ja mahdollistaja on koneoppiminen (*machine learning*), jossa kone (*agent*) oppii ja muodostaa asetettuun ongelmaan ratkaisumallin suuresta määrästä aineistoa ilman, että ohjelmalle syötetään valmiita malleja tai sääntöjä. Koneoppimisen sopiikin erityisen hyvin tilanteisiin, joissa opetusdataa on runsaasti saatavilla mutta ilmiön mallintaminen on ihmisen tekemänä työlästä – jopa mahdotonta – tai tutkimusongelman ratkaisemiseksi epämielikästä. (Goodfellow et al. 2016, s. 1–2; Vartiainen et al. 2021, s. 106) Koneoppimisen arkisia sovelluksia on käytössä lukemattomia, aina automatisoiduista turvajärjestelmistä navigaattorin reittivalintojen optimointiin ja verkkokauppojen henkilökohtaisiin ostosuosituksiin (Vartiainen et al. 2021, s. 108). Koneoppimista toteutetaan kolmella pääasiallisella menetelmällä: ohjattu oppiminen (*supervised learning*), ohjaamaton oppiminen (*unsupervised learning*) ja vahvistusoppiminen (*reinforcement learning*). (Russell & Norvig 2016, s. 693–694; Vartiainen et al. 2021, s. 106–107)

Ohjatussa oppimisessa koneelle annetaan opetusdatana esimerkkejä halutuista syöte-tuloste-pareista (*input-output*), joiden perusteella kone oppii funktion, jolla annetusta syötteestä muodostetaan tuloste tai ulostulo. Ohjattua oppimista käytetään esimerkiksi sähköpostien roskapostisuodattimissa, joiden suoritus paranee sitä mukaa, kun sähköposteja merkitään roskapostiksi tai ei-roskapostiksi. (Russell & Norvig 2016, s. 695; Vartiainen et al. 2021, s. 106–107)

Ohjaamattomassa oppimisessa kone oppii havainnoimaan saamistaan syötteistä samankaltaisuuksia tai yhteyksiä ilman palautetta tekemistään havainnoista. Ohjaamattoman oppimisen yleinen toteutus on klusterointi (*clustering*) eli ryhmittely tai ryvästys, jossa kone pyrkii tunnistamaan hyödyllisiä ryhmiä ilman, että sille on ensin annettu luokiteltuja esimerkkejä tällaisista. Ohjaamatonta oppimista käytetään esimerkiksi verkko-kaupan ostajalle tehdyissä, samaan tuoteryhmiin perustuvissa ostosuosituksissa. (Russell & Norvig 2016, s. 694–695; Vartiainen et al. 2021, s. 107)

Vahvistusoppimisessa koneelle kerrotaan jokin kriteeri toivotulle tai ei-toivotulle toiminnalle, mutta toisin kuin ohjatussa oppimisessa, haluttua toimintaa ei voida suoraan määrittää jokaiselle mahdolliselle syötteelle. Algoritmi pyrkii vahvistusten – palkitsemisen tai rankaisemisen – kautta optimoimaan toimintaansa palkintojen maksimoimiseksi. Vahvistusoppimista voidaan käyttää muun muassa työvuorolistojen suunnitteluun. Kun sovellukselle annetaan palautetta sen ehdottamiin työvuorolistoihin, sovellus oppii suunnittelemaan käyttäjälle mieluisampia vaihtoehtoja. (Russell & Norvig 2016, s. 695; Vartiainen et al. 2021, s. 107; Laatikainen 2024) Ihmisen toteuttama vahvistusoppiminen on myös olennainen osa suurten kielimallien opettamista, mitä käsitellään luvussa 5.3. Koneoppimisen muodostamat mallit koostuvat erilaisista tilastollisista todennäköisyyksistä, joihin kuitenkin perehdytään tässä tutkielmassa vain pintapuolisesti. Koneoppimisen matemaatiikkaan voi perehtyä yksityiskohtaisesti muun muassa Kevin P. Murphyn teoksessa *Machine Learning: A Probabilistic Perspective* (2022).

Tekoälyä hyödyntävien järjestelmien kehitystyön kulmakiveksi on etenkin 2020-luvulla voimakkaasti tullut uudenlainen tekoälymalli: **perustamalli** (*foundation model*). Siinä missä perinteinen tekoälymalli on opetettu tiettyä käyttötarkoitusta varten, perustamallit ovat suurella aineistolla esiopetettuja (*pretrained*) yleismalleja, joita voidaan edelleen opettaa hyvinkin monipuolisia käyttötarkoituksia varten. Niitä voidaan opettaa eri medioilla (teksti, kuva, ääni) ja sen jälkeen käyttää tuottamaan ja ymmärtämään sisältöä eri aloilla, kuten luovilla aloilla, koodauksessa tai biologian tutkimuksessa. (Schneider et al. 2024, s. 1)

Tärkeä sovellus perustamalleista ovat laajalla kieliaineistolla opetetut suuret kielimallit, joihin perehdytään tarkemmin luvussa 5. Samassa yhteydessä käydään läpi myös suurten kielimallien – ja siten samalla perustamallien – opettamiseen käytettyjä tekniikoita, joiden avulla voidaan saavuttaa merkittäviä tuloksia kohtuullisella työpanoksella ja maltillisilla kustannuksilla. Tekoälymallin ja suurien kielimallien opettaminen on perinteisesti erittäin kallista opettamiseen tarvittavien laskentaresurssien takia: OpenAI:n kehittämän GPT-3 kielimallin yhden opetuskierron on arvioitu maksaneen noin viisi miljoonaa dollaria (Smith 2003). Kokonaisen mallin opettaminen on vaatinut useita opetuskierron ja kuukausia koneiden suoritusnopeutta sekä kehittäjien työpanosta, tosin todelliset kustannukset ovat liikesalaisuuksia. Perustamallien avulla laaditut monipuoliset tekoälysovellukset, joiden kehittäminen on nopeaa ja kustannuksiltaan maltillista, ovat ymmärrettävästi houkutteleva tapa tekoälyn hyödyntämiseen.

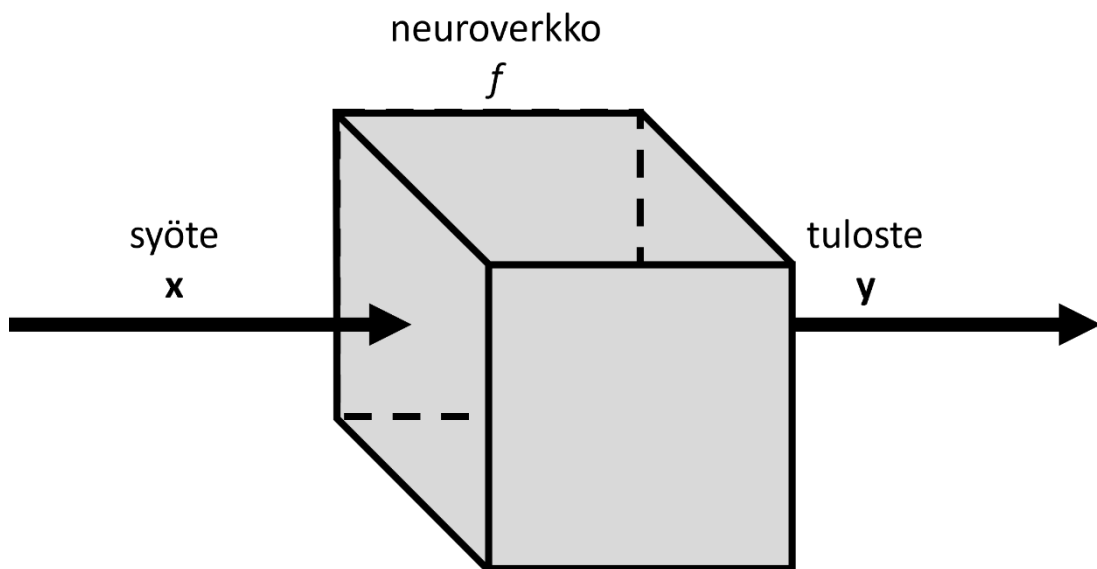
Saatu huomio on tosin tuomassa perustamallit Euroopan Unionin sääntelyn alaiseksi, mikäli tekeillä oleva EU:n tekoälyasetus toteutuu ehdotetussa laajuudessa (Hallamaa 2023a). Ei pidä myöskään aliarvioida tekoälyn kaupallista merkitystä. Suurten teknologiapanostusten ja mahdollisesti näistä saatavien hyötyjen lisäksi myös epäonnistumiset käyvät kalliiksi: Googlen Bard-kielimallin epäonnistunut demonäytös sulatti yhtiön pörssi-arvosta häkellyttävät 100 miljardia Yhdysvaltain dollaria (Teubner 2023, s. 96). Vaikka tällainen isku on teknologiajätin ydinosaamisalueella erityisen raskas, ei ole syytä vähätellä muihinkaan toimijoihin kohdistuvia vaikutuksia tekoälyn onnistuessa tai epäonnistuesssa.

Edelleen on aiheen mainita alati laajemmalle sovellusalueelle levittyvä **generatiivinen tekoäly** (*generative AI*), joka kykenee laajan opetusaineiston omaksuttuaan tuottamaan käyttäjän kehoitteesta (*prompt*) eri medioihin täysin uutta sisältöä, kuten tekstiä, kuvaa, ääntä ja videota. Mainitut perustamallit ja suuret kielimallit ovat tärkeitä esimerkkejä generatiivisesta tekoälystä ja sen käytöstä tekstin tuottamisessa. Generatiivista tekoälyä hyödyntävistä sovelluksista mainittakoon keskustelubotti ChatGPT, kuvia tuottava DALL-E ja GitHubin koodaustyökalu Copilot. Kaikille näille on yhteistä kyky ihmismäiseen vuorovaikutukseen: sovellukselle annetaan jokin tehtävä toteutettavaksi tai esitetään jokin kysymys tai väite, jolloin sovellus reagoi tuottamalla vastauksen tai vastineen samaan tapaan kuin ihminen vastaavan pyynnön saadessaan. Generatiivinen tekoäly ei suinkaan ole erehtymätöntä, mutta sen antamat näytöt ja nopeasti kehittyvä, inhimillistä osaamista yhä paremmin jäljittelevä suorituskyky ovat olleet vaikuttavia jo lyhyessä ajassa. (Martineau 2023; Teubner et al. 2023, s. 95; Kallio 2024)

Seuraavassa perehdytään neuroverkkoihin, joiden ymmärtäminen auttaa ymmärtämään myös tärkeitä suuria kielimalleja.

4.2 Syväoppiminen ja neuroverkot

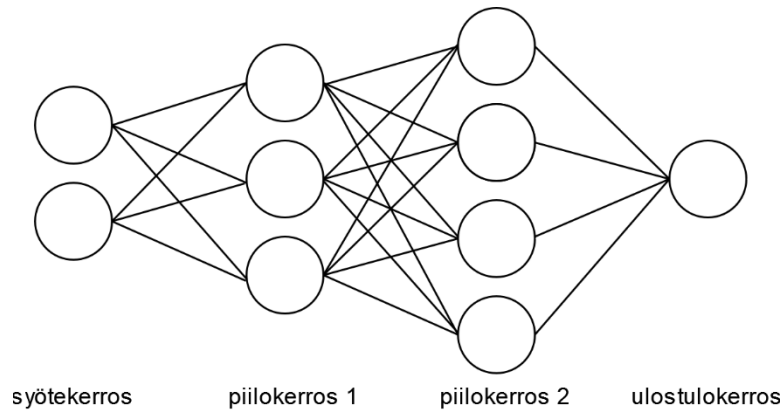
Koneoppimisen kehittyneempi muoto on lineaaristen mallien sijaan keinotekoisiiin neuroverkkoihin (*artificial neural networks*) perustuva syväoppiminen (*deep learning*). Koneälyn ja toisaalta koneoppimisen suuret edistysaskeleet perustuvat juuri neuroverkkojen kehitykseen. (Vartiainen et al. 2021, s. 106) Yksinkertaisimmillaan neuroverkkoa voidaan havainnollistaa jotain kiinnostavaa ilmiötä kuvaavana funktiona f , $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$, jolla n -ulotteisesta syötevektorista \mathbf{x} saadaan m -ulotteinen tulostevektori \mathbf{y} , kuten kuvassa 11 on esitetty.



Kuva 11. Yksinkertaistettu neuroverkko on musta laatikko.

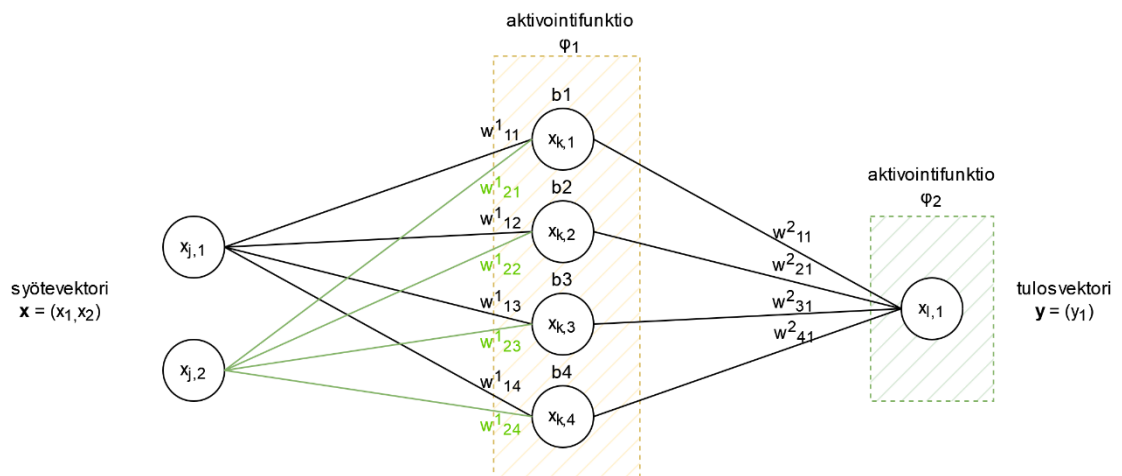
Neuroverkkoa kuvaavaa funktiota f ei tunneta, mutta kone pyrkii oppimaan sen saamansa opetusdatan perusteella. Käytännön neuroverkot ovat luonteeltaan epälineaarisia; itse asiassa useita piilokerroksia sisältäviin syväneuroverkkoihin liittyy luonteenomaisesti eräänlainen perinteisiä koneoppimisen lineaarisia malleja merkittävämpi ennustamattomuus ja epävarmuus siitä, kuinka neuroverkkoja hyödyntävä kone on tarkalleen ottaen päätyyn tiettyyn lopputulokseen. Jopa kehittäjilleen ne ovat eräänlaisia kuvan 11 mukaisia mustia laatikoita (*black box*), jossa mallin osasten tarkka vaikutus mallin toimintaan ja mallin tapa käsitellä annettua syötettä ovat epäselviä (Russell & Norvig 2016, s. 693; Vartiainen et al. 2021, s. 113). Silti niiden suorituskyky ja riittävän luotettava toiminta ovat edelleen mahdollistaneet suuria edistysaskeleita sekä tekoälyn että kone- ja syväoppimisen kehityksessä (IBM 2024a). Neuroverkot ovat käytössä sekä tavanomaisessa koneoppimisessä että syväoppimisessä, mutta erityisen leimallisia ne ovat syväoppimiselle.

Neuroverkot ovat ihmisen aivoja ja biologisia hermosoluja eli neuroneita jäljitteleviä rakenteita. Neuroverkko koostuu **syötekerroksesta** (*input layer*), yhdestä tai useammasta **piilokerroksesta** (*hidden layer*) ja **ulostulokerroksesta** (*output layer*). Kukin kerros koostuu keinotekoisista **neuroneista** (*neuron*), joissa toteutetaan neuroverkon laskutoimituksia ja säilötään verkon opettamisen käytettäviä **parametreja** (*parameter*). (Tuominen 2024) Neuroverkon parametrit muodostuvat **neuronien välisten yhteyksien** painoarvoista (*weight*) ja neuronien vakiokertoimista (*bias*) (Korpihalkola 2024). Esimerkki neuroverkosta on esitetty kuvassa 12.



Kuva 12. Neuroverkko, jossa kahden neuronin syötekerros yhdistyy kolmen neuronin piilokerrokseen, joka edelleen yhdistyy neljän neuronin piilokerrokseen ja edelleen ulostulokerrokseen.

Tarkastellaan seuraavaksi neuroverkon matematiikkaa lähteen Tuominen (2024) pohjalta. Kuvaan 13 on merkitty neuronin saaman syötteen x lisäksi neuroverkon toimintaan vaikuttavat painot w , vakiokertoimet b ja aktivaatiofunktiot φ , joita käsitellään luvussa 4.2.2. Eri kerroksilla voidaan käyttää samaa aktivaatiofunktiota tai eri aktivaatiofunktioita.



Kuva 13. Neuroverkko, jossa kahden neuronin syötekerros yhdistyy neljän neuronin piilokerrokseen, joka edelleen yhdistyy yhden neuronin ulostulokerrokseen. Neuroneilla on lisäksi paino w ja vakiotermi b , jotka vaikuttavat neuronin laskutoimituksiin.

Neuroverkossa seuraavan kerroksen neuronin syötteenä toimii edellisen kerroksen neuronin painotettu summa, johon on lisätty kerroksen vakio-termi b . Otetaan neuroverkko, joka syötekerros on $l = 0$ ja ulostulokerros $l = L$. Tällöin N_l on kerroksen l neuronien lukumäärä, w_{ij}^l on kerroksen $l-1$ neuronin i ja kerroksen l neuronin j välinen paino, kun taas b_j^l on kerroksen l neuronin j vakio-termi.

Kerroksen l neuronin j vastaava painotettu summa saadaan tällöin kaavalla

$$z_j^l = \sum_{i=1}^{N_{l-1}} w_{ij}^l a_i^{l-1} + b_j^l, \quad (1)$$

jossa a_i^{l-1} on edellisen kerroksen neuronin tulos.

Kerroksen l neuronin j painotettu summa z_j^l syötetään aktivaatiofunktioon, jolloin saadaan neuronin tulos eli syöte seuraavaan kerrokseen kaavalla

$$a_j^l = \varphi(z_j^l) = \varphi\left(\sum_{i=1}^{N_{l-1}} w_{ij}^l a_i^{l-1} + b_j^l\right), \quad (2)$$

jossa φ (fii) on kunkin kerroksen aktivaatiofunktio.

Neuroverkot ovat pohjimmiltaan vektori- ja matriisilaskentaa. Vektoreilla ja matriiseilla ilmaistuna kerroksen l vakio-termejä vastaa vektori

$$\mathbf{b}^l = (b_1^l, \dots, b_{N_l}^l), \quad (3)$$

kerroksen l neuroneiden painotettuja summia vastaa vektori

$$\mathbf{z}^l = (z_1^l, \dots, z_{N_l}^l), \quad (4)$$

ja kerroksen l painoja vastaa $N_{l-1} \times N_l$ -ulotteinen matriisi

$$\mathbf{W}^l = \begin{pmatrix} w_{11}^l & w_{12}^l & \dots & w_{1N_l}^l \\ w_{21}^l & w_{22}^l & \dots & w_{2N_l}^l \\ \vdots & \vdots & \ddots & \vdots \\ w_{N_{l-1}1}^l & w_{N_{l-1}2}^l & \dots & w_{N_{l-1}N_l}^l \end{pmatrix}. \quad (5)$$

Tällöin

$$\mathbf{z}^l = \mathbf{a}^{l-1} \mathbf{W}^l + \mathbf{b}^l \quad (6)$$

ja

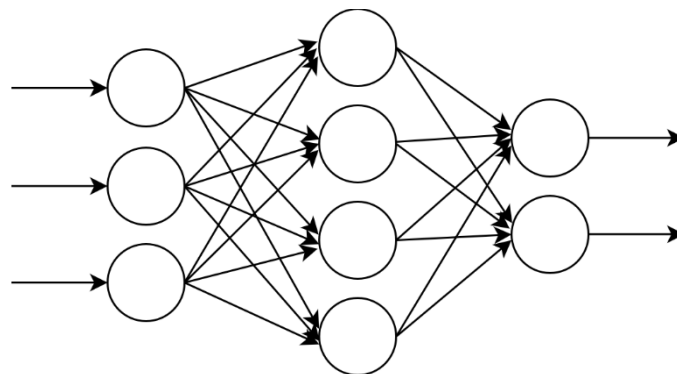
$$\mathbf{a}^l = \boldsymbol{\varphi}(\mathbf{z}^l) = (\varphi(z_1^l), \dots, \varphi(z_{N_l}^l)). \quad (7)$$

Tarkastellaan seuraavaksi erityylyisiä neuroverkon rakenteita.

4.2.1 Neuroverkon rakenne

Neuroverkon rakenne ja neuroverkossa käytetyt kerrokset valitaan käyttötarkoituksen mukaan. Merkittäviä neuroverkkoarkkitehtuuria ovat eteenpäinkytketty neuroverkko (*feedforward neural network*, FFNN tai FNN), takaisinkytketty neuroverkko (*recurrent neural network*, RNN) ja konvoluutioneuroverkko (*convolutional neural network*, CNN). Näistä yleisin on eteenpäinkytketty neuroverkko, jossa tieto kulkee aina yhteen suuntaan, edelliseltä kerrokselta seuraavalle. Sekvenssien käsittelyssä ja tilatietoa säilyttävässä takaisinkytketyssä neuroverkossa puolestaan voi olla silmukoita, joissa tietoa palautetaan myös edeltäville kerroksille. Etenkin kuvantunnistuksessa ja aikasarjadataan kuten äänen käsittelyssä tärkeä konvoluutioneuroverkko puolestaan hyödyntää konvoluutiokerroksia ja muun muassa ristikorrelaatiota eli liukuvaa pistetuloa verkkoa päivittävisissä laskutoimituksissaan. (Murphy 2022, s. 424, 465–467)

Eteenpäinkytketty neuroverkko on käytössä suurille kielimalleille tärkeässä transformer-arkkitehtuurissa, johon palataan luvussa 5. Kuvassa 14 on esitetty eteenpäinkytketty neuroverkko, joka on lisäksi täysin yhdistynyt neuroverkko (*fully-connected neural network*), sillä kunkin kerroksen kaikilla neuroneilla on yhteys seuraavan kerroksen neuroneihin. Useimmat neuroverkot ovat täysin yhdistyneitä. (Korpihalkola 2024)



Kuva 14. Eteenpäinkytketty ja täysin yhdistynyt neuroverkko. Tieto kulkee vain yhteen suuntaan ja kunkin kerroksen neuronit kytkeytyvät aina seuraavan kerroksen neuroneihin.

Syväoppimisessa piilokerroksia on useita (Chhabra & Goyal 2023, s. 220), ja useita piilokerroksia sisältävää neuroverkkoa kutsutaan syväneuroverkoksi (*deep neural network*). Syväneuroverkko voi rakentua sadoista, jopa tuhansista piilokerroksista. Eri kerrokset voivat olla yhdistelmä useita erityyppisiä rakenteita (Sainath et al. 2015, s. 4581). Syötekerroksen kukin neuroni edustaa jotain opetusdatan ominaisuutta tai piirrettä (*feature*). Nämä parametrit esitetään numeroarvoja sisältävänä vektorina. Parametreja kutsutaan toisinaan myös attribuuteiksi (*attribute*). (Murphy 2022, s. 1–2; Korpihalkola 2024)

4.2.2 Aktivaatiofunktio

Aktivaatiofunktioita (*activation function*) käytetään neuroverkon piilokerroksissa ja ulostulokerroksessa muuntamaan edellisen kerroksen syöte epälineaariseksi. Aktivaatiofunktioille vietään edellisen kerroksen neuroneiden syötteiden painotettu summa, johon on lisätty neuronin vakiotermi, jolloin saadaan seuraavalle kerrokselle vietävä epälineaarinen syöte. (Tuominen 2024) Epälineaarisuuden avulla neuroverkon käyttö ei rajoitu vain lineaaristen regressio-ongelmien ratkaisuun, vaan verkon syötteen ja tulosteen välille on mahdollista luoda monimutkaisia, hierarkkisia suhteita, joiden avulla neuroverkolla voidaan mallintaa monimutkaisia, epälineaarisia ongelmia. (Lee 2023, s. 3)

Neuroverkon yleisiä aktivaatiofunktioita ovat logistinen sigmoid-funktio, hyperbolinen tangenti (\tanh) ja transformer-arkkitehtuurissa käytetty ReLU-funktio (*Rectified Linear Unit* tai *rectifier function*) muunnoksineen (Vaswani et al. 2017, s. 5; Tuominen 2024). Näiden rinnalle ovat nousseet GPT-kielimalleissa käytetty GELU-funktio (*Gaussian Error Linear Unit*) (Hendrycks & Gimpel 2016, s. 10) ja tuoreimpana Swish (Ramachandran et al. 2017).

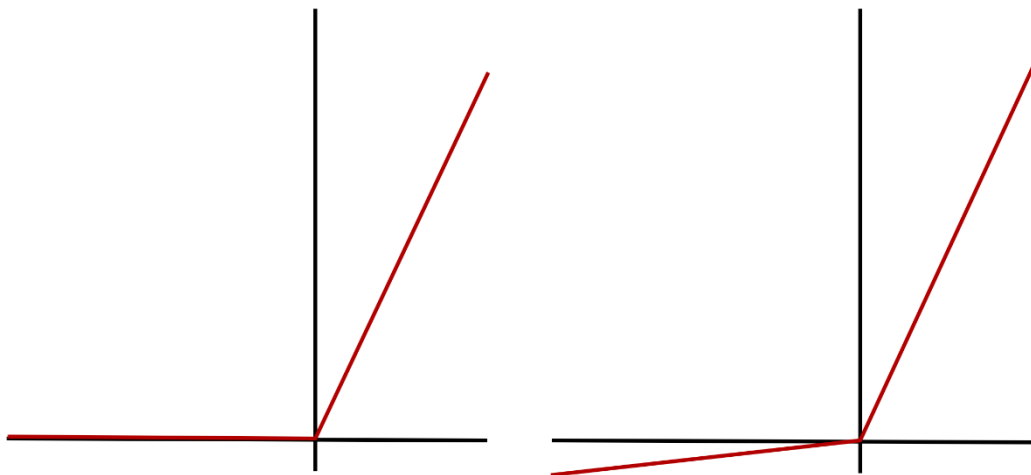
ReLU-funktio voidaan esittää kaavalla seuraavasti (Agarap 2019, s. 2):

$$\text{ReLU}(x) = \begin{cases} x, & \text{kun } x \geq 0 \\ 0, & \text{kun } x < 0 \end{cases} \quad (8)$$

ReLU-funktion variantti, niin sanottu vuotava ReLU (*leaky ReLU*) puolestaan voidaan esittää kaavalla seuraavasti (Maas et al. 2013, s. 3):

$$\text{vuotava ReLU}(x) = \begin{cases} x, & \text{kun } x \geq 0 \\ 0.1x, & \text{kun } x < 0 \end{cases} \quad (9)$$

Molempien ReLU-versioiden kuvaajat on esitetty kuvassa 15.



Kuva 15. ReLU-funktion kuvaaja vasemmalla ja vuotava ReLU -funktion kuvaaja oikealla.

ReLU-funktioiden heikkoudet ilmenevät kuvassa 15. Funktio ei ole derivoituva nollassa, ja negatiivisilla arvoilla alkuperäisen funktion arvo on nolla. Neuroverkon opettamisessa neuronien paino voi tällöin pudota nolnaan, jolloin neuroni kuoleentuu. Tätä ongelmaa ratkaisee osin vuotava ReLU, joka edelleenkin ei ole derivoituva nollassa mutta joka negatiivisilla arvoilla ei myöskään ole arvoltaan nolla. (Tuominen 2024) Neuroverkon opettamiseen palataan tarkemmin luvussa 4.3.

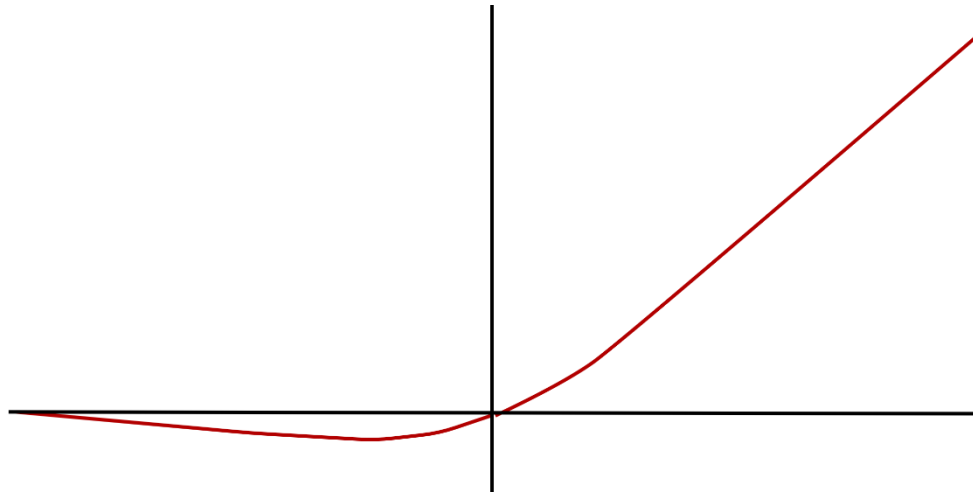
GELU-funktio puolestaan voidaan esittää kaavalla (Lee 2023, s. 3)

$$\text{GELU}(x) = 0.5x \left(1 + \tanh \left(\sqrt{\frac{2}{\pi}} (x + 0.044715x^3) \right) \right), \quad (10)$$

jossa esiintyvä hyperbolinen tangentti (\tanh) määritellään kaavalla

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (11)$$

GELU-funktiolla, joka on aiemman ReLU-funktion sileä approksimaatio (*smooth approximation*), on useita neuroverkon hallinnassa tärkeitä ominaisuuksia. Se on epälineaarinen, kaikkialla derivoituva funktio, jonka derivaatat ovat jatkuvia. Nämä ominaisuudet ovat tärkeitä etenkin silloin, kun verkon opettamisessa käytetään gradienttimenetelmää. (Lee 2023, s. 5) Gradienttimenetelmä on esitelty neuroverkon opettamisen yhteydessä luvussa 4.3. GELU-funktion kuvaaja puolestaan on esitetty kuvassa 16.



Kuva 16. GELU-funktion kuvaaja.

Neuroverkon ulostulokerroksen aktivaatiofunktiona on mahdollista käyttää myös softmax-funktiota. Softmax eli normalisoitu eksponenttifunktio palauttaa neuroverkon ulostulovektorin arvot välille $[0, 1]$ ja arvojen summan arvoon 1. Sitä käytetään sigmoid-funktion ohella etenkin luokitusjärjestelmissä, joilla haetaan todennäköisyyksiä eri luokkien esiintymiselle. (Korpihalkola 2024; Tuominen 2024)

Softmax-funktio voidaan esittää kaavalla seuraavasti (Korpihalkola 2024):

$$\text{softmax}(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}} \quad (12)$$

Neuroverkon opettamisessa – toisin sanoen ilmiötä sopivasti kuvaavan matemaattisen mallin kertoimien etsimisessä – hyödynnetään kahta menetelmää: vastavirta-algoritmia ja gradienttimenetelmää. Käsitellään näitä seuraavassa.

4.3 Neuroverkon opettaminen

Neuroverkon opettamisessa ja mallintamisessa on pohjimmillaan kyse matriisi- ja vektorilaskennasta. Ohjatussa oppimisessa neuroverkon parametreja (kertoimia) eli painoja ja vakiotermejä päivitetään iteratiivisesti siten, että neuroverkon ulostulo saadaan vastaamaan haluttua tulostetta annetulla syötteellä mahdollisimman tarkasti. Ohjaamattomassa oppimisessä haluttu tuloste ei ole tiedossa, mutta myös tämä koneoppimisen laji hyödyntää tässä luvussa esitellyjä matemaattisia metodeja neuroverkon painojen ja vakiotermin määrittämiseen. Koneoppimisen eri lajeja käsiteltiin aiemmin luvussa 4.1. Havainnollisuuden vuoksi tässä käsittelyssä keskitytään ohjattuun oppimiseen lähteen Tuominen (2024) pohjalta. Kerrataan ja esitellään ensin tässä esityksessä käytettävät merkinnät, joita on sivuttu läpi luvun 4.

Neuroverkon parametreja eli kertoimia ovat painot \mathbf{w} ja vakio kertoimet \mathbf{b} . Neuroverkon opettaminen siten, että syötteestä \mathbf{x} saadaan tavoiteltu tuloste \mathbf{y} , edellyttää näiden kertoimien määrittämistä iteratiivisesti. Opetusvaiheessa syötteestä \mathbf{x} saadaan tulos \mathbf{t} , jonka arvoa verrataan virhefunktion E avulla haluttuun tulokseen \mathbf{y} . Neuroverkon opettaminen on siten myös virhefunktion minimointia. Virhefunktion minimoinnissa käytetään useimmiten gradienttimenetelmää, joka puolestaan hyödyntää vastavirta-algoritmia piilokerrosten parametrien päivittämiseen. On huomattava, että tässä käsitellyt matemaattiset kaavat ja neuroverkossa käytetyt algoritmit vaihtelevat neuroverkon käyttötarkoituksen mukaan.

4.3.1 Vastavirta-algoritmi ja virhefunktio

Virhefunktio E (*error function*) on usean muuttujan (kaikkien neuronien painojen ja vakiotermin) funktio, jonka arvoa vastavirta-algoritilla (*backpropagation*) pyritään minimoimaan. Virhefunktio kuvaa neuroverkon tavoitetuloksen \mathbf{y} ja neuroverkon antaman tuloksen \mathbf{t} välisen tuloksen erotusta. (Tuominen 2024)

Neuroverkon opettaminen tapahtuu vastavirta-algoritmin avulla. Aluksi lasketaan halulla syötejoukolla neuroverkon kunkin neuronin aktivaatio ja siitä seuraava tulos \mathbf{t} , joiden

perusteella voidaan mitata virhefunktion saama lukuarvo kyseiselle syötejoukolle. Tämän jälkeen lasketaan painojen ja vakiotermin osittaisderivaatat $\partial E/\partial w$ ja $\partial E/\partial b$ verkon tekemän virheen suhteen, jotta tiedetään, miten painoja tulisi muuttaa virheen pienentämiseksi. Koska gradienttien laskenta etenee iteratiivisesti taaksepäin alkaen verkon viimeisestä kerroksesta aina edeltävän kerroksen virheen gradienttia ja derivaatan ketjusääntöä hyödyntäen, kutsutaan tätä menetelmää vastavirta-algoritmiksi. Verkon uudet painot ja vakiotermit puolestaan saadaan gradienttimenetelmällä (*gradient descent*) muuttamalla hieman näiden parametrien arvoja saatujen gradienttien vastavektoreiden suuntaan, jossa virhefunktion arvo pienenee nopeimmin. Käytännössä vastavirta-algoritmi siis mahdollistaa gradienttimenetelmän soveltamisen monikerroksiselle neuroverkolle. Verkon opettamisessa on tärkeää, että pieni muutos neuronin painossa muuttaa neuroverkon tulosta vain vähän. (Tuominen 2024)

Käytetty virhefunktio valitaan kulloisenkin käyttötarkoituksen mukaan, ja neuroverkon eri kerroksilla voidaan käyttää eri virhefunktioita. Virhefunktio voidaan ilmaista esimerkiksi euklidisen normin neliönä kaavalla

$$E = \frac{1}{2} \|\mathbf{t} - \mathbf{y}\|^2 = \frac{1}{2} \sum_{k=1}^m (t_k - y_k)^2 \quad (13)$$

jossa \mathbf{t} on neuroverkon antama tulos, \mathbf{y} on tavoiteltu tulos, k on neuronien lukumäärä ja m on neuroverkon kerrosten lukumäärä. Ulostulokerroksella hyödynnetään usein etenkin luokitteluun ja todennäköisyyksien laskemiseen soveltuvaa logistisen regression virhefunktiota eli ristientropian virhefunktiota (*cross-entropy cost function*), jonka avulla neuroverkon ulostuloksi saadaan todennäköisyyksiä $[0 \dots 1]$. (Tuominen 2024) Logistisen regression virhefunktio ilmaistaan kaavalla

$$E = -\frac{1}{N} \sum_x \sum_{k=1}^m (y_k \log(a_k^L) + (1 - y_k) \log(1 - a_k^L)), \quad (14)$$

jossa vektori $\mathbf{a}^L = (a_1^L, \dots, a_m^L)$ kuvaa verkon antamia tuloksia ja N on opetusesimerkkijoukon koko. Yleinen regressioanalyysissä käytetty virhefunktio on myös keskineliösumma (*mean squared error*, MSE) (Tuominen 2024). Virhefunktiosta voidaan käyttää myös nimitystä tappio- tai hukka-funktio (*loss function*) tai kustannusfunktio (*cost function*) tai käytetyillä nimityksillä voi olla vivahde-eroja.

Neuroverkolle, jonka ulostulokerrosta merkitään indeksillä L , voidaan esittää virhefunktion osittaisderivaatta viimeisen piilokerroksen painon suhteen kaavalla

$$\frac{\partial E}{\partial w_{ij}^L} = \frac{\partial E}{\partial z_j^L} \frac{\partial z_j^L}{\partial w_{ij}^L} = (t_j - y_j) \varphi'(z_j) a_i^{L-1}, \text{ kun } i = 1 \dots k \text{ ja } j = 1 \dots m, \quad (15)$$

jossa t on neuroverkosta saatu tulos, y on neuroverkon tavoitetulos, φ' on kerroksen aktivaatiofunktion ensimmäinen derivaatta, z on yksittäisen neuronin tulos ja a^{L-1} on verkon viimeisen piilokerroksen neuronin painotettu summa, kun viimeisellä piilokerroksella on i neuronia, $i = 1 \dots k$, ja ulostulokerroksella j neuronia, $j = 1 \dots m$. Kaavan muodostamisessa on hyödynnetty derivoinnin ketjusääntöä. Osittaisderivaatan kaavan tarkka johto on esitetty lähteessä Tuominen (2024).

Virhefunktion osittaisderivaatta ulostulokerroksen vakiotermien suhteen voidaan puolestaan esittää kaavalla

$$\frac{\partial E}{\partial b_j^L} = \frac{\partial E}{\partial z_j^L} \frac{\partial z_j^L}{\partial b_j^L} = (t_j - y_j) \varphi'(z_j), \text{ kun } j = 1 \dots m. \quad (16)$$

Myös tämän kaavan johto, jossa hyödynnetään ketjusääntöä, on esitetty lähteessä Tuominen (2024).

Kuten kaavoista 13 ja 14 voi päätellä, vastavirta-algoritmi on luonteeltaan rekursiivinen: kerroksen parametrien osittaisderivaatat voidaan laskea, kun tunnetaan edeltävän kerroksen osittaisderivaatat. Viimeisen piilokerroksen eli kerroksen $L-1$ virhefunktion osittaisderivaatat painojen ja vakioiden suhteen antavat osittaisderivaatat sitä edeltävälle kerrokselle $L-2$, ja niin edelleen. Näin ollen algoritmi etenee vastavirtaan, viimeisestä kerroksesta kohti ensimmäistä. (Tuominen 2024)

Jatketaan vielä algoritmin kaavojen käsittelyä yleisillä piilokerrosten esityksillä. Aiemmin esiteltyjä merkintöjä käyttäen piilokerrokselle virhefunktion osittaisderivaatta kerroksen l painon suhteen voidaan esittää kaavalla

$$\frac{\partial E}{\partial w_{ij}^l} = a_i^{l-1} \varphi'(z_j^l) \sum_{k=1}^{N_{l+1}} ((t_k - y_k) \varphi'(z_k))^{l+1} w_{jk}^{l+1}. \quad (17)$$

Kuten kaavasta nähdään, piilokerroksen l virhefunktion osittaisderivaatta edellyttää tietoja myös edellisen kerroksen $l-1$ neuronin summasta, kyseisen kerroksen l neuronin j aktivaatiofunktion ensimmäiseen derivaattaan vietävästä neuronin summasta ja seuraavan kerroksen $l+1$ virheen summasta. Kaavan muodostamiseen tarvitaan derivoinnin ketjusäännön lisäksi osittaisderivaattojen ketjusääntöä. Myös tämän kaavan johto on esitetty tarkemmin lähteessä Tuominen (2024).

Piilokerroksen l virhefunktion osittaisderivaatta vakio kertoimien suhteen voidaan vastavasti esittää kaavalla

$$\frac{\partial E}{\partial w_{ij}^l} = \varphi'(z_j^l) \sum_{k=1}^{N_{l+1}} ((t_k - y_k) \varphi'(z_k))^{l+1} w_{jk}^{l+1}. \quad (18)$$

Seuraavaksi käsitellään gradienttimenetelmää, joka hyödyntää vastavirta-algoritmeilla laskettuja derivaattoja.

4.3.2 Gradienttimenetelmä ja neuroverkon oppimisnopeus

Gradienttimenetelmällä minimoidaan virhefunktiota. Virhefunktio on edellä esitettyjen kaavojen mukaisesti usean muuttujan funktio, jota minimoidaan iteratiivisesti. Tuolloin funktiolle haetaan lokaali minimi siten, että neuroverkon neuronien aktivaatiot ja niistä seuraava virhefunktio lasketaan opetusesimerkkijoukon alkuparametrien avulla. Tämän jälkeen parametrien arvoja päivitetään vastavirta-algoritmeilla ja korjataan sopivin askelein gradientin vastavektorin suuntaan, jotta virhefunktion arvo pienenee. Matriisi- ja vektorimuodossa parametrien muutokset piilokerroksella l voidaan esittää kaavoilla

$$w^l \rightsquigarrow w^l - \frac{\alpha}{N} \sum_{x \in A} ((t_k - y_k) \varphi'(z_k))^l (a_x^{l-1})^T \text{ ja } b^l \rightsquigarrow b^l - \frac{\alpha}{N} \sum_{x \in A} ((t_k - y_k) \varphi'(z_k))^l, \quad (19)$$

joissa α on verkon oppimisnopeus (*learning rate*) tai askelpituus (*step size*) ja N on opetusesimerkkijoukon A alkioiden lukumäärä. Parametrien päivitystä jatketaan, kunnes virhefunktion arvo on kyllin pieni, lähellä nollaa. Vaihtoehtoisesti verkon päivitys päätetään, kun opetuskierruksesta toiseen päivittyvä askel menee nollaan. (Tuominen 2024)

Usein käytetään stokastista gradienttimenetelmää (*stochastic gradient descent*, SGD), jossa verkkoa päivitetään jatkuvasti kunkin opetusesimerkkijoukon A syötteen jälkeen. Tällainen verkko on nopeasti oppiva mutta toisaalta herkkä yksittäisten poikkeavien syötteiden aiheuttamille häiriöille. Stokastisen ja perinteisen gradienttimenetelmän välimuotona käytetään minisatsigradienttimenetelmää (*mini batch gradient descent*), jossa opetusesimerkkijoukko syötetään verkolle pienempinä osajoukkoina. Tällainen vähittäinen päivittäminen on laskennallisesti helpompaa hallita, ja samalla verkon päivittyminen lokaaliin minimiin on vakaampaa. (Tuominen 2024)

4.3.3 Mallin yli- ja alisovittaminen

Neuroverkon opettamisen – ja samalla koneoppimisen – yleisiä ongelmia ovat neuroverkon kuvaaman matemaattisen mallin yli- ja alisovittaminen. Ylisovittamisessa malli toimii tarkasti opetusesimerkkijoukolle, mutta ei kykene ennustamaan opetusesimerkkien ulkopuolista laajempaa ilmiötä eikä edes mallin testaamiseen käytettyjä testiesimerkkejä (Cunningham & Delany 2021, s. 3; Tuominen 2024). Malli on toisin sanoen oppinut opetusdatan liian hyvin, eikä kykene yleistämään ilmiötä. Erityisen suuri riski ylisovittaminen on suurissa, tuhansien parametrien neuroverkoissa, kun opetusesimerkkien määrä on rajallinen suhteessa verkon kokoon.

Ylisovittamista voidaan estää useilla tavoilla (Tuominen 2024):

- **Opetusesimerkkien lukumäärää kasvattamalla**
- **Opettamisen lopettamisella riittävän aikaisin** (*early stopping*), jolloin neuroverkon tulos tarkastetaan kunkin opetuskierroksen jälkeen käyttäen näytteitä, joita ei ole käytetty verkon opettamisessa. Opettaminen pysäytetään, kun neuroverkon tulos heikkenee edeltävän opetuskierroksen tuloksen tarkkuudesta
- **Neuroneiden osittaisella poistamisella verkosta** (*dropout layer*) **ja siten verkon koon pienentämisellä**, jolloin yksittäisistä piilokerrosten neuroneista tulee itsenäisempiä ja verkosta kestävämpi (*robust*). Tämä menetelmä keskiarvoistaa verkon toimintaa, kun poistettuja neuroneita muutetaan kunkin kierroksen jälkeen
- Painojen pienentämisellä **L2- ja L1-säännöstelyllä** (*regularization*), jolloin virhefunktioita muutetaan siten, että sen minimi löytyy pienemmillä painoilla. Karkeasti kuvaten L1-säännöstely pyrkii arvioimaan käytetyn datan mediaania ja L2-säännöstely datan keskiarvoa. L1-säännöstely on käytännöllinen etenkin silloin, kun opetusdatan piirteiden joukko on huomattavan suuri ja tärkeiden piirteiden rajaaminen pienempään joukkoon tarpeen (Taunk 2020). Säännöstely hyödyntää edelleen gradienttimenetelmää painojen päivittämisessä

Näiden tekniikoiden yhteisenä tavoitteena on tehdä neuroverkosta kestävämpi ja säätää mallin painoja siten, että neuroverkosta tulee toimivampi (Tuominen 2024). Tässä tutkielmassa erityisen kiinnostavaa on opetusesimerkkien lisääminen, johon palataan opetusesimerkkien generoimisen ja siirto-oppimisen yhteydessä luvussa 5.4.

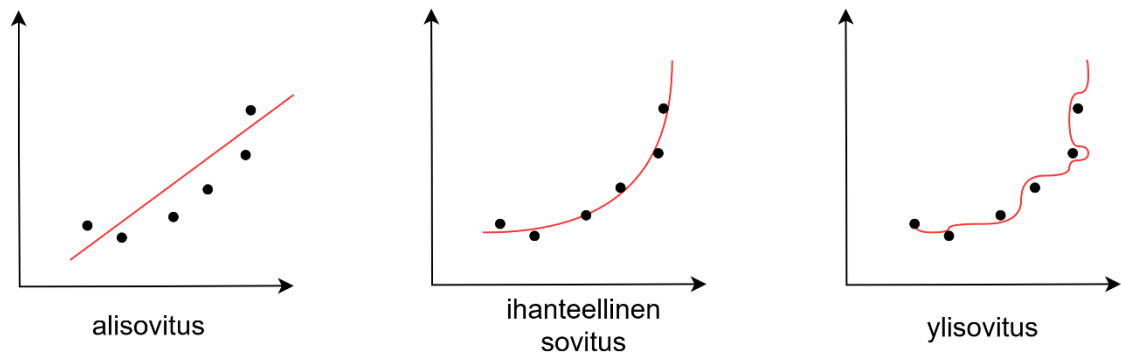
Alisovittamisessa neuroverkon parametrit eivät kykene mallintamaan haluttua ilmiötä, ja verkko oppii huonosti (Tuominen 2024). Siinä missä ylisovitettu malli ennustaa esimerkijoukon täydellisesti mutta ei osaa ennustaa laajempaa ilmiötä eikä verkon toimivuuden varmistamiseen käytettävää testidataa, alisovitettu malli ei kykene ennustamaan edes opetusdataa, vaan syötteen ja neuroverkon antaman tuloksen välinen ero on liian suuri ja verkon antamat tulokset virheellisiä. Yleensä alisovittaminen havaitaan nopeasti aikaisessa vaiheessa opettamista.

Alisovittamista voidaan estää useilla tavoilla (Rahman 2020; IBM 2024b):

- **Lisäämällä verkon piilokerroksia**
- **Lisäämällä piilokerrosten neuronien lukumäärää**
- **Lisäämällä verkon opettamiseen käytettyä aikaa ja muuttamalla verkon arkitekhtuuria**, kuten vaihtamalla käytettyjä aktivaatiofunktioita

Näiden teknikoiden yhteisenä tavoitteena on tehdä neuroverkosta monimutkaisempi, jolloin neuroverkko kykenee mallintamaan tutkittua ilmiötä yksityiskohtaisemmin. Monimutkaisen mallin käsittely vaatii kuitenkin enemmän laitteiston laskentatehoa ja -aikaa, ja monimutkaisuus puolestaan lisää ylisovittamisen riskiä.

Yli- ja alisovittamisen dilemmaa voidaan visualisoida yksinkertaisella kuvaajan sovittamisella mittausdataan. Alisovittamisessa malli – tässä tapauksessa kuvaaja – ei huomioi edes mallin luomisessa käytettyjä mittauspisteitä riittävän tarkasti. Ylisovittamisessa malli huomioi opetusdatan ja sen poikkeavatkin arvot liian yksityiskohtaisesti, jolloin saatu malli ei yleisty tutkittavaan ilmiöön ja on laskennallisesti epäkäytännöllinen. Ihanteellisessa sovituksessa malli kuvaa mittausdataa riittävän hyvin olematta silti tarpeettoman monimutkainen. Sovittamisen haasteita on esitetty kuvassa 17.



Kuva 17. Alisovitetun mallin kuvaaja vasemmalla ja ylisovitetun mallin kuvaaja oikealla. Alisovitettu malli ei huomioi opetusdataa riittävästi, kun taas ylisovitettu malli on liian yksityiskohtainen, eikä yleisty laajempaan ilmiöön. Ihanteellinen sovitus keskellä kuvaa mittauspisteitä riittäväällä tarkkuudella.

Koneoppimiseen perustuvat järjestelmät eivät tyypillisesti yllä täydelliseen tarkkuuteen, mutta usein järjestelmä saadaan toimimaan riittävän hyvin riittävän suurella todennäköisyydellä. Neuroverkko voi muodostua jopa tuhansista eri tavoin rakennetuista piiloverkoista, mutta usein neuroverkon rakenne voidaan lopulta ajatella vain mustana laatikkona, joka toteuttaa annettua tehtävää riittävän hyvin, kuten luvun 4.2 alussa on todettu.

Erilaisten syväoppimisen rakenteiden joukosta esiin nousee kuitenkin yksi ylitse muiden: luonnollisen kielen käsittelyn mullistanut transformer-arkkitehtuuri, jota käsitellään seuraavassa.

4.4 Transformer-arkkitehtuuri

Transformer-arkkitehtuurista on vaikea puhua sortumatta ylisanoihin tai jo sanotun toistamiseen. Googlen tutkijaryhmän vuoden 2017 julkaisussa ”Attention Is All You Need” (Vaswani et al. 2017) esitelty transformer, muunnin, mullisti tekoälytutkimuksen.

Julkaisua on yksin tieteellisten julkaisujen hakupalvelun Google Scholarin mukaan siteerattu yli 144 000 kertaa (joulukuu 2024), ja tekoälyn hätkähdyttävän nopeasta kehityksestä ja arkipäiväistymisestä huolimatta kyseinen arkkitehtuuri on seitsemän vuotta myöhemmin edelleen tekoälykentän johtava teknologia (*state-of-the-art*, *SOTA*) neuroverkkojen rakentamiseen.

Eriyisen leimallista ja tälle tutkielmalle kiinnostavaa ovat olleet harppaukset luonnollisen kielen käsittelyssä ja suurten kielimallien kehitys, sillä arkkitehtuurin avulla on luotu niin sanotusti ”keskustelemaan kykenevä” ChatGPT-palvelu, jossa tekoäly vastailee ihmistä muistuttavasti ja usein järkevän kuuloisesti sille esitettyihin kysymyksiin. Samaan arkkitehtuuriin perustuu myös käännöspalvelu Google Translate, jonka konekäännökset ovat harppauksia ja taas harppauksia edellä lähinnä huvittuneisuutta herättäneitä 2000-luvun alun konekäännöksiä. Huomautettakoon, kuten luvussa 4.1 mainittiin, että tällainen inhimillistä suorituskykyä muistuttava tekoäly on näennäisestä inhimillisyydestään huolimatta heikkoa tekoälyä eikä todellista älykkyyttä, mutta transformerilla saavutetut tulokset ovat joka tapauksessa vaikuttavia: jopa niin, että tekoälyn ja ihmisen tuottaman tekstin ja taiteen rajat ovat alkaneet hämärtyä ja vaatimukset tarkemmasta sääntelystä vahvistuneet (edelleen luku 4.1).

Transformerin käsittelyssä joudutaan valitettavasti turvautumaan englannin ja suomen sekaiseen kieleen, sillä useilla tekoälytutkimuksen ja transformer-arkkitehtuurin nimityksillä ei ole suomenkielisiä nimiä, ei vakiintuneita versioita sen paremmin kuin satunnaisesti käytettyjäkään.

4.4.1 Transformerin syötteen muodostuminen

Käsittely aloitetaan varsinaisen transformerin ulkopuolisesta ja silti siihen kiinteästi liittyvästä asiasta: tokenoijasta ja transformerin syötteen muodostumisesta. Transformer käsittelee syötteenä saamaansa tekstiä pieninä palasina nimeltään *token*, jotka saadaan, kun syötteenä tai tulosteena annettu sana jaetaan pienempiin osasiin. Tokeneiksi jakoa kutsutaan nimellä tokenointi (*tokenization*), ja tokenointi toteutetaan tokenoijassa (*tokenizer*). (Janakiram 2024)

Tokenointi tapahtuu seuraavissa vaiheissa (Restack 2024):

- **Tekstin esikäsittely** (*text preprocessing*), jossa syöte siivotaan ylimääräisistä välilyönneistä, muutetaan pienellä kirjoitetuksi (*lowercase*) ja normalisoidaan. Tämän vaiheen toteuttaa esitokenoija (*pretokenizer*) (Hugging Face 2024)
- **Sanan jakaminen osiin** (*subword segmentation*), jossa algoritmi pilkkoo sanat tokeneiksi käyttämänsä algoritmin mukaan. Tokenoijaa yleisesti kutsutaan sen

tässä vaiheessa käyttämällä algoritmilla, kuten BPE (Sennrich et al. 2015) tai WordPiece (Schuster & Nakajima 2012)

- **Tokenin kuvaus** (*token mapping*), jossa tokenille annetaan pysyvä kokonaislukutunniste **token ID** (*integer ID*), jota käytetään läpi prosessoinnin. Pysyvällä tunnisteella varmistetaan, että sama teksti tokenoidaan aina samalla tavoin

Tokenoijia on useita eri tyyppisiä, kuten GPT-kielimalliperheen ja suomalaisen kielimallin FinGPT:n hyödyntämä BPE (*Byte-Pair Encoding*) eli tavu-parikoodaus, joka ennustaa symboleihin sitomiensa tavuparien todennäköisyyttä. Merkittävä tokenoijatyyppeistä on myös niin ikään transformereihin perustuvien BERT-kielimallien hyödyntämä WordPiece, joka ennustaa malliin syötetyn opetusdatan todennäköisyyttä oppimiensa sääntöjen perusteella. Kolmas transformeriin perustuvissa kielimalleissa käytetty tokenoijatyyppeistä on SentencePiece (Kudo & Richardson 2018), jota hyödynnetään muun muassa kiinan, japanin ja thain kielen tokenoinnissa. Tokenoija voi olla integroitu, kuten GPT-3-kielimallissa, tai sopiva tokenoija voidaan valita kielimallin kulloisenkin käytön mukaan. (Luukkonen et al. 2023, s. 4; Hugging Face 2024; Restack 2024)

Olelliset vaiheet GPT:n BPE-tokenoijan toiminnasta voidaan esittää seuraavasti (tarkka kuvaus Hugging Face 2024):

- Syöte pilkotaan sanoihin esitokenoijassa (*pretokenizer*). BPE oppii sanojen esiintymistiheyden ja luo sanaston, jonka symboleita voidaan myös yhdistellä sen oppimien sääntöjen mukaisesti
- BPE laskee kaikkien mahdollisten symboleiden yhdessä esiintymiselle todennäköisyyden (kuten onko todennäköisempää, että "t" ja "h" esiintyvät yhdessä kuin että "t" ja "j" esiintyvät yhdessä). Vaiheittain etenevien symboleiden yhdistämisten ja mahdollisimman yleisten yhdistelmien tunnistamisen myötä muodostuu lopullinen tokenoijan käyttämä sanasto (*vocabulary*), joka Hugging Facen (2024) englanninkielisessä esimerkissä on

["b", "g", "h", "n", "p", "s", "u", "ug", "un", "hug"]

ja kokoelma yksilöllisiä sanoja, joka on

("hug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("hug" "s", 5)

Esimerkin sanakokoelmassa luvut viittaavat esiintymistodennäköisyyksiin. Sanaston lopullinen koko on ennen tokenointia määriteltävä hyperparametri eli koneoppimisessa ennen opettamisen aloittamista asetettava parametri.

Transformer-pohjaisen kielimallin tavoin myös tokenoija on opetettava ennen käyttöä. Saatavilla on valmiiksi opetettuja tokenoijia, mutta käytetyn sanaston voi laatia ja tokenoijan opettaa myös itse. Näin toimi muun muassa FinGPT-kielimallin laatinut tutkijaryhmä, joka käytti pohjana ohjelmistoyhtiö Hugging Facen tokenoijaa, mutta tyypillisiin englannin kielen tokenoijiin verrattuna toteutti tokenoijalle poikkeksellisen laajan sanaston, yli 131 000 tokenia. Tällä pyrittiin huomioimaan suomen kielen erityispiirteitä. (Luukkonen et al. 2023, s. 4)

Vaikka transformerin tärkeät käsitteet *embedding* and *positional encoding* esitetäänkin osana transformerin rakennetta luvun 4.4.2 kuvassa 19, niitä on aiheen käsitellä jo tässä. Tokeneiden ohella molemmat koodaavat matemaattiseen muotoon sellaista kielen semanttista ymmärrystä, joka osaltaan sallii tilastollisen mallin tuottaa inhimillistä jäljittelevää kieltä.

Termi *embedding* viittaa eräänlaiseen 'upotteeseen' tai 'upottamiseen': embedding-vektorilla sanaan upotetaan semanttista ja leksikaalista merkitystä. Metzger (2022) kuvaa osuvasti, että embeddingin avulla sanat (tokenit) tuodaan vektoriavaruuteen, jossa malli oppii yhtäläisyyksiä sanojen ja niiden merkitysten välillä. Jättimäinen embedding-vektori sisältää liukulukuja, jotka mallin opettamisen aluksi asetetaan johonkin alkuarvoon (*initialization*), ja kukin tällaisen vektorin alkio kuvaa jotain sanan ominaisuutta.

Kaksi samankaltaista sanaa, kuten "ruusu" ja "neilikka", sijaitsevat vektoriavaruudessa lähekkäin. Ne voisivat jakaa vaikkapa ominaisuuksia (*feature*) "kukkamaisuus", "kasvimaisuus", "punaisuus" ja "leikkokukkamaisuus", jolloin tällaiset ominaisuudet saisivat toisiaan lähellä olevia lukuarvoja embedding-vektorissa. Puolestaan kaksi erilaista sanaa, kuten "malmi" ja "ymmärrys", sijaitisivat kaukana toisistaan. Ne saisivat toisistaan kaukana olevia lukuarvoja vaikkapa ominaisuuksissa "metallimaisuus", "kaivostoimintamaisuus" ja "luonteenpiirremäisyys". Oikeassa mallissa tämänkaltaisia matemaattisesti arvoitettuja ominaisuuksia olisi kullakin sanalla satoja tai tuhansia. Embedding kuvaa siten sanojen liittymistä toisiinsa ja niiden merkitystä. (Metzger 2022)

Tärkeä ominaisuus on niin ikään *positional encoding*, jota tässä kutsuttakoon nimellä sijainnin koodaus. Transformer-arkkitehtuuri ei sisällä aiemmin merkittävässä roolissa olleita konvoluutiokerroksia ja takaisinkytkettyjä kerroksia, vaan niiden roolin on ottanut sijainnin koodaus, jonka avulla ilmaistaan ja säilytetään sanojen keskinäistä järjestystä sekvenssissä (Vaswani et al. 2017, s. 6; Li et al. 2023, s. 3). Samat sanat eri järjestyksessä voivat ilmaista eri asiaa tai muuttaa sanotun kontekstia. Esimerkiksi lauseet "car ran over the deer" tai "deer ran over the car" tarkoittavat aivan eri asiaa. Alkuperäinen tutkimusryhmä hyödynsi sijaintitiedon tallennuksessa siniaallon muotoa, jonka arveltiin

toimivan myös tilanteessa, jossa syötteen sekvenssit ovat mallin opetusdatassa käytettyjä pidempiä (Vaswani et al. 2017, s. 6). Sijainnin koodaus voidaan lisätä suoraan embedding-vektorin päälle (*superimpose*), sillä vektorit ovat samanulotteisia. (Li et al. 2023, s. 3)

Lopullinen transformerin syöte – samoin kuin ulostulo – muodostuu token ID:n, embeddingin ja sijainnin koodauksen summana. On syytä huomauttaa, että token ID:t itsessään eivät ole vektorimuotoisia, vaan niistä tehdään sellaisia *one-hot*-koodauksella (*one-hot encoding*) käsittelyn mahdollistamiseksi. One-hot-koodauksessa luodaan vektori asettamalla merkityksellisen indeksin alkio ykköseksi ja täyttämällä muut vektorin alkiot nolilla (*padding*). Todellinen merkityksellinen vektorisointi syntyy vasta embedding-osan liittämisen myötä (Metzger 2022). Transformerin syötteen muodostumista tekstisyötteelle on havainnollistettu kuvassa 18.

$$\begin{array}{ccccccc}
 & \text{token} & & \text{embedding} & & \text{sijainnin} & & \text{syöte} \\
 & & & & & \text{koodaus} & & \\
 \begin{bmatrix} t_{0,0} & \cdots & t_{0,d} \\ \vdots & \ddots & \vdots \\ t_{d,0} & \cdots & t_{d,d} \end{bmatrix} & + & \begin{bmatrix} e_{0,0} & \cdots & e_{0,d} \\ \vdots & \ddots & \vdots \\ e_{d,0} & \cdots & e_{d,d} \end{bmatrix} & + & \begin{bmatrix} pe_{0,0} & \cdots & pe_{0,d} \\ \vdots & \ddots & \vdots \\ pe_{d,0} & \cdots & pe_{d,d} \end{bmatrix} & = & \begin{bmatrix} x_{0,0} & \cdots & x_{0,d} \\ \vdots & \ddots & \vdots \\ x_{d,0} & \cdots & x_{d,d} \end{bmatrix}
 \end{array}$$

Kuva 18. Transformerin syötteen muodostuminen tekstisyötteelle.

Token ja embedding ovat niin tärkeitä rakennusosia, että niihin ja etenkin tokenoinnin haasteisiin suomen kielessä palataan vielä suurten kielimallien yhteydessä luvussa 5.2.

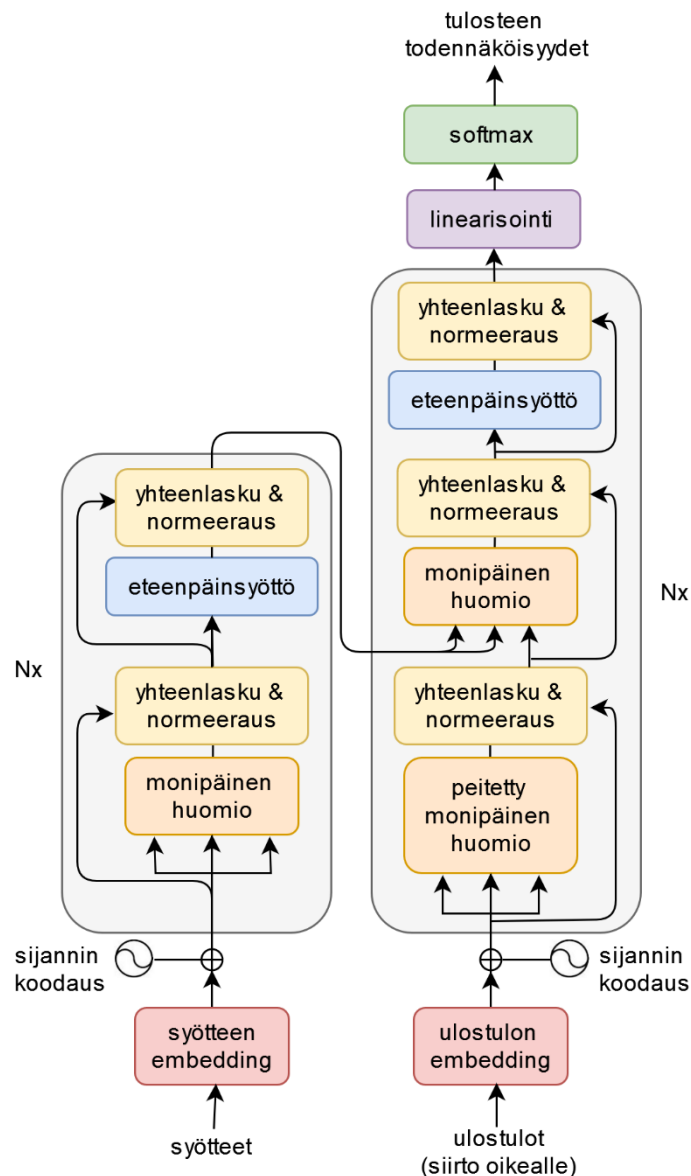
4.4.2 Transformerin rakenne

Kenties suurimpana innovaationa transformer perustuu tarkkaavaisuuteen (*attention*), jonka avulla neuroverkon syöte ja tuloste kytketään toisiinsa (Vaswani et al. 2017, s. 1–2). Tarkkaavaisuus on syväoppimisen mekanismi, joka neuroverkkojen tapaan jäljittelee ihmisen luontaista toimintaa. Siinä missä neuroverkot matkivat ihmisen fyysistä hermostoa, tarkkaavaisuus matkii ihmisen kognitiota.

Kun ihminen poimii kokonaisuudesta olennaiset asiat silmäilemällä tekstin tai kuvan tiettyjä osia, sama valikoiva keskittyminen (*selective focus*) toteutetaan syväoppimisessa tarkkaavaisuusmekanismilla, jossa annettujen syötteiden tietyille osille annetaan suurempi painoarvo kuin toisille (Hong 2023). Tällä tavoin neuroverkkomalli oppii tunnistamaan syötteistä tärkeitä elementtejä ja muun muassa suurissa kielimalleissa ja luonnollisen kielen käsittelyyn liittyvissä tehtävissä tunnistamaan yhteenkuuluvia sanoja ja elementtejä. Transformerin käyttämä itsetarkkaavaisuus (*self-attention* tai *intra-attention*)

on tarkkaavaisuuden laji, jolla yksittäisen sekvenssin osaset suhteutetaan toisiinsa ennen sekvenssin perusteella laskettavaa ulostuloa (Vaswani et al. 2017, s. 2).

Aiemmin luonnollisen kielen käsittelyyn erikoistuneet neuroverkot hyödynsivät konvoluutiokerroksia ja takaisinkytkettyjä kerroksia (*recurrent layer*), mutta transformereissa ne korvannut tarkkaavaisuusmekanismi mahdollisti erinomaisten tulosten lisäksi myös aiempaa tehokkaamman rinnakkaislaskennan (*parallel computing*), jolla mallin suorituskykyä voitiin merkittävästi parantaa. Tarkkaavaisuusmekanismilta puuttuva sekvenssi-tieto on transformer-rakenteessa tuotettu sijainnin koodauksella (*positional encoding*), joka esiteltiin luvussa 4.4.1. Transformerin rakenne on esitelty kuvassa 19.



Kuva 19. Transformerin rakenne. Enkooderi vasemmalla ja dekooderi oikealla. Suomennettu lähteestä Vaswani et al. (2017, s. 3).

Alkuperäinen transformer-arkkitehtuuri käyttää enkooderi-dekooderi-rakennetta (*encoder-decoder*), pinottua itsetarkkaavaisuutta (*stacked self-attention*) ja täysin kytkettyjä, eteenpäinkytkettyjä kerroksia (*fully-connected feedforward layer*) (Vaswani et al. 2017, s. 3). Tällaisessa rakenteessa enkooderi prosessoi syötteen ja dekooderi tuottaa tuloksen. Enkooderi-dekooderi-rakenteen kielimallit soveltuvat erityisen hyvin kielen ja kontekstin ymmärtämiseen sekä luokitteluun liittyviin tehtäviin. Nimenomaisesti tekstintuottoon (*text generation*) erikoistuneet kielimallit, kuten GPT, puolestaan hyödyntävät rakenteenaan tyypillisesti pelkkää dekooderia (*decoder-only*) sekvenssissä seuraavan tokenin ennustamiseen. (Fu et al. 2023, s. 1; Naveed et al. 2023, s. 24; Nielsen et al. 2024, s. 1)

Enkooderi koostuu kuudesta samanlaisesta kerrospinosta, $N = 6$, jossa kukin kerros koostuu kahdesta alikerroksesta (*sublayer*). Ensimmäisellä alikerroksella sijaitsee monipäinen itsetarkkaavaisuusmekanismi ja toinen kerros on täysin kytketty ja eteenpäinkytketty. Kukin kerros normeerataan siten, että kerroksen ulostulo on

$$\text{alikerroksen_ulostulo} = \text{kerrosnormi}(x + \text{alikerros}(x)) \quad (20)$$

jossa $\text{alikerros}(x)$ on alikerroksen funktio. Normeeraus parantaa transformerin suorituskykyä ja vakautta merkittävästi (Naveed et al. 2023, s. 22).

Dekooderissa alikerroksia on kolme, ja kolmannella kerroksella toteutetaan monipäinen tarkkaavaisuus enkooderin ulostulosta. Itsetarkkaavaisuuden alikerros on peitetty (*masked*), ja ulostulon *embedding* siirretään yhdellä.

Vaswani et al. (2017, s. 3–4) kuvaavat tarkkaavaisuusfunktioita kyselyn ja avain-hyötykuorma-parien yhdistämisenä tulosteeseen. Kysely \mathbf{Q} (*query*), avaimet \mathbf{K} (*key*) ja hyötykuormat \mathbf{V} (*value*) ovat matriiseja, joiden avulla tuloste lasketaan painotettuna summana (*weighted sum*). Nämä matriisit saadaan projektiona eli lineaaritransformaationa alkuperäisestä syötesekvenssistä kullekin tokenille, ja neuroverkon opettamisen aikana matriisit oppivat eri painokertoimet käyttötarkoitustensa mukaan. Laveasti kuvaten \mathbf{Q} liittyy kerroksen ulostuloon, \mathbf{K} ulostulon toteuttaneeseen syötteeseen, ja \mathbf{V} on syötteeseen liittyvä, laskennallisesti opetettu kuvaus. Yhdessä näillä toteutetaan tarkkaavaisuutta ja määritetään syötteen eri osien ja eri tokeneiden keskinäistä tärkeyttä. (Dugas 2024; Nguyen 2024)

Hyödyntäen \mathbf{Q} -, \mathbf{K} - ja \mathbf{V} -matriiseja transformerin käyttämä monipäinen itsetarkkaavaisuus (*multi-head self-attention*) voidaan esittää kaavalla (Vaswani et al. 2017, s. 5)

$$\text{monipäinen}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{konkat}(\text{pää}_1 \dots \text{pää}_h) \mathbf{W}^0, \quad (21)$$

jossa konkat viittaa konkatenointiin (*concatenation*) eli merkkijonojen yhdistämiseen ja \mathbf{W}^O on ulostulon paino. Yksittäinen pää (*head*) eli **rinnakkainen tarkkaavaisuuskerros** (*parallel attention layer*) saadaan kaavalla

$$\text{pää}_i = \text{tarkkaavaisuus}(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V), \quad (22)$$

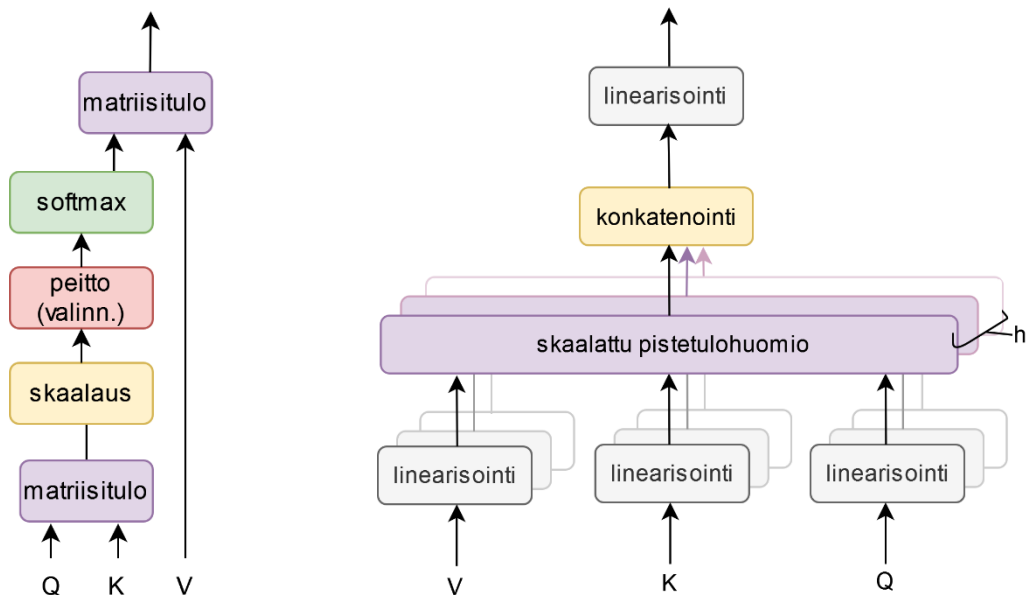
jossa tarkkaavaisuusfunktion parametrit ovat matriiseja.

Monipäisen tarkkaavaisuuden osa skaalattu pistetulotarkkaavaisuus (*scaled dot-product attention*) tuottaa painot avaimille \mathbf{Q} , \mathbf{K} ja \mathbf{V} kaavalla (Vaswani et al. 2017, s. 4)

$$\text{tarkkaavaisuus}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d_k}}\right)\mathbf{V}, \quad (23)$$

jossa d_k on syötteen ulottuvuus. Softmax-funktion kaava on esitetty luvussa 4.2.2.

Edellä kuvattu tarkkaavaisuusmekanismi on esitetty kuvassa 20. Kuvassa esiintyvä peitto (*masking*) toteutetaan vain dekooderissa asettamalla softmax-funktion parametreja mahdollisimman pieneksi ($-\infty$), jolloin funktion tuottama arvo menee nolliin ja dekooderissa virheellinen tulosten siirtyminen vasemmalle estyy (Vaswani et al. 2017, s. 5; Murphy 2022, s. 518). Merkintä h (head) kuvaa rinnakkaisia tarkkaavaisuuskerroksia. Linearisoinnin ja softmax-funktion avulla dekooderin ulostulona saadaan todennäköisyyksiä, joilla ennustetaan järjestyksessä seuraavaa tokenia.



Kuva 20. Transformerin tarkkaavaisuusmekanismien kerrosrakenteita. Skaalattu pistetulotarkkaavaisuus vasemmalla ja monipäinen tarkkaavaisuus oikealla. Suomennettu lähteestä Vaswani et al. (2017, s. 4).

4.4.3 Transformerin sovellusalueet

Transformer-arkkitehtuurin vahvuuksia ovat soveltuvuus hyvin erityyppisiin käyttötarkoituksiin, rinnakkainen ja siten nopea sekvenssin osien prosessointi ja lyhyt opetusaike verrattuna aiemmin hallitsevina olleisiin konvoluutiota ja takaisinkytkentää hyödyntäviin neuroverkkoihin (Vaswani et al. 2017, s. 1). Transformer voi myös selvittää luonnollisen kielen käsittelyn tehtävistä ilman tehtäväkohtaista opettamista. Neuroverkon opettamisessa tämä vaihe on yleensä työläs ja kallis toteuttaa (ibid., s. 1, 10; mainittu myös luku 4.1). Suurin vahvuus on kuitenkin yksittäisten – vaikka sinänsä merkittävien ja ainutlaatuistenkin – teknisten ominaisuuksien sijaan transformereiden suhteellisen lyhyessä ajassa antamat näytöt hyvin erityyppisillä tekoälyn sovellusalueilla myös kaupallisessa käytössä.

Tämän tutkielman kiinnostuksen kohteena ovat erityisesti luonnollisen kielen käsittelyyn liittyvät tehtävät, mutta transformereita on hyödynnetty erityisen laajassa valikoimassa tekoälytehtäviä, mukaan lukien konenäkö, konekääntäminen, automaattinen puheentunnistus ja puhesynteesi sekä lääketieteellinen kuvantaminen (Latif et al. 2023). Sitä on käytetty kuvien, musiikin ja proteiinisekvenssien luomiseen (Murphy 2022, s. 524). Transformeriin perustuva ohjelma on jopa voittanut ammattilaispelaajia reaaliaikaisessa strategiapelissä StarCraftissa luomalla innovatiivisia strategioita, joita ihminen ei osannut odottaa (AlphaStar 2019).

Menestyksekkäällä teknologialla on myös heikkoutensa, joita Thangavel (2023) tunnistaa useita. Huolimatta edistysaskeleista edeltäviin tekniikoiden nähden transformerille olennainen *embedding* eli syötteen tai tulosteen muuntaminen vakiomittaiseksi vektoriksi vaatii merkittävästi muistia ja laitteiston suoritustehoa. Tämä rajaa jonkin verran laitteita, joilla transformereita voidaan käyttää. Toisaalta sekvenssien oletettu vakiomittaisuus ja itsetarkkaavaisuusmekanismi eivät skaalaudu kovin hyvin vaihtuvamittaisiin tai erittäin lyhyisiin syötteisiin, vaan niiden käyttö heikentää transformerin toimintaa ja luotettavuutta (myös Ji et al. 2022, s. 13).

Osa Thangavelin (2023) nimeämistä heikkouksista johtuu pikemminkin yksinkertaisesti siitä, että transformer – tai mikään muukaan olemassa oleva neuroverkko – ei todellisuudessa ole älykäs, vaikka sillä voidaan tuottaa keskustelua simuloiva sovellus, joka osaa vastata kysymyksiin ihmisen lailla. Kuten luvussa 4.1 todetaan – ja mitä on syytä korostaa – kehittyneinkin tekoäly on edelleen näennäisen älykäästä heikkoa tekoälyä, joka ei kilpaile ihmisen kanssa. Transformer ei kykene luotettavasti arkijärjen mukaiseen päätelyyn, uusiin laskutoimituksiin tai johtopäätöksiin (Teubner et al. 2023, s. 97; Thangavel 2023; Peng et al. 2024, s. 2), koska näiden tehtävien moninaisuus vaatii joko luontaista

älyä tai niin kattavan aineiston opetusdataa, ettei sellaista toistaiseksi ole olemassa. Transformerin toiminta on läpinäkymätöntä, joten jopa kehittäjien on vaikea päätellä, miksi malli päätyy väärin tuloksiin tai miten ongelma estetään jatkossa. Etenkin satunnaiset tai harvoin toistuvat tapahtumat ovat transformerille ongelmallisia. (Thangavel 2023) Läpinäkyvyyden puutetta on kuitenkin vaikea pitää nimenomaan transformerin heikkoutena. Kuten luvussa 4.2 on esitetty, kaikkien neuroverkkojen tapaan transformerin on lopulta eräänlainen musta laatikko, jonka luotetaan toimivan riittävän hyvin. Vanhan ohjelmistosuunnittelun viisauden mukaisesti tämä on ominaisuus, ei virhe.

Transformerit kärsivät myös suurten kielimallien ja etenkin ChatGPT-sovelluksen tunnetuksi tekemästä hallusinoinnista (*hallucination*), jossa sovellus esittää täysin valheellisia, jopa hupaisan epätodellisia väitteitä hyvinkin vakuuttavasti täytenä totena. Peng et al. (2024, s. 1–2) esittävät, että hallusinointi on transformer-arkkitehtuurin sisäsyntyinen, väistämätön ominaisuus enemmän kuin mallin opettamisen tai opetusdatan laadusta aiheutuva ongelma, tosin Ji et al. (2022, s. 11–14) esittävät useita keinoja ainakin vähentää transformerin hallusinointia juuri näihin tekijöihin paneutumalla. Keinoja ovat opetusdatan laadun parantaminen, relevantin datan määrän lisääminen ja mallin vahvistusopettaminen (luku 4.1), joilla kaikilla vähennetään kohinaisen datan (*noisy data*) aiheuttamaa hallusinointia.

Myös pienet muutokset transformerin rakenteeseen voivat tehdä sen tarkkaavaisuusmekanismista kestävämmän ja estää transformerin luontaista taipumusta hallusinointiin (Ji et al. 2022, s. 13). Hallusinointiin palataan suurten kielimallien yhteydessä luvussa 5. Vaikka nykyisiin transformer-pohjaisiin kielimalleihin on toteutettu alkuperäisestä poikkeavia rakenteita, kuten pelkän dekooderin käyttö enkooderi-dekooderi-rakenteen sijaan ja Swish-aktivaatiofunktion käyttö ReLU:n sijaan (mm. Ramachandran et al. 2017; Naveed et al. 2024), Vaswanin et al. vuoden 2017 julkaisu on edelleen koneellisen kielenkäsittelyn *de facto* -standardi.

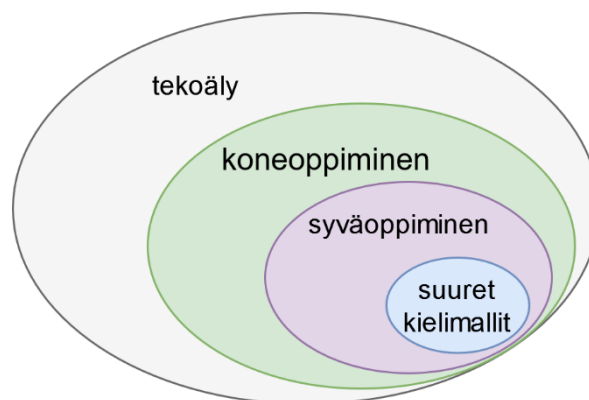
Seuraavassa luvussa syvennyttään suuriin kielimalleihin ja palataan alkuperäiseen peittämissovellukseen, jonka avulla ohjelmallinen peittäminen onkin toteutettavissa tekoälypohjaisesti. Samalla esitellään tekniikoita, joilla raskasta laskentatehoa ja huomattavaa työmäärää vaatineita kielimalleja voidaan hyödyntää tarkkarajaisiin käyttötarkoituksiin yllättävän kevyellä panostuksella, jopa kannettavalla tietokoneella ja yksinkertaisilla kehoitteilla.

5. PEITTÄMINEN KIELIMALLEILLA

Tässä luvussa käsitellään suuria kielimalleja, joiden kehittämisen ja sittemmin esiinmarssin transformer-arkkitehtuuri ja tekoälyn harppaukset mahdollistivat. Uutena mahdollisuutena esitetään tapa toteuttaa peittäminen sanankorvauksella kielimallien avulla. Siinä missä kielimallien laatiminen vaatii mittavaa opetusaineistoa ja laskentatehoa, vapaasti julkaistujen mallien hyödyntäminen tällaista erikoistarkoitusta varten voi olla mahdollista jopa kannettavalla tietokoneella. Erityinen mielenkiinnon kohde ovat suomalaiset kielimallit. Kun sääntöpohjaisessa peittämisessä suomen kielen erityispiirteet vaativat kehittäjältä aivan erityistä osaamista ja syvää ymmärrystä kielen rakenteesta, tekoälypohjainen peittäminen on haasteellista pienen kielialueen ja vielä pienempien kehitys- ja tutkimusresurssien vuoksi. Tarvittavan opetusdatan tuottamiseen tekoälymallille on kuitenkin ratkaisu: Hyödynnetään laadittua peittämisohjelmaa lisäaineiston tuottamiseen. Lopussa raportoidaan kaupallisella mallilla toteutetun peittämistestin tulokset.

5.1 Suuret kielimallit

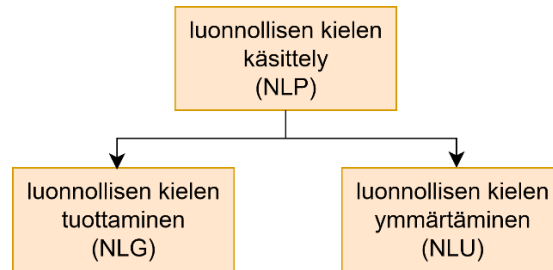
Suuria kielimalleja (*large language models*, LLMs) on sivuttu tutkielmassa useaan kertaan. Yksinkertaisesti määritellen suuret kielimallit ovat suurella datamäärällä opetettuja tilastollisia malleja, jotka laskevat yhteistodennäköisyyksiä sanojen tai tokeneiden esiintymiselle. Suurten kielimallien kehitys pohjautuu luvussa 4.4 esiteltyyn transformer-arkkitehtuuriin ja laajemmin syväoppimisen ja neuroverkkojen edistysaskeleisiin. Suurten kielimallien asemaa tekoälykentällä on havainnollistettu kuvassa 21.



Kuva 21. Suurten kielimallien suhde muihin tekoälykentän osa-alueisiin.

Suurten kielimallien vahvuus on erilaisissa **luonnollisen kielen käsittelyyn** (*natural language processing*, NLP) liittyvissä tehtävissä: näihin kuuluvat **luonnollisen kielen ymmärtäminen** (*natural language understanding*, NLU) ja **luonnollisen kielen**

tuottaminen (*natural language generation*, NLG). Luonnollisen kielen jaottelua eri osa-alueisiin on havainnollistettu kuvassa 22.



Kuva 22. Luonnollisen kielen käsittelyn osa-alueet.

Kielen ymmärtämiseen liittyviä tehtäviä ovat luetun ymmärtämisen ohella tekstityyppien luokittelu, tunnetilojen tunnistaminen tekstissä, järkeily ja matemaattinen päättely, kun taas tekstin tuottamiseen liittyviä tehtäviä ovat lauseiden täydennys, tiivistelmien tai uuden tekstin laatiminen ja konekäännökset. (Naveed et al. 2023, s. 24) Esimerkkejä luokittelutehtävistä (*classification*) ovat käyttäjäarvostelun tulkitseminen myönteiseksi tai kielteiseksi tai tekstityypin tunnistaminen proosaksi tai uutiseksi. Erityisen huomionarvoista ovat suurten kielimallien näytöt generatiivisen tekoälyn käytössä, kun suurella tekstiaineistolla opetetut mallit kykenevät tuottamaan täysin uutta tekstiä.

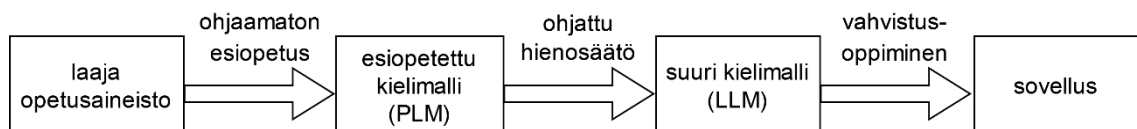
Tunnettuja kaupallisia sovelluksia ovat suuret kielimallit popularisoineet OpenAI:n GPT-kielimalliperhe ja Googlen T5-sarja, Bard sekä tämän jatkaja Gemini. Yhteensä Naveed et al. (2023, s. 2) nimeävät vuosilta 2019–2024 lähes 60 suurta kielimallia, yhtä monta suljetun lähdekoodin kuin avoimen lähdekoodinkin julkaisua lukuisilta toimijoilta, ei ainoastaan teknologiajäteiltä. Tähän lukemaan ei sisälly tässä tutkielmassa GPT:n ohella erityisen kiinnostava FinGPT, valmistuessaan laajin suomen kielellä laskettu kielimalli (Hallamaa 2023b). Turun yliopiston tutkijaryhmän TurkuNLP:n ja ohjelmistoyhtiö Hugging Facen yhteistyössä luoma FinGPT on kehitetty OpenAI:n kielimallin GPT-3 pohjalta (Luukkonen et al. 2023, s. 2), mutta on sen lähdekoodi ja lopullinen jäädytetty (*frozen*) kielimalli ovat vapaasti saatavilla. Sittemmin tutkijaryhmä on jatkanut pienten kielten kehitystyötä Silo AI:n kanssa toteutetulla suomenkielisellä Poro-kielimallilla ja siihen perustuvalla yhteispohjoismaisella, monikielisellä Viking-kielimallilla. Myös nämä kielimallit ovat vapaasti saatavilla. (Aamuset 2024; Turun yliopisto 2024)

On kuitenkin huomattava, että sinänsä ansiokkaat suomenkieliset suuret kielimallit eivät ole läpikäyneet mallin inhimillistämässä olennaista vaihetta: **vahvistusoppimista ihmispalautteella** (*reinforcement learning from human feedback*, RLHF) (Stöffelbauer 2023). ChatGPT:n menestys johtuu osin juuri siihen kohdistuneesta monipuolisesta kiinnostuksesta ja käytetystä mittavasta työpanoksesta (Teubner et al. 2023, s. 96).

LUMI-supertietokoneella kuukausia lasketut kielimallit ovat tavallaan raakadataa, joka sisältää todennäköisyyksiä ja tarvittavat luonnollisen kielen käsittelyn rakennusosat keskinäisine suhteineen mutta jota ei ole keskustelubotti ChatGPT:n tapaan opetettu ja palkittu antamaan sellaista ulostuloa, jota ihmiskäyttäjät todellisuudessa tarkoittaa. Tästä ihmisen tarkoittamasta intentiosta käytetään nimitystä *alignment*. Siinä missä ChatGPT:lle esitettyihin kysymyksiin on mahdollista saada järkevän oloisia vastauksia jopa suomeksi, FinGPT tuottaa kyllä kieliasultaan muodollisesti oikeaa suomea mutta vailla järjen hiventä. Osaltaan tämä voi johtua suomenkielisen opetusaineiston rajallisesta saatavuudesta englanninkieliseen verrattuna. GPT-3:n ja GPT-4:n kaltaisen suuren kielimallin opettaminen voidaan jakaa kolmeen päävaiheeseen (Stöffelbauer 2023):

1. **Esiopetus** (*pretraining*), jossa luodaan tilastollinen malli suurella määrällä raakadataa. Tämä oppiminen on ohjaamatonta (*unsupervised*), tarkemmin määriteltynä itseohjautuvaa (*self-supervised*)
2. **Ohjehienosäätö** (*instruction fine tuning*), jolla mallia opetetaan luokitellulla datalla noudattamaan saamia ohjeita ihmisen toivomalla tavalla. Tämä oppiminen on ohjattua (*supervised*)
3. **Vahvistusoppiminen ihmispalautteella** (*reinforcement learning from human feedback*), jolla edelleen parannetaan mallin kykyä noudattaa saamia ohjeita ja tuottaa ihmisen toivomia ulostuloja

Vahvistusoppiminen on siis jatkoa hienosäädölle, ja yhdessä näiden avulla tilastollinen malli saadaan vastaamaan ihmisen tarkoitusperiä tai intentiota (*align with human intention*). Kielimallin opettamisen vaiheet on esitetty kuvassa 23.



Kuva 23. Suuren kielimallin opettaminen opetusaineistosta sovellukseksi.

Suuren kielimallin muokkaamiseen ihmisen toiveita vastaavaksi ja perinteistä hienosäätöä ja vahvistusoppimista kevyempiin tekniikoihin palataan luvussa 5.4. Vahvistusoppimisen ja hienosäädön tarkoitus ei ole pelkästään tuottaa ihmisen kielimallille antamiin tehtäviin mahdollisimman tarkkoja ja osuvia tuloksia: niillä pyritään myös karsimaan ja estämään epätoivottuja vastauksia. Suositun ChatGPT-keskustelubotin vastauksista on karsittu julkaisijan epäeettiseksi tai loukkaavaksi katsomaa materiaalia (OpenAI 2023), tosin tällaisella moderoinnilla on vastareaktionsa ja verkkohauilla löytyvät *miten teet* -oppaat (*how-to*) antavat yksityiskohtaisia neuvoja rajoitusten kiertämiseen. Niin ammattilaiset kuin harrastelijatkin testaavat näitä rajoituksia: on raportoitu, että ChatGPT on saatu tuottamaan erilaisia haittaohjelmia ja kalastelukirjeitä, vaikka tällaisten tuottaminen pitäisi nimenomaan olla estetty (Marcelline 2023).

Kielimallin onnistuneisuus ja osuvuus on riippuvainen aineistosta, jolla se on opetettu, ja koska opetusaineiston on oltava valtavan suurta, sen laadukkuuden varmistaminen ei ole yksinkertaista. Laadukkuus on myös subjektiivista: kielioopin lisäksi asiasisältö voi tehdä aineistosta laadukkaan tai huonolaatuisen, mutta laadukkaan aineiston erottaminen on vaikeaa. Otetaan esimerkiksi sairauksien hoito-ohjeet: vaivaan kuin vaivaan löytyy monta, usein päinvastaista näkemystä. Kun sovelluksen halutaan antavan vain oikeaa faktatietoa, faktojen valinta jääkin paradoksaalisesti laatijan varaan.

Edellä kuvattu moderointi on eettisen debatin asia, mutta lopulta suurten kielimallien käyttö ja hyödyntäminen päätyvät samaan johtopäätökseen, jota tässä tutkielmassa korostetaan toistuvasti tekoälyn kanssa: tärkeintä on ymmärtää, mitä kielimallilla voi tehdä ja mitä ei, ja miksi näin on. Aivan kuten transformereita käsitellessä todettiin (luku 4.4.3), olennaista on ymmärtää, mihin tekoäly ja suuret kielimallit soveltuvat. Yksi huomionarvoinen heikkous suurilla kielimalleilla on: esiopettamisen jälkeen kielimalli ei enää päivity, vaikka neuroverkko oppisikin käyttäjien toiminnan perusteella. Alkuperäisen ChatGPT:n opetus päättyi vuoteen 2021, joten sillä ei ole kykyä vastata kysymyksiin myöhemmistä tapahtumista. Sittemmin rajoitetta on loivennettu integroimalla bottiin hakukoneen ominaisuuksia, jolloin botti voi linkittää käyttäjälle verkosta löytämäänsä tietoa. Jos sovellus kuitenkin esittää omaa faktatietoa jäädyttämisen jälkeisistä tapahtumista, kyseessä on kielimallin hallusinointi.

Hallusinointi on suurten kielimallien lieveilmiö, mahdollisesti jopa transformer-arkkitehtuurin sisäsyntyinen ominaisuus, kuten Peng et al. esittävät (2024, s. 1–2). Hallusinoinnissa kielimalli esittää totena täysin keksittyä, jopa absurdia tietoa (Naveed et al. 2023, s. 33). Hallusinoinnilla voidaan myös viitata tilanteeseen, jossa kielimallille esitetyt jatkokysymykset ovat ristiriidassa aiempien vastausten kanssa tai eivät liity annettuun kehoitteeseen (ibid.). Esimerkiksi ChatGPT on saatu esittämään keksitylle tiedejulkaisulle täysin keksittyjä lähdeluetteloita julkaisutietoineen, tosin on syytä huomata, että tässä yhteydessä kyseessä oli aiemmin mainitun moderoinnin kiertäminen tutkimustarkoituksessa (Teubner et al. 2023, s. 97). Myös hallusinoinnin yhteydessä korostuu tarve ymmärtää tekoälyn sovellusalueita ja rajoitteita.

Edellä on käsitelty suuria kielimalleja ja niiden opettamista. Seuraavaksi paneudutaan suurten kielimallien rakenteeseen.

5.2 Suurten kielimallien rakenne

Suuret kielimallit pohjautuvat yleensä transformer-arkkitehtuuriin, joka esiteltiin luvussa 4.4. Käsittelyn painopiste on OpenAI:n GPT-kielimalleilla ja erityisesti GPT-3:lla, johon suomenkielinen kielimalli FinGPT perustuu (Luukkonen et al. 2023, s. 1).

Ennen yleistä suurten kielimallien rakennusosasten esittelyä on syytä kerrata tärkeimmät havainnot transformereiden ja neuroverkkojen opettamisesta. Alkuperäinen transformer perustuu enkooderi-dekooderi-rakenteeseen (kuva 19), tosin suurissa kielimalleissa on myös pelkästään enkooderiin tai pelkästään dekooderiin perustuvia ratkaisuja, jotka niin ikään kykenevät suoriutumaan luonnollisen kielen käsittelyyn liittyvistä tehtävistä. Muun muassa kielimalli GPT-3 ja sen seuraajat ovat perustuneet pelkkään dekooderiin. (Naveed et al. 2023, s. 24) GPT-3:n opettamisessa on käytetty 175 miljardia (*175 billion* tai 175B) parametria, kun taas FinGPT:n opettamisessa on käytetty 13 miljardia parametria (13B) (Luukkonen et al. 2023, s. 1; Naveed et al. 2023, s. 9).

Suomalaisen mallin suhteessa pientä kokoa selittää se, että tutkimusryhmällä ei yksinkertaisesti ollut enempää opetusdataa saatavilla, vaan kaikki sähköiset suomenkieliset tekstit oli jo käytetty. Tuolloin arvioitiin, että tätä paremman mallin tuottaminen edellyttäisi peräti kymmenien miljardien sanojen edestä uutta tekstiä. (Hallamaa 2023b) FinGPT on kuitenkin selkeästi suuri kielimalli, jolle rajana voidaan pitää miljardia parametria (1B) (Stöffelbauer 2023).

Token ja embedding ovat suurten kielimallien olennaisia rakennusosasia. Niitä käsiteltiin jo transformereiden syötteen muodostumisen yhteydessä luvussa 4.4.1, mutta tarkastellaan molempia vielä hieman kielen muodostumisen ja merkityksen näkökulmasta.

5.2.1 Token

Kielimalli jakaa saamansa syötteen oman koodausmallinsa (*encoding scheme*) mukaisesti tokeneiksi. Syöte voi olla tekstiä tai etenkin myöhemmissä kielimalleissa myös muuta dataa, kuten kuvia.

Kuvassa 24 on esimerkki tokenoinnista, joka on toteutettu OpenAI:n tokenointityökalulla osoitteessa <https://platform.openai.com/tokenizer>. Kyseisellä työkalulla voi tokenoida tekstiä siten, kuin tokenointi on GPT-kielimalleissa toteutettu. Englanninkielinen lause "Unknown vehicles sighted en route to field hospital" sisältää kielimallin GPT-3 mukaisella tokenoinnilla 51 merkkiä ja yhdeksän tokenia. Näiden tokenien saamat numeroarvot ovat [20035, 5672, 6504, 276, 551, 6339, 284, 2214, 4436].

Tokens	Characters
9	51

Unknown vehicles sighted en route to field hospital

Kuva 24. Englanninkielisen lauseen tokenointi.

Kuvassa 25 on esitetty vastaavan suomenkielisen lauseen tokenointi. Suomenkielinen lause ”Tuntemattomia ajoneuvoja havaittu matkalla kenttäsairaalaan” sisältääkin nyt 59 merkkiä ja peräti 28 tokenia. Näiden tokenien saamat numeroarvot ovat [51, 2797, 368, 1078, 296, 544, 257, 73, 505, 84, 13038, 6592, 387, 6862, 715, 84, 2603, 74, 30315, 479, 298, 83, 11033, 82, 958, 64, 6081, 272].

Tokens	Characters
28	59

Tuntemattomia ajoneuvoja havaittu matkalla kenttäsairaalaan

Kuva 25. Suomenkielisen lauseen tokenointi.

Yksinkertainen esimerkki kuvastaa hyvin tokenoinnin haasteellisuutta. Englanninkielisestä lauseesta voi helposti huomata, että tokenit tässä yhteydessä kuvaavat merkityksellisiä osia: itse asiassa ainoa sana, joka on jaettu useampiin tokeneihin, on sana ”sighted”, jossa imperfektin tunnus *-ed* on erotettu pääverbistä (*to*) *sight*. Toisessa ääripäässä suomen sana ”kenttäsairaalaan” on jaettu peräti yhdeksään tokeniin, eikä toke-neiksi jako perustu tavurajoihin tai sijapäätteisiinkään: jo yhdyssanan alkuosassa ensimmäinen tavu *kent-* on eroteltu keskeltä tokeneihin *k* ja *ent*, ja sanaa *matkalla* ei ole jaoteltu sijapäätteen mukaan *matka + lla* (yksikön adessiivi) vaan *mat + k + alla*.

Lempinen (2024) huomauttaa, että nimenomaan suomalaisia – suomen kieltä ja suomalaista perspektiiviä ymmärtäviä – kielimalleja tarvitaan, koska analyyttisen englannin kanssa hyvin suoriutuvat kielimallit eivät välttämättä kykene suoriutumaan suomen kaltaisen agglutinoivan kielen käytöstä. Kielimallit ennustavat tokeneiden yhteensopivuuden todennäköisyyksiä, ja liian pienet tai lukuisat tokenit heikentävät mallin varmuutta. Lempinen edelleen esittää, että toimimaton tokenointi voi myös lisätä kielimallin hallusiointia.

Tokenointi vaikuttaa mallin luotettavuuden ja toimivuuden ohella myös kustannustasoon. Tokenin tai enimmillään muutaman tokenin ennustaminen kerrallaan vaatii koko verkon läpi etenevän laskentasyklin. Palveluina tarjottavat kielimallit kuten GPT hinnoitellaan tyypillisesti tokeneiden lukumäärän mukaan, ja niiden käyttämät, englannille optimoidut

tokenoijat pilkkovat muut kielet huomattavasti pienempiin osasiin (Petrov et al. 2023, s. 1–2, 4, 7). Kun edellä testattiin alkuperäisen ChatGPT:n ja kielimalli GPT-3:n käyttämää tokenoijaa suomen ja englannin lauseille, huomattiin merkittävä ero saatujen tokeneiden määrässä. Pienillä kielillä myös käytettävissä olevan opetusdatan määrä on paljon pienempi kuin englannilla. Tämä huomattiin jo kotoisan FinGPT:n yhteydessä: mallin opettaminen lopetettiin, kun laadukasta opetusdataa ei yksinkertaisesti enää ollut saatavilla (Hallamaa 2023b). Suurten kielimallien luominen ja käyttäminen pienille ja harvinaisille kielille voi siten olla huomattavan kallista.

Vielä eräs huomionarvoinen heikkous liittyy tokenoinnin toimintaan: Kielimallin tokeneiden suuri määrä kasvattaa kielimallia hyödyntävien sovellusten latenssia (Petrov et al. 2023, s. 7–8). Maailmassa, joka alati reaaliaikaistuu ja toimii yhtä enemmän älypuhelin-ten varassa, viiveeseen liittyvät huolenaiheet ovat merkittäviä. Jos eivät vielä, niin tulevaisuudessa.

5.2.2 Embedding

Embeddingeihin upotetaan kielimallin ymmärrys – nyansseja ja kontekstia jäljittelevä matemaattinen kuvaus – kielen rakenteesta. Ajatellaan havainnollisuuden vuoksi, että yksi embedding kuvaa yhtä sanaa. Tällöin yhdessä embeddingissä on satoja tai jopa tuhansia parametreja, ja kukin parametri muodostaa oman ulottuvuutensa d -ulotteisessa koordinaatistossa. Jos opetusdata koostuu tekstistä, kukin tekstin token tai tässä tapauksessa sana muodostaa oman embeddinginsä samaan koordinaatistoon. Lähekkäin koordinaatistossa olevat parametrit liittyvät todennäköisesti toisiinsa, ja tilastollinen malli on ennen kaikkea todennäköisyyksien laskemista. (Janakiram 2024)

Havainnollisia ja hauskoja esimerkkejä varsin abstraktista konseptista esittävät Vylomova et al. (2015, s. 1). Otetaan kaksi moniulotteista embedding-vektoria, joista toisen merkitys on *kuningas* ja toisen *kuningatar*. Vektoriin *kuningas* on tallennettu parametrit eli ominaisuudet *hallitsija* ja *mies*. Vektoriin *kuningatar* on tallennettu parametrit *hallitsija* ja *nainen*. Vektori *kuningatar* voidaan ennustaa yksinkertaisella laskutoimituksella

$$\textit{kuningas} - \textit{mies} + \textit{nainen} \approx \textit{kuningatar}. \quad (24)$$

Toisessa esimerkissä laskutoimituksella

$$\textit{pariisi} - \textit{ranska} + \textit{puola} \approx \textit{varsova} \quad (25)$$

voidaan ennustaa pääkaupunkeja, ja kolmannessa laskutoimituksella

$$\textit{autot} - \textit{auto} + \textit{omena} \approx \textit{omenat} \quad (26)$$

voidaan ennustaa monikollisuutta. Tällainen semanttinen ymmärrys on ihmiselle helppoa ja luontevaa, kuten kuka tahansa Alias-sanaselityspeliä pelannut tai englanninkielisiä sanantäydennystehtäviä ”king is to queen as man is to ___” tehnyt tietää. Kielimalleilla tätä inhimillistä ymmärrystä voidaan jäljitellä odottamattomankin osuvasti. Luonnollisesti todelliset embeddingit sisältävät valtavasti enemmän ominaisuuksia kuin nämä esimerkit, mutta ne auttavat hahmottamaan ajatusta toisiinsa liittyvien asioiden samankaltaisuuksista.

5.3 Suurten kielimallien opettaminen

Koneoppimisen ja samalla syväoppimisen tekniikoista on aiemmin luvussa 4.1 esitelty **ohjattu oppiminen, ohjaamaton oppiminen ja vahvistusoppiminen**. Näiden rinnalla neuroverkkojen opettamisessa tärkeää on **osittain ohjattu oppiminen** (*semi-supervised learning*, SSL), jota käytettiin myös alkuperäisen transformerin opettamisessa (Vaswani et al. 2017, s. 9). Osittain ohjattu oppiminen on ohjatun ja ohjaamattoman oppimisen välimuoto: koneelle annetaan sekä luokiteltua aineistoa – siis esimerkkejä syöte-tuloste-pareista – että luokittelematonta aineistoa, ja koneen annetaan muodostaa malli kohinaisesta, alustavasti luokitellusta aineistosta (Russell & Norvig 2016, s. 695). Suurten kielimallien esiopettamisessa puolestaan käytössä on **itseohjautuva oppiminen** (*self-supervised learning*, myös SSL), joka on täysin ohjaamatonta oppimista (Stöckbauer 2023).

Edelleen neuroverkon opettamista voidaan jakaa hienompiin osasiin. Kielimallien opettamisessa tärkeää on luokitteluongelmissa hyödynnetty **muutaman näytteen oppiminen** (*few-shot learning*, FSL), joka sijoittuu osittain ohjatun oppimisen ja ohjaamattoman oppimisen välimaastoon (Ahuja 2023). Muutaman näytteen oppimista ja edelleen hienosäätöä käytettiin GPT-3-kielimallissa (Brown et al. 2020, s. 1), joka oli GPT-3.5:n ja siihen pohjautuneen ensimmäisen ChatGPT:n edeltäjä. Hienosäätöön ja muihin opetusmetodeihin, joilla kielimallia räätälöidään tiettyyn tehtävään sopivaksi, palataan luvussa 5.4. Luokittelussa voidaan käyttää myös **näytteetöntä oppimista** (*zero-shot learning*, ZSL), jolloin koneelle ei anneta minkäänlaisia valmiiksi luokiteltuja esimerkkejä mallin muodostamiseksi, tai **yhden näytteen oppimista** (*one-shot learning*, OSL), jolloin koneelle annetaan vain yksi esimerkki kustakin luokasta mallin muodostamiseksi (Murphy 2022, s. 651).

Muutaman näytteen oppiminen edellyttää asianmukaista opetusdataa mallin luomiseksi. Aiemmin luvussa 4.4.3 on todettu, että transformer-arkkitehtuuri kärsii herkästi opetusdatan puutteesta. Sama heikkous on periytynyt myös transformereihin perustuville suurille kielimalleille. Ongelmaa voidaan helpottaa **siirto-oppimisella** (*transfer learning*),

jossa laajalla opetusaineistolla esiopetettu kielimalli voidaan hienosäätää valittua loppupään tehtävää (*downstream task*) varten pienehköllä luokitellulla opetusaineistolla (Murphy 2022, s. 627). Siirto-oppiminen voidaan luonnollisesti toteuttaa laajallakin luokitellulla opetusaineistolla, jos sellaista on käytettävissä. Laajaa opetusaineistoa on tyypillisesti käytettävissä kuvantunnistustehtävissä, joissa alkuperäisestä opetusaineistosta saadaan kierroilla ja skaalauksilla edelleen tunnistettavissa olevia kuvia mallin käyttöön (ibid., s. 629–630).

Pienillä kielialueilla ja erikoisaloilla käytettävissä olevan tekstiaineiston määrä on kuitenkin yleensä rajallinen, ja tällöin siirto-oppiminen tulee kyseeseen. Siirto-oppimisessa malli opetetaan toisintamaan sen yhtäällä oppimaa lopputulosta ensin testiaineistoon ja myöhemmin täysin uuteen aineistoon toisaalla. Esimerkiksi *cloze*-testit (*cloze tasks*) eli tekstin aukkotäydennystehtävät tutulla saatelauseella ”Täydennä puuttuva sana” ovat hyvin tyypillisiä siirto-oppimisella opetettavia tehtäviä (Murphy 2022, s. 630). Mallille annetaan opetusmerkkejä, joissa puuttuva sana on täydennetty valmiiksi, minkä jälkeen sille annetaan tehtäväksi toteuttaa sama toiminnallisuus uuteen aineistoon samaa täydentämistä pyytävää kehotetta käyttäen. Myös aiemmin mainittu näytteetön oppiminen on siirto-oppimista.

Seuraavaksi palataan tekstin peittämiseen sanankorvauksella, mutta nyt aiemman ohjelmallisen toteutuksen sijaan esitellään mahdollisia tapoja toteuttaa sama toiminnallisuus tekoälyn ja suurten kielimallien avulla. Samalla palataan laadittuun peittämisohjelmaan, jota voidaan nyt hyödyntää opetusaineiston tuottamisessa kielimallia varten.

5.4 Suuren kielimallin opettaminen omalla aineistolla

Suurten kielimallien opettaminen on työlästä ja vaatii merkittäviä resursseja. Tämä tässäkin tutkielmassa toistettu havainto on edelleen todenmukainen ja paikkansapitävä arvio, mutta ei koko totuus. Ensin laajan kielimallin laskeminen vaatii kuukausia supertietokoneen tai rinnakkain kytkettyjen supertietokoneiden laskentatehoa, minkä jälkeen mallin hienosäätö ja halutun toiminnan vahvistaminen voidaan vasta alkaa, jotta mallia voidaan käyttää mielekkäällä tavalla loppupään tehtävien toteuttamiseen.

Siinä missä mallin laskenta on edelleen joko suurten kaupallisten toimijoiden tai tieteellisten tutkimusyhteisöjen varassa, mallin opettaminen tarkkaan määriteltyihin tehtäviin on mahdollista jopa yritteliäälle harrastajalle, kunhan laskettu kielimalli on saatavilla. Esi-tetyt tekniikat ovat sellaisia, jotka on suunniteltu toteutettavaksi vapaasti saatavilla olevan FinGPT:n tai mahdollisesti Poron kanssa. Avainasemassa tällaisessa pienimuotoisessa mutta tarkkarajaisessa toteutuksessa on siirto-oppiminen, sillä kielimallia ei ole

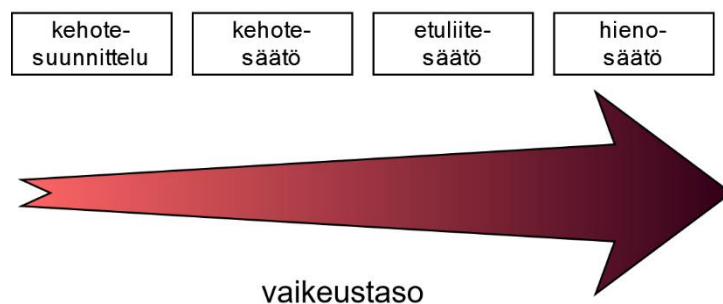
esiopetettu tai vahvistusopetettu ihmispalautteella sanankorvauspeittämiseen. Näin ol-
len opetusvaiheessa joudutaan antamaan esimerkkejä halutuista syöte-tuloste-pareista
eli selkoteksti-peiteteksti-lausepareista.

Opetusdatan lisäksi tarvitaan testidataa, jolla opetetun mallin oikeanlaisesta toiminnasta
voidaan varmistua. Esimerkkejä oikeista tulosteista eli peiteteksteistä voidaan toteuttaa
laaditulla peittämishjelmalla, sillä ohjelma osaa taivuttaa merkittävää määrää suomen
sanoja annetussa sijassa ja muodossa. Tarvittavan aineiston luominen käsin ei olisi rea-
listista tai järkevää, mutta toteutettua ohjelmaa voidaan käyttää tuottamaan suuri määrä
peitelauseita kerralla yksittäisten peitelauseiden sijaan, jolloin kielimallille saadaan riittä-
västi sekä opetusaineistoa että testidataa. Tutkielman olennaisin käytännön osuus on jo
toteutettu peittämishjelmalla laadittaessa, joten esitetyt opetustekniikat esitetään pää-
osin teoreettisina vaihtoehtoina.

Soveltuvia tekniikoita kielimallin opettamiseen omalla aineistolla on neljä. Helpoimmasta
vaikeimpaan nämä ovat:

1. **Kehotesuunnittelu** (*prompt engineering*)
2. **Kehotesäätö** (*prompt tuning*)
3. **Etuliitesäätö** (*prefix tuning*)
4. **Hienosäätö** (*fine tuning*)

Lukuun ottamatta ensimmäistä vaihtoehtoa, keחותsuunnittelua, erityyppiset säätötek-
niikat edellyttävät aina opetusaineiston käyttöä, ja myös keחותsuunnittelussa mallille
annettava esimerkit ovat hyödyllisiä. Säätötekniikoiden käyttö on kokonaisuudessaan
keחותsuunnittelua vaativampaa, kuten havainnollistettu kuvassa 26.



Kuva 26. Kielimallin säätämisen tekniikat helpoimmasta vaikeimpaan.

Käydään seuraavassa läpi erilaisia kielimallin opettamiseen soveltuvia tekniikoita. Toisin
kuin luvussa 5.1 esitetty, supertietokoneiden varaan rakentuva suurten kielimallien opet-
taminen ja etenkin raskas hienosäätö, kevyempiä tekniikoita voi olla mahdollista toteut-
taa jopa kannettavalla tietokoneella, etenkin maltillisen kokoisilla kielimallin versioilla.

5.4.1 Kehotesuunnittelu

Kehotesuunnittelu (*prompt engineering*; myös *prompt design* tai *priming*) viittaa kehoitteiden laatimiseen tavalla, jolla kielimalli todennäköisimmin antaa haluttuja tuloksia. Kehotesuunnittelussa käytössä ovat kovat kehoitteet (*hard prompts*), toisin sanoen tekstimuodossa annettavat, ihmiselle ymmärrettävät käskyt ja esimerkkitulosteet, joiden mukaan mallin halutaan toimivan. Itse malli ja neuroverkon parametrit pysyvät muuttumattomana. Kehotesuunnittelun etuna voidaan pitää sitä, että kielimallia ei tarvitse toisintaa kutakin loppupään tehtävää varten erikseen. Suuret kielimallit ovat nimensä mukaisesti suuria ja niiden koko on kasvanut harppauksin – hyvänä esimerkkinä GPT-3:n 175 miljardista (175B, 10^9) parametrilla GPT-4:n väitetyksi lähes 1,8 biljoonaan (1.8T, 10^{12}) parametriin alle kolmessa vuodessa – joten on mielekäästä käyttää samaa mallia useisiin tehtäviin. (Lester 2021, s. 1; Guinness 2023; Grammarly 2024)

Kehotteessa on tyypillisesti tehtävän kuvaus ja mahdollisesti esimerkkejä halutuista lopputuloksista (Lester 2021, s. 1). OpenAI (2024) nimeää oppaassaan useita strategioita, joilla kehoitteiden laatua voi parantaa. Parhaan lopputuloksen saavuttaa yhdistelemällä eri strategioita ja ennen kaikkea kokeilemalla. Tekoäly oppii, mutta niin oppii tekoälyn kanssa toimiva ihminenkin. Hyvä kehoite

- on täsmällinen. Se määrittelee tarkasti lopputuloksen halutun muodon (haikurunomitta, CSV-taulukko, binääriluku, amerikanenglanti) ja antaa esimerkkejä samankaltaisista tuotoksista.
- pyytää tarkkaan eriteltyjä lähteitä ja tarkkoja sitaatteja lähteistä, jos vaarana on, että kielimallin vastaus voisi sisältää hallusinoitua.
- pilkkoo monimutkaiset kokonaisuudet pieniin osatehtäviin, jolloin kielimalli todennäköisemmin päätyy oikeaan lopputulokseen. Hyvässä tehtävässä kielimalli voi hyödyntää osatehtävien tuloksia lopullisen vastauksen kokoamisessa.
- johdattelee kielimallia ja pyytää tulosten tarkistamista, etenkin silloin, kun mallia pyydetään suorittamaan tehtäviä, joissa kielimallit ja transformer-arkkitehtuuri ovat heikoimmillaan: täysin uudet laskutehtävät, looginen päättely ja järkeily (luku 4.4.3). Tällainen ajatuksenkulkua (*chain of thought*, CoT) jäljittelevä, johdatteleva kehoitteiden käyttö parantaa merkittävästi kielimallien onnistumista näillä perinteisesti heikoilla osa-alueilla (Wei et al. 2022b, s. 1–9) ja myös vähentää niiden hallusinoitua (Peng et al. 2024, s. 2).
- ohjaa kielimallin tarvittaessa hyödyntämään ulkopuolisia lähteitä: tietokantoja, koodaus- tai kirjoitustyökaluja ja ohjelmarajapintoja (*application programming*

interface, API). Tätä generatiivisen tekoälyn metodia kutsutaan haulla parannetuksi luomiseksi (*retrieval-augmented generation*, RAG).

- testaa välillä mallia, jotta varmistetaan, että käytetyt kehotteet varmasti parantavat saatuja tuloksia kokonaisuudessaan, eivät vain yksittäisinä onnistuneina tulosteina. Tämä voidaan toteuttaa pyytämällä tarkkaan määritelty selonteko jostain yleisesti tunnetusta faktasta, kuten Kuuhun laskeutumisesta (kuka, mitä, missä?) ja vertaamalla saatua vastausta ihmisen laatimaan esimerkkivastaukseen. Jos malli ei palauta oikeanlaista vastausta kaikkiin kysymyksessä esitettyihin osakysymyksiin kyllin suurella toistomäärällä (vähintään kymmenen mutta jopa 10 000 toistoa hyvän tilastollisen luotettavuuden saamiseksi), malli ei toimi hyvin. Tämä strategia sopii tarkistuksiin automatisoiduissa ympäristöissä silloin, kun halutaan arvioida osien summaa ennemmin kuin osia. Toisin kuin muut esitetyt strategiat, jotka selvästi hyötyvät ihmisen harkinnasta ja kokonaisuuksien hallinnasta, testaus nojaa automatisointiin jo toistojen suuren määrän vuoksi.

Esimerkkejä kielimallille annetuista kehotesuunnittelun mukaisista hyvistä kehotteista voivat olla vaikkapa seuraavat (mukaillen OpenAI 2024):

Tiivistä triplalainausmerkkien sisällä annettu teksti 50 sanaan.

```
""" teksti """
```

Kuinka pääsen Helsingistä Düsseldorfiin mahdollisimman halvalla? Listaa käytetyt liikennevälineet kullakin etapilla, matkaan kuluva aika ja arvioitu kokonaismatka-aika.

Nimeä viisi ihmiskunnan merkittävintä keksintöä 1900-luvulla ja perustelee, miksi ne ovat merkittäviä.

Käännä triplalainausmerkkien sisällä annettu teksti suomesta englantiin.
""" käännettävä teksti """

Samaa toimintaa toistavissa tehtävissä, kuten lasku- tai sanastotehtävissä, voidaan käyttää yhden näytteen kehotteita (*one-shot prompting*) näin (mukaillen DAIR Institute 2024):

Käännä suomesta englantiin: juusto -> cheese

Käännä suomesta englantiin: leipä ->

tai muutaman näytteen kehotteita (*few-shot prompting*) näin:

Laske: 3+5=8

Laske: 8+100=108

Laske: 5+12=17

Laske: 6+4=

Yhden näytteen ja muutaman näytteen kehoitteet ovat tapa toteuttaa luvussa 5.3 esiteltyä yhden näytteen ja muutaman näytteen oppimista (*one-shot learning*, *few-shot learning*). Samoin näytteetön oppiminen (*zero-shot learning*) näytteettömän kehoitteen avulla (*zero-shot prompting* tai *direct prompting*) avulla on mahdollista. Näytteetön kehote sisältää ainoastaan toteutettavan tehtävän ilman esimerkkejä halutuista ulostuloista. Etenkin näytteettömän kehoitteen onnistunut käyttö kuitenkin edellyttää, että mallin esiopeutusvaiheessa (*pretraining*) käytetty aineisto on sisältänyt vastaavanlaisia koulutus-esimerkkejä. (Microsoft Learn 2024)

Kehotesuunnittelu ja kehoitesuunnittelijana (*prompt engineer*) toimiminen nousivat kiinnostuksen kohteeksi vuonna 2022, ja jo puolessatoista vuodessa alkoi vaikuttaa siltä, että sen aika on ohi. Ihmisten laatimien, ihmisen ajattelua vaativien kehoitteiden haastajaksi ja jopa voittajaksi ovat – ironista kyllä – nousseet tekoälyn itse itselleen laatimat kehoitteet. Niiden muotoilu voi olla ihmisen silmään kummallinen tai jopa järjetön, mutta ne toimivat ja antavat hyviä tuloksia. Syynä tähän voi olla se, että koneen itsensä laatimat kehoitteet työstävät paremmin matemaattista mallia, kun taas ihmissuunnittelija pyrkii kielelliseen ja inhimilliseen vuorovaikutukseen, jota tilastollinen malli todellisuudessa ei toteuta, näennäisestä ymmärryksestä huolimatta. (Genkina 2024; Lui 2024) Tällainen *autotune* on ehkä haastanut ihmisen, mutta nähtäväksi jää, onko se korvannut tätä.

Lester et al. (2021, s. 4) huomauttavat, että muutaman näytteen oppimista hyödyntävä GPT-3 tuottaa hyviä tuloksia kehoitteilla. Kehotesuunnittelulla onkin potentiaalia ChatGPT:n kaltaisten, vahvistusoppimisella hienosäädettyjen mallien käytössä. Osuvilla kehoitteilla saadaan mallin potentiaali parhaiten käyttöön. Kehotesuunnittelu ei kuitenkaan sovellu kielimalleihin kuten FinGPT, joiden esiopeutusvaiheessa ei ole käytetty haluttuja tehtäviä mukailevaa opetusaineistoja ja joita ei ole muutenkaan hienosäädetty tai vahvistusopetettu.

Vaihtoehto kehoitesuunnittelulle ovat kielimallien säätötekniikat, joilla mallin ulostuloihin voidaan vaikuttaa syvällisemmin. Tässä avainasemassa ovat opetusdatan käyttö ja malliin liitettyjen lukuarvojen tai mallin parametrien säätö.

5.4.2 Kehotesäätö

Kehotesäätö (*prompt tuning*) on kehoitesuunnittelua vaativampi tapa opettaa kielimalli omaan käyttöön. Kehotesäätö – kuten muutkin säätötekniikat – hyödyntää pehmeitä kehoitteita (*soft prompts*). Kehote on tässä yhteydessä kenties hieman harhaanjohtava nimitys, sillä kyseessä eivät ole pienet näppärät muutaman rivin ohjeistukset, joina

kehoitteita yleensä ajatellaan. Nyt kehote on malliin lisättävä osa, reaaliarvoinen vektori, jonka lukuarvoja opettamisen aikana säädetään.

Kehote voidaan esittää matemaattisesti seuraavasti (Lester et al. 2021, s. 2–3): malli saa syötteenä vektorin \mathbf{X} ja tuottaa ulostulona vektorin \mathbf{Y} . Kehote on siten syötteeseen \mathbf{X} lisätty sarja kehotetokeneita \mathbf{P} , $\mathbf{P} = \{p_1, p_2, \dots, p_n\}$, jonka avulla malli todennäköisimmin tuottaa oikean ehdollisen todennäköisyyden $Pr_{\theta}(\mathbf{Y}|\mathbf{P}; \mathbf{X})$. Tämä todennäköisyys on siis tapa, jolla kielimalli ennustaa järjestyksessä seuraavaa tokenia tai sanaa. Mallin varsinaisia parametreja θ ei kehoitteita käytettäessä muuteta. GPT-3:ssa kehotetokenit \mathbf{P} ovat osa transformerin embedding-taulukkoa.

Kehotesäädössä todennäköisyyden laskeminen muuttuu. Halutun ulostulon \mathbf{Y} todennäköisyys saadaan vastavirta-algoritmilla päivittämällä parametreja θ_p gradienttimenetelmällä, ja laskettava ehdollinen todennäköisyys on nyt $Pr_{\theta, \theta_p}(\mathbf{Y}|\mathbf{P}; \mathbf{X})$.

Lester et al. (2021, s. 3) esittävät kehoitesäädön käyttämisen T5-kielimallilla, joka GPT:n tavoin pohjautuu transformer-arkkitehtuuriin. Järjestelmälle syötetään n kappaletta tokenia, $\mathbf{x} = (x_1, x_2, \dots, x_n)$. Tokeneihin lisätään embedding, jolloin muodostuu matriisi $\mathbf{X}_e \in \mathbb{R}^{n \times e}$, jossa n on tokeneiden lukumäärä ja e on embedding-avaruuden ulottuvuus. Kehote konkatenoidaan eli yhdistetään tähän matriisiin, jolloin saadaan kehoteparametrit \mathbf{P}_e ja embedding-token-matriisin sisältävä yhteismatriisi $[\mathbf{P}_e; \mathbf{X}_e] \in \mathbb{R}^{(p+n) \times e}$. Todennäköisyyttä \mathbf{Y} laskiessa vain mallin syötteeseen lisättyjä kehoteparametreja \mathbf{P}_e päivitetään.

Lester et al. (2021, s. 1) päätyivät tutkimuksessaan siihen, että kielimallin koon ja mallin parametrien määrän kasvaessa 10 miljardiin voidaan kehoitesäädöllä saavuttaa yhtä hyviä tuloksia kuin koko mallin hienosäädöllä. Merkittävänä etuna yhtä mallia voidaan tällöin edelleen käyttää useisiin loppupään tehtäviin, kun taas hienosäädön ideana on luoda samasta kielimallista itsenäinen kopio kutakin tehtävää varten ja opettaa se laajalla aineistolla suoriutumaan juuri kyseisestä tehtävästä. Hyvä suorituskyky, monikäyttöisyys ja kokonaisuudessaan dramaattisesti pienemmät prosessointikapasiteetin varastointikulut tekevät kehoitesäädöstä ymmärrettävästi erittäin houkuttelevan vaihtoehdon hienosäädölle. Saman toiminnallisuuden toteuttaminen vaati tutkimuksessa kehoitesäädössä 20 tuhatta parametria ja hienosäädössä 11 miljardia. Ero siis yli viisi suuruusluokkaa (*orders of magnitude*, 10^5), kun parametrin koko oli viisi tokenia (Lester et al. 2021, s. 2).

On kuitenkin huomattava, että kehoitesäätö kuroo hienosäädön suorituskyvyn eron umpeen vasta silloin, kun mallin koko on vähintään mainittu 10 miljardia (10^{10}) parametria. Jos mallin koko on noin miljardi parametria (10^9), mitä pidetään suuren kielimallin rajana (Stöffelbauer 2023), hienosäädetty malli suoriutuu merkittävästi kehoitesäädettyä mallia

paremmin. Kehotesuunnittelun ja kovien kehoitteiden hyödyntäminen puolestaan tuottaa merkittävästi heikompia tuloksia mallin koosta riippumatta. (Lester et al. 2021, s. 1)

Kehotesäädön onnistuminen ja suorituskky erilaisissa luonnollisen kielen käsittelyn tehtävissä korreloi suoraan säädön suunnittelussa tehtyihin valintoihin. Parhaita tuloksia yleisesti antavat pitkät, yli 20 tokenin kehoitteet, ja parametrien alkuarvojen asettaminen opetusdatasta johdettuihin arvoihin satunnaistuksen sijaan. Kuitenkin suurimpana tekijänä kehoitesäädetyin mallin toimivuudessa on alkuperäisen mallin koko: melkein minkä hyvänsä kehoitesäädön parannuksen voi jopa sivuuttaa sillä, että käsiteltävä kielimalli on yksinkertaisesti mahdollisimman suuri. (Lester et al. 2021, s. 5–6) Parhaita tuloksia tutkimuksessa antoi T5-XXL, jonka koko oli 11 miljardia parametria ja kehoitesäädössä käytettyjen parametrien määrä yli 600 000. (ibid., s. 17)

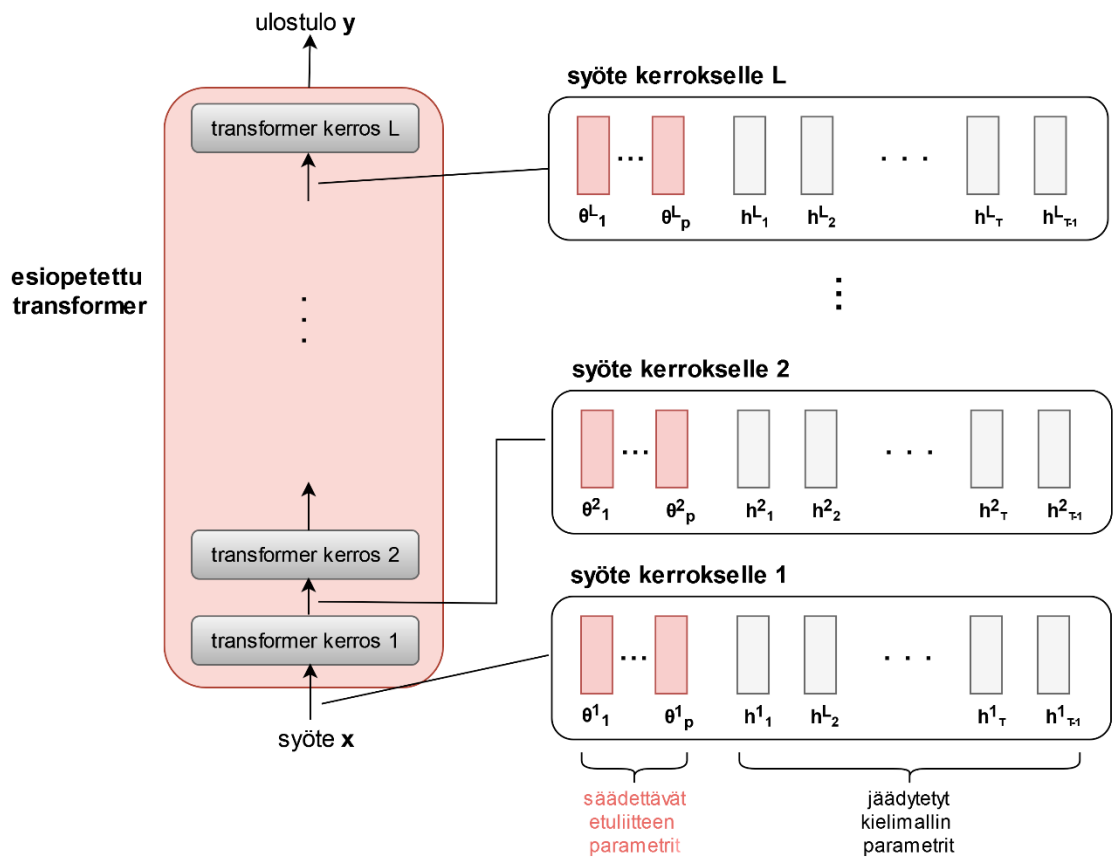
Suoraviivaisesti voi ajatella tämän perusteella, että FinGPT:n 13 miljardia parametria (Luukkonen et al. 2023, s. 1) on riittävän kookas malli, että kehoitesäätöä voitaisiin hyödyntää mallin opettamisessa tekstisteganografiaan siitä missä kielimallien parhaana pidettyä tekniikkaakin, hienosäätöä. Samoin Lester et al. (2021, s. 2) tutkimuksessa on lähtökohtana käytetty samaan tapaan esiopetettua (*pretrained*) mutta loppupään tehtäviin hienosäätämätöntä mallia, jollaisena myös FinGPT on kuvattu (Luukkonen et al. 2023, s. 1, 9). Kehotesäädön kanssa toimiessa selvää on myös välttämätön tarve tuottaa omaa opetusaineistoa ja testidataa, mikä voi pienen kielialueen ja erikoisalan kohdalla olla haastavaa.

Kuvatun kehoitesäädön (*prompt tuning*) kanssa toisinaan päällekkäisenä sukulaiskäsitteenä esitetään hieman yksinkertaistettu *P-tuning*, jossa päivitettävät parametrit käyvät läpi lyhemmän opetusprosessin. Kummankin säätölajin perusajatus on sama: itse kielimallin parametreja ei muuteta eikä kielimallista tehdä kopioita, vaan ainoastaan säädetään siihen liitettävän kehoitteen parametreja. Tässä tutkielmassa *P-tuning* ei kuitenkaan ole erityisen merkityksellinen, sillä se soveltuu ainoastaan luonnollisen kielen ymmärtämiseen (NLU) liittyviin tehtäviin, ei luonnollisen kielen tuottamiseen (NLG), toisin kuin muut esitetyt tavat opettaa kielimallia. (Li & Liang 2021, s. 1; Sinha 2024)

5.4.3 Etuliitesäätö

Etuliitesäätö (*prefix tuning*) sijoittuu haastavuudessaan kehoitesäädön ja hienosäädön väliin. Tämän säätömenetelmän lähestymistapa on samantapainen kuin kehoitesäädön: mallin parametreja ei muuteta, vaan muokataan ainoastaan malliin liitettyä, kehoiteparametrit sisältävää tehtäväkohtaista reaaliarvoista vektoria (*continuous task-specific vector*), josta nyt käytetään nimitystä etuliite.

Kehotesäädön ja etuliitesäädön erottaa se, että kehoitesäädössä kehoiteparametrivektori lisätään ainoastaan neuroverkon syötekerrokseen embedding-kerroksella (kuva 19), kun taas etuliitesäädössä kehoiteparametrivektorit eli etuliitteet lisätään myös neuroverkon piilokerroksiin. Mallia ei kummassakaan tekniikassa toisinneta eri loppupään tehtäviä varten, vaan ainoastaan kehoiteosasta tehdään eri tavalla säädettyjä kopioita, jolloin datan varastointikustannukset ovat maltilliset. Loppupään tehtäviä voivat olla vaikkapa tekstin tiivistäminen, kääntäminen ja esittäminen taulukon sijaan tekstimuodossa. (Li & Liang 2021, s. 1) Etuliitesäätöä transformerin rakenteessa on havainnollistettu kuvassa 27.



Kuva 27. Etuliitesäädön havainnollistus.

Verkon säätämiseen vaadittu aika ja laskentateho, säädetävien parametrien määrä ja verkon monimutkaisuus kasvavat etuliitesäädössä kehoitesäätöä suuremmaksi, mutta pysyvät edelleen maltillisina hienosäätöön verrattuna (Li & Liang 2021, s. 1–2; Toloka 2024). Jo kehoitesäädöllä voidaan saavuttaa erinomaisia tuloksia etenkin riittävän suurilla kielimalleilla. Etuliitesäätö soveltuukin tilanteisiin, joissa mallinnettava ongelma on niin monimutkainen, etteivät yksinkertaisempi kehoitesäätö tai vain syötekerrokselle liisäetty kehoitevektori riitä tuottamaan mallin saamista syötteistä haluttua vastetta. (Lester et al. 2021, s. 5–6; Toloka 2024) Etuliitesäätö on esitelty Li & Liang (2021) julkaisussa

erityisesti luonnollisen kielen tuottamiseen (NLG) liittyville tehtäville, ja myöhemmin Liu et al. (2022) ulottivat menetelmän myös luonnollisen kielen ymmärtämiseen (NLU), tosin he kutsuivat menetelmää nimellä *P-Tuning v2*. Tässä tutkielmassa käytetään yksinomaan nimitystä etuliitesäätö.

Etuliitesäädössä etuliite lisätään sekä transformerin enkooderiin että dekodeeriin (kuva 19), jolloin saadaan $\mathbf{z} = [\mathbf{ETULIITE}; \mathbf{x}; \mathbf{ETULIITE}'; \mathbf{y}]$ (Li & Liang 2021, s. 4). Tämän avulla alustetaan θ :lla parametrisoitu, opetettava matriisi \mathbf{P}_θ , joka sisältää etuliitteen parametrit. Etuliitteen indeksejä merkitään \mathbf{y}_{in} , ja opetettava matriisi on kooltaan $|P_{idx}| \times \dim(h_i)$, kun itseisarvolla $|P_{idx}|$ tarkoitetaan etuliitteen pituutta. Edelleen määritellään opetettava funktio h_i siten, että

$$h_i = \begin{cases} P_\theta[i, :], & \text{jos } i \in P_{idx} \\ LM_\phi(z_i, h_{h_{<i}}) \text{ muutoin,} & \end{cases} \quad (27)$$

missä *LM* viittaa kielimalliin (*language model*). Varsinaisesti etuliitesäädössä maksimoidaan samaa suurimman uskottavuuden kaavaa kuin hienosäädössä, mutta ainoastaan etuliitteen parametreja θ opetetaan. Kielimallin parametreja ϕ , jotka on parametrisoitu matriisiin \mathbf{P}_θ , ei opeteta. (Li & Liang 2021, s. 4) Suurimman uskottavuuden maksimointi on esitetty hienosäädön yhteydessä (luku 5.4.4).

Etuliitesäädöllä on mahdollista saavuttaa vaikuttavia, osin jopa hienosäätöä parempia tuloksia kielimallin opettamisessa vain häviävän pienellä määrällä parametreja (0,1–2 %) koko mallin hienosäätöön verrattuna. Etuliitteen pituus vaikuttaa säädön onnistumiseen olennaisesti: tutkimuksessa tiettyihin kynnyksarvoihin saakka mallin suoritus parani, mutta liian pitkä etuliite oli laskennallisesti raskas ja peräti heikensi mallin suorituskykyä. (Li & Liang 2021, s. 7)

Suurin kysymys mallin säätötekniikan valitsemisessa lieneekin tarkoituksenmukaisuus: sekä kehotēsäätö että etuliitesäätö soveltuvat nimenomaan luonnollisen kielen tuottamiseen (NLG) liittyviin tehtäviin. Näistä etuliitesäätö valitaan silloin, kun mallinnettava asia on niin monimutkainen, ettei kehotēsäädöllä kyetä saavuttamaan kyllin hyviä tuloksia, mutta kuitenkin niin yksinkertainen, ettei mallin kaikkien parametrien opettaminen hienosäädöllä ole tarpeen. Laskentatehon ja datan varastointivanvaatimukset kasvavat aina, kun säätötekniikoissa siirrytään vaativampiin ratkaisuihin. Vaikka aloitushetkellä resurssitilanne olisi salliva, ennakointi on olennaista: suuren kielimallin pienetkään osat eivät ole vähäisiä, ja useiden pienten osien – kehote- tai etuliitematriisien – opettaminen ja varastointi kumuloituvat herkästi.

5.4.4 Hienosäätö

Hienosäätö (*fine tuning*; myös *model tuning*) on kielimallien opettamisen parhaaksi katsottu käytännö (de facto standard). Vaikka yksinkertaisemmat kehoitesäätö ja etuliitesäätö ovat esittäneet vakuuttavan haasteen hienosäädölle ja osin myös ylittäneet sen suorituskyvyn valikoiduissa tehtävissä, hienosäädön yleinen suorituskyky eri tehtävissä on edelleen vaikuttava ja tavoiteltava. Myös tässä tutkielmassa kevyempien säätötekniikoiden ja kehoitesuunnittelun käyttö on suhteutettu nimenomaan hienosäädön suorituskykyyn.

Kielimallille $p_\phi(y|x)$, jota kuvaavat parametrit ϕ , hienosäätö voidaan esittää seuraavasti (Li & Liang 2021, s. 3). Kaikkien kerrosten konkatenoitu aktivoitiefunktio merkitään h_i , $h_i = [h_i^{(1)}, \dots, h_i^{(n)}]$. Aktivaatiofunktio askeleella i saadaan kaavalla

$$h_i = \text{LM}_\phi(\mathbf{z}_i, \mathbf{h}_{<i}), \quad (28)$$

jossa LM viittaa kielimalliin (*language model*), \mathbf{z} syötteen ja ulostulon konkatenaatioon, $\mathbf{z} = [\mathbf{x}; \mathbf{y}]$, ja $\mathbf{h}_{<i}$ askelta edeltäneiden vasemman puolen aktivaatioiden konkatenaatioon. Seuraavan tokenin todennäköisyys lasketaan käyttäen vektorin \mathbf{h}_i viimeistä kerrosta kaavalla

$$p(\mathbf{z}_{i+1} | \mathbf{h}_{\leq i}) = \text{softmax}(\mathbf{W}_\phi h_i^{(n)}), \quad (29)$$

jossa \mathbf{W}_ϕ on kielimallin sanastoa kuvaava esiopetettu matriisi, toisin sanoen one-hotkoodattu token ID -matriisi (luku 4.4.1). Softmax-funktion kaava on esitetty luvussa 4.2.2.

Hienosäädössä gradienttimenetelmän avulla maksimoitava suurimman uskottavuuden logaritmi (*log-likelihood*) voidaan siten esittää kaavalla

$$\max_{\phi} \log p_\phi(\mathbf{y} | \mathbf{x}) = \sum_{i \in Y_{idx}} \log p_\phi(\mathbf{z}_i | \mathbf{h}_{<i}), \quad (30)$$

jossa Y_{idx} on ulostulon \mathbf{y} indeksi. Käytännössä tämä on sama kuin luvussa 4.3.1 kuvattu logistisen regression eli ristientropian virhefunktion minimointi ohjatussa oppimisessä.

Edellä esitetty soveltuu GPT-tyyppisiin autoregressiivisiin kielimalleihin, joissa aikasarjadataan edelliset havainnot selittävät tulevaa. Saman kehittäjän OpenAI:n BERT-kielimallit puolestaan ovat kaksisuuntaisia (*bidirectional*), eikä niiden opettamista käsitellä tässä.

Hienosäätöä pidetään parhaana vaihtoehtona monimutkaisiin, päättelyä, kontekstia tai nyanssien tulkintaa vaativiin ongelmiin, joiden mallintaminen edellyttää monimutkaista ja mahdollisimman taitavasti ihmisen päättelykykyä mallintavaa neuroverkkoa. Heikkoutena hienosäätö on laskennallisesti raskas ja vaativa toteuttaa, ja kärsii herkästi mallin ylisovittamisesta (luku 4.3.3). Hienosäädetty malli ei ole yleiskäyttöinen: kielimalli

toisinnetaan jokaista loppupään tehtävää varten, ja kukin kopio opetetaan erikseen toteuttamaan juuri kyseistä tehtävää parhaalla mahdollisella tavalla. Hienosäätö, tästäkin erikoistumisesta huolimatta, on kaikkien neuroverkkojen tapaan eräänlainen musta laatikko. Hienosäädetyin mallin toiminta ei välttämättä ole ihmisen ennustettavissa tai epätoivottu toiminta korjattavissa ja estettävissä, vaan etenkin ihmisten kanssa toimivat kielimallit, kuten ChatGPT, vaativat rinnalleen vielä vahvistusoppimista ihmispalautteella.

5.5 Suomen kielen tekoälypohjainen tekstisteganografia

Edellä on esitelty kehote- ja säätötekniikoita suuren kielimallin opettamiseen. Tekniikat vaihtelevat laskennallisesti kevyestä kehotteiden käyttämisestä koko kielimallin ja kaikkien sen parametrien muokkaamiseen. Englanninkielisten kielimallien säätämisestä etenkin pehmeiden kehotteiden ja opetusdatan avulla on saatu hyviä tuloksia. Toisinaan jo pelkkä kielimallin esiopettaminen ilman hienosäätöä toimii erinomaisesti. Itse asiassa hienosäätö voi toisinaan johtaa ilmiöön katastrofaalinen unohtaminen (*catastrophic forgetting*), jossa hienosäädetty kielimalli ei enää kykene toteuttamaan sellaisia loppupään tehtäviä, joista se pelkän muutaman näytteen oppimisen ja esiopettamisen jälkeen vielä suoriutui (Jain et al. 2024).

Ydinkysymys onkin se, soveltuvatko englanninkielisten kielimallien opettamisessa toimivat tekniikat ja havainnot suomenkieliseen materiaaliin. FinGPT on ansiokas kielimalli, mutta sitä ei ole vahvistusopetettu eikä hienosäädetty minkäänlaisiin peittämissä tai sanankorvausta sivuaviin tehtäviin. **Kehotesuunnittelu** ei siten todennäköisesti ole realistinen vaihtoehto suomenkieliseen peittämiseen ja tekstisteganografiaan tällä kielimallilla. Käytettävissä on vain esiopetettu kielimalli, jolla saatujen tulosten oivallisuuteen ei osuvillakaan kehoitteilla pysty vaikuttamaan, jos mallia ei ole opetettu toteuttamaan peittämistä. Hyvänä puolena lähtötilanteessa toki on se, että edellä kuvattu katastrofaalinen unohtaminen ei liene FinGPT:ssä samalla tavalla riski kuin pidemmälle opetetuisissa kielimalleissa.

Luvussa 5.6 on kuitenkin testattu kehotteiden suunnittelua monikielisellä, etenkin englannille optimoidulla kaupallisella mallilla. Testauksessa käytetty Microsoft Copilot perustuu kattavasti vahvistusopetettuun kielimalliin GPT-4 Turbo. Siinä missä FinGPT:n käyttäminen kehotteiden suunnittelulla ei vaikuta nykymuodossaan todennäköiseltä mahdollisuudelta peittämisen toteuttamiseen, yleiskäyttöinen Copilot antoi testissä lupaavia tuloksia suomenkielisessä sanankorvauksessa. Kokeilun tuloksia on käsitelty lähemmin seuraavassa luvussa, mutta konseptitodistuksena (*proof of concept*) testi osoittaa, että peittäminen tekoälyllä on mahdollista.

Mikäli kuitenkin osoittautuisi, että yleiskäyttöinen ja muulle kuin suomen kielelle optimoitu malli ei lopulta toimisi kyllin luotettavasti, ratkaisu voisi sen sijaan löytyä suomalaisesta kielimallista ja pehmeistä kehoitteista tai jopa hienosäädöstä. Voidaan spekuloida, että sanankorvauspeittämisen mallin **hienosäädöllä** on tarpeettoman raskas keino, ja peittäminen tekoälypohjaisesti olisi mahdollista toteuttaa jo kevyimmän vaihtoehdon, **kehoitesäädön**, avulla. Peittäminen sanankorvauksella on mekaanista toimintaa, ja jos kielimalli kykenee tuottamaan kieliopiltaan oikeaa suomea, on mahdollista, että peittäminen onnistuu melko kevyillä säädöillä ja siirto-oppimisella.

Samalla on kuitenkin luultavaa, että kielimallin siirto-oppiminen ei todellisuudessa ole näin yksinkertaista. Selkoteksti on todellista, järkevää tekstiä, jonka mukaan myös kielimallin todennäköisyydet ja kyky ennustaa seuraavaa tokenia on laskettu. Haluttu peiteteksti kuitenkin on sellaista, jonka sanat eivät loogisesti kuulu yhteen. Siinä missä kielimalli ennustaa sanojen ja tokeneiden järjestystä todennäköisyyksien pohjalta, peiteteksti tietoisesti rikkoo yhdessä esiintymisen todennäköisyyksiä. Mahdollinen kompromissi tässä ristiriidassa voi olla peitesanastojen laatiminen teeman mukaan ja yhteenkuuluvien peitesanojen hyödyntäminen. Jos peitesanoina käytetään esimerkiksi yksinomaan vaikkapa vesilintuihin liittyviä termejä, kielimallin toiminta peittämiskäytössä saattaa parantua.

Tilanne muuttuu myös, mikäli sanankorvausta ajatellaan vasta lähtökohtana ja peittämistä suunnitellaan toteutettavaksi muillakin keinoin, kuten korvaamalla lauseita täysin erimuotoisilla, selkotekstistä kieliopillisesti poikkeavilla lauseilla. Myös peittäminen, jossa hyödynnetään salauksellisia elementtejä, kuten kiertäviä salausavaimia tai muita kieliä, voi hyötyä mutkikkaampien säätötekniikoiden käytöstä mallin opettamisessa. Tällainen peittäminen voi jopa edellyttää koko mallin hienosäätöä erilaisiin tehtäviin, jos suhteellisen kevyitä tehtäväkohtaisia etuliitteitä hyödyntävä **etuliitesäätökään** ei tyydyttävästi mallinna tilannetta.

Valittavan tekniikan laji ei ole yhdentekevä: se vaikuttaa olennaisesti tarvittavan laskenta- ja datanvarastointikapasiteetin valitsemiseen, samoin mallin täyden kyvykkyyden hyödyntämiseen, sillä yksittäisissä tehtävissä jokin säätölaji voi yllättäen suoriutua sinänsä kyvykkäämpää tekniikkaa paremmin. Olennainen kysymys kaikkien säätötekniikoiden kanssa on opetus- ja testidatan määrä ja laatu: aineiston riittävyys ja uuden, edustavan aineiston tuottaminen ovat huolenaiheita etenkin silloin, kun toimitaan kapealla kielialueella ja vielä kapeammalla erikoistumisalueella. Mallin opettamiseen ja testaamiseen tarvittavan aineiston tuottamista on käsitelty luvussa 5.4.

Hienosäädön ansiokasta historiaa ei ole syytä sivuuttaa, vaikka muut säätötekniikat ovat tuottaneet lupaavia tuloksia ja haastaneet sen valta-asemaa. On mahdollista, että kevyempien pehmeiden kehoitteiden suorituskyvyn raja tulee vastaan nopeastikin, kun peittämistä kehitetään syvällisemmin, ja lopulta hienosäädöstä tulee ainoa luotettava vaihtoehto toteuttamaan erilaisia loppupään tehtäviä. Tällaisten edistyneempien peittämismenetelmien kehittäminen jää lopulta tämän käsittelyn ulkopuolelle, mutta mahdollisia jatkotutkimusaiheita pohditaan vielä luvussa 6.

5.6 Peittäminen Microsoft Copilotin ja kehoitesuunnittelun avulla

Edellä on spekuloitu, että kehoitesuunnittelu ei todennäköisesti olisi toimiva vaihtoehto FinGPT:n kaltaisen kielimallin kanssa, vaikka se tai mahdollisesti Poro mieluisin vaihtoehto juuri suomen kielelle suunniteltuna, laadittuna ja tokenoituna olisikin. Sen sijaan kiinnostava kysymys on, voisiko monikielisyudessa osin tahattomastikin (mm. Eliot 2023; Petrov et al. 2023, s. 1) ansioitunutta ChatGPT:tä hyödyntää peittämiseen.

Testaus toteutetaan Microsoft Copilot -ohjelmalla, joka toimii ChatGPT:n tavoin chattibottina ja perustuu kielimalliversioon GPT-4 Turbo. Toisin kuin ChatGPT:n ilmaisversio osoitteessa <https://chatgpt.com>, joka käyttää mallia GPT-4 mini, maksullinen ohjelma vastaili testauksen ohella auliisti myös kysymyksiin omista teknisistä tiedoistaan. Marraskuussa 2024 testatun version opetus päättyy lokakuuhun 2023 (ilmaisversion syyskuuhun 2021), minkä jälkeisiin tapahtumiin se voi hakea verkosta vastauksia hakukoneen tavoin.

Ohjelmien vertailu on sinällään kiinnostavaa ja etenkin sävyerot niiden antamissa vastauksissa erityisen havaittavia inhimillisen innokkaan Copilotin ja kliinisen GPT-4 minin välillä, samoin toisinaan huomattavat erot vastausten tarkkuudessa, kun teknisiä tietoja kysyttiin molemmilta englanniksi. Välillä ilmaisversio tuli jopa antaneeksi hyödyllisempää tietoa, mutta useimmiten sen vastaukset olivat suppeampia. Tämä pohdinta teoriassa samaa periaatetta noudattavien chattibottien toiminnan eroista on kuitenkin tässä yhteydessä vain kiinnostava sivujuonne, ja suomenkielisen peittämisen kokeilemiseen valitaan Copilot.

Tämän kielimallin ulostuloja ja syötteen ymmärrystä on kattavasti ohjattu ihmisen toteuttamalla vahvistusoppimisella, mikä voi olla eduksi kokeilussa, vaikka nimenomaisesti peittämistä ja suomenkielistä sanankorvausta sen opetusaineisto ei olisikaan sisältänyt. ChatGPT:hen kohdistunut mittava vahvistusopettaminen myös antaa vapautta kehoitteiden laatimiseen, sillä ohjelma ymmärtää usein puutteellisestikin laadittu pyyntöjä ainakin

englanniksi. Näin ollen kehoitesuunnittelulle luvussa 5.4.1 mainittuja hyviä käytänteitä ja OpenAI:n ehdottamia merkintätapoja ei luultavasti ole tarpeen noudattaa tarkasti.

Testauksessa käytetään luvussa 3.1 esiteltyjä taulukoita 4 ja 5. Tässä kuvattu testi on dokumentoitu tutkielman liitteessä.

Ennen testausta arvio oli, että vaikka englanniksi opetettu chattibotti luultavasti taipuisi suomeksi vähintäänkin helppoon sanankorvaukseen (perusmuotoisten sanojen korvaaminen toisilla), botin ohjeistaminen saattaisi edellyttää kehoitteiden antamista englanniksi. Huoli osoittautui aiheettomaksi: Ohjelma suoriutui yleiskielellä suomeksi annetuista ohjeista ja korjauksista sangen hyvin.

Testissä käytiin keskustelua yhteensä 20 Copilotin antaman vastauksen verran. Ensin botille annettiin taulukko 4, jossa yhtä selkosanaa vastaa yhtä peitesana, ja pyydettiin sitä korvaamaan annetussa virkkeessä esiintyvät sanat annetuilla korvaajillaan. Botti onnistui tässä ensiyrittämältä taivuttaen peitesanat erikseen määrittelemättä oikein, ja testausta jatkettiin eri virkkeillä. Kun testattava virke kirjoitettiin pienillä kirjaimilla ilman lopetuspistettä, botti antoi vastauksessaan silti isolla alkukirjaimella alkavan, pisteeseen päätetyn vastauksen.

Ongelmia ilmeni, kun botille annettiin taulukko 5, jossa yhtä selkosanaa vastaa useita peitesanoja, ja pyydettiin tuottamaan samasta lauseesta useita versioita. Tässä vaiheessa botti yhtäkkiä unohti taivuttaa sanoja, mutta korjasi tulosteensa palautteesta, eikä tehnyt samaa virhettä myöhemmissä vaiheissa.

Tässä kyseessä ei ole luvussa 5.5 mainittu kielimallin katastrofaalinen unohtaminen, vaan mahdollinen syy voisi olla kielimallin konteksti-ikkunan (*context window*) rajallinen koko, joka Copilotissa on sen oman ilmoituksen mukaan 8 192 tokenia. Konteksti-ikkunalla tarkoitetaan sitä määrää tokeneita, jonka malli kerrallaan muistaa keskustelun aikana (Viebrock 2024). Lukuun sisältyvät Copilotissa myös välimerkit ja välilyönnit. Kuitenkin testin pituus oli tässä vaiheessa melko lyhyt, vaikka GPT tokenoisikin suomen kieltä huomattavan pieniin osasiin Englantiin verrattuna (luku 5.2.1). Lisäksi annettu sanankorvaustaulukko säilyi botin muistissa, joten selitys on kyseenalainen. Samoin sen jälkeen, kun bottia pyydettiin säilyttämään alkuperäisen tekstin kirjoitusasu, tulosteeseen ei myöhemmissä vaiheissa tullut muutoksia.

Seuraavaksi botille viritettiin ansa, joka sääntöpohjaisessa ohjelmatoteutuksessa oli huomioitu yksinkertaisella rajauksella (luku 3.4.1). Selkoteksti sisälsikin nyt sellaisia koodisanoja, joita normaalisti käytettäisiin ainoastaan peitesanoina. Rajaus ei sinänsä ole kohtuuton: Taktista viestintää ei mainittavasti rajoittane, jos viestinnän ulkopuolelle rajataan vaikkapa keskustelu kivilajeista tai lintujen liikkeistä silloin, kun tällaiset sanat on

varattu yksinomaan peitteistöön. Botti kuitenkin päätteli, että lauseessa esiintyvät, peite-sanastossa jossain kohtaa esiintyvät sanat täytyy korvata.

Tässä kohtaa hyödylliseksi strategiaksi osoittautui luvussa 5.4.1 esitetty ajatuksenkulku (*chain of thought*), jossa tulostetta korjataan johdattelemalla mallia vaiheittain. Samoin käyttökelpoiseksi osoittautui yhden näytteen kehote (*one-shot prompting*), jossa yhdestä peitevirkkeestä annetaan mallille korjattu versio ja pyydetään sitä toteuttamaan vastaavat korjaukset loppuihin vastauksiin.

Viimeisenä testinä kokeiltiin, kuinka malli suoriutuu viestin purkamisesta. Mallin käyttämää sanankorvaustaulukkoa muutettiin liittämällä siihen kuvitteellinen otsikkorivi ja nimeämällä sarakkeet nimityksillä alkuperäinen sana, salaus1, salaus2 ja salaus3. Lisäksi annettiin yhden näytteen kehote siitä, kuinka salaus2:llä toteutettu virke palautettaisiin alkuperäiseen muotoonsa ja mikä näin saadun vastauksen tulisi olla. Sen jälkeen bottia pyydettiin palauttamaan alkuperäiseen muotoonsa salaus3:lla toteutettu virke, minkä botti osasi heti.

Purkamista kokeiltiin uudestaan eri virkkeellä, nyt palaten takaisin metodiin salaus2, ja tässä vaiheessa botti joutui testin vaikeimman ongelman eteen: kahdesta lauseesta koostuneen virkkeen purkaminen ei enää onnistunutkaan. Useista korjausyrityksistä huolimatta vaikutti siltä, kuin botti olisi soveltanut eri salausta virkkeen kumpaankin lauseeseen. Pyyntö käyttää samaa salaus2:ta koko virkkeeseen johti väärin korjauksiin. Edelleen tehtävänannon, käytettävän virkkeen ja käytettävän salausmetodin toistaminen kehotteessa johti niin ikään väärään tulokseen.

Lopulta purkaminen onnistui siten, että botille lueteltiin tulosteessa olleet virheet ja sanat, joita on käytettävä korvaajina. Lopputulokseen pääseminen vaati alkuperäisen kehotteen lisäksi kolme korjauksia sisältävää kehotetta, vaikka aiempi samaa salausta hyödyntänyt, niin ikään kahden lauseen virke oli mennyt kerralla oikein nyt vaaditun neljän kehotteen sijaan. Tässä kohtaa konteksti-ikkunan rajallisuus olisi aiempaa uskottavampi selitys, sillä viestiketju oli tässä vaiheessa kestänyt merkittävästi pidempään. On mahdollista, että ongelma olikin botin näkökulmasta jokin aivan muu kuin käyttäjän tulkinta siitä, että käytetty salaus oli vaihtunut kesken virkkeen, ja tästä syystä pyyntö käyttää samaa salausta koko virkkeeseen ei korjannutkaan tilannetta. Tässä tapauksessa tarkasti ihmisen intentiota noudattava malli olisi luultavasti ollut pätevämpi laatimaan itselleen omat kehotteensa (*autotune*) kuin ihmiskäyttäjä (myös luvun 5.4.1 pohdinta).

On myös mahdollista, että bottia sekoitti se, että eri salauksissa käytettiin samoja peite-sanoja. Tällä emuloitiin todellista peitteistön käyttöä, jossa sanoja voidaan kierrättää jonkin ennalta sovitun rotaation mukaisesti, mutta tekoäly olisi kenties toiminut paremmin,

jos kukin salaus olisi sisältänyt vain ainutkertaisia peitesanoja. Vaikka botti yhtäkkiä unohtikin käyttämänsä salauksen keskellä virkettä, se osasi kuitenkin väistää sille aiemmin viritetyt ansat: Se osasi ohittaa sanan, jota ei ollut tarpeen purkaa (peitesanojen ja selkosanojen erillisyyttä oli jälleen rikottu tahallisesti), ja antaa vastaukset pyydetyllä kieliasulla.

Testin aikana ohjelma oli kohtelias ja avulias sekä valmis korjaamaan toimintaansa. Ohjelma reagoi samaansa kiitokseen ja kaupanpäällisinä käytti hymiöitäkin osuvasti. Ohjelmaa väkisinkin puhuttelee miellyttävästi vastaan. Ystävällinen käytös chattibottia kohtaan on käytöstapojen ohella perusteltua: ChatGPT toimii paremmin, kun käyttäjä on kohtelias oman kielialueensa normien mukaisesti (Marr 2024; Yin et al. 2024).

Reiluuden vuoksi on myönnettävä, että tietyissä asioissa jo peittämiseen opettamatonkin tekoäly voitti sääntöpohjaisen peittämisohjelman. Sääntöpohjainen toteutus on tällä hetkellä lausepohjainen, eikä mahdollista virkkeiden peittämistä siten, että välimerkit ja isot kirjaimet säilytettäisiin myös peitelauseessa. Tämä ominaisuus on tiedossa, ja luultavasti toteutus kokee muutoksia, mikäli laadittua toiminnallisuutta päädytään työstämään myöhemmin lisää. Tekoäly myös automaattisesti korjasi peitetyn virkkeen kielioppia huolehtimalla virkkeen aloittavan ison kirjaimen ja sen lopettavan pisteen käytöstä, kunnes bottia ohjeistettiin säilyttämään alkuperäinen kieliasu. Tämän ominaisuuden ei voi katsoa olevan yksiselitteisesti hyvä tai huono. Toisaalta aiotussa käyttötarkoituksessa ei ole suotavaa, että ohjelma tekee käyttäjän puolesta päätöksiä. Toisaalta käyttäjäsyötteen oikoluku on tunnistettu jatkokehitystä tai ainakin harkintaa vaativaksi tarpeeksi.

Iloinen yllätys oli myös botin mainio selviytyminen yleiskielellä annetuista, sanamuodoltaan vaihtelevistakin kehoitteista. Sanoja ”peittäminen” tai ”purkaminen” ei testissä käytetty tai botille opetettu, vaan haluttu toiminta kuvattiin sanankorvauksena (itse asiassa ”sanakorvauksena”, minkä botti ymmärsi) ja tekstin palauttamisena alkuperäiseen muotoonsa. Näiden termien opettamisella ei olisi ollut lisäarvoa, sillä Copilot käyttää jäädytettyä kielimallia, eikä olisi voinut hyödyntää tietoa myöhemmissä keskusteluissa.

Tärkeimmät havainnot tekoälyn potentiaalin ja odottamattoman hyvän suomenkielisen suorituskyvyn ohella ovat myös tässä toteutuneet, sinänsä odotetut huolenaiheet. Tekoälyn toteuttamien vastausten varaan on hyvin helppo tuudittautua. Usean oikean vastauksen jälkeen saatu väärä vastaus melko samanlaiseen tehtävään jää helposti ihmistarkastajalta huomaamatta ja voi tulla yllätyksenä. Ongelman aiheuttajan löytäminen on tällaisessa tapauksessa vaikeaa, ellei suorastaan mahdotonta. Virheiden havaitseminen on hankalaa, sillä tekoälyn antamat vastaukset ovat tavallaan syötettä parempia: Tekoäly säilytti moitteetta välimerkit, käsitteli usean lauseen virkkeitä ja korjasi jopa

oikeinkirjoitusta. Luotettavuudessa sääntöpohjainen ohjelmatoteutus kuitenkin voitti tekoälyn tässä pienimuotoisessa testissä. Myöskään Copilotin kykyä taivuttaa suomea ei testattu kattavasti kaikissa sijamuodoissa, joten etenkin jo nyt ongelmallinen purkaminen nominatiivia, genetiiviä ja partitiivia harvinaisemmilla sijapäätteillä voi tuoda yllätyksiä.

Kun peittämistä toteutetaan massatoteutuksena, on välttämätöntä suunnitella tapoja kontrolloida suuria määriä opetus- ja testiaineistoa. Tässä apua antaa laadittu sääntöpohjainen peittämistoiminnallisuus, mutta senkään erehtymättömyyteen ei ole syytä luottaa varauksettomasti. Sääntöpohjainen toteutus ei sisällä kaikkia suomen sanoja, vaikka sen kattavuus onkin erittäin laaja. Jotta pitäydytään mielekkäässä toteutuksessa, kaikkia sanaluokkia kattavaa ohjelmaa ei todennäköisesti ole järkevää toteuttaa myöhemminkään, vaan opetus- ja testaustarkoituksiin on löydettävä keinoja myös muista strategioista, viime kädessä ihmisen tarkastuksesta.

Kuvattu testi toteutettiin yhdessä viestiketjussa, jolloin tekoälyllä oli mahdollisuus oppia palautteesta ja parantaa aiemmin saamiensa tietojen avulla myöhempiä vastauksiaan. Palautteen antaminen, vastausten arvioiminen ja tekoälyä johdatteleva, vaiheittain toteutettu ajatuksenkulku eivät kuitenkaan ole loppukäyttäjän tehtävä taktisessa viestinnässä, vaan todennäköisesti sovellus käynnistetään aina uudelleen lyhyitä vastauksia varten.

Tällöin avainasemaan sovelluksen käyttämän mallin opettamisen ohella nousee myös aloituskehotteen sisältö. Kehotteen ei tarvitse eikä pidäkään näkyä käyttäjälle sellaisenaan, mutta sen tulisi sisältää tehtävänannon ohella ainakin käytetty sanankorvaustaulukko ja esimerkkejä halutuista tulosteista. Edelleen olennainen kysymys on testissä esiin tulleiden ongelmien välttäminen, etenkin äkillisesti heikentynyt suorituskky ja mallin näennäinen salausmetodin vaihtaminen kesken virkkeen. Järjestelmällisellä kehotesuunnittelulla ja testauksella olisi toivottavasti mahdollista löytää sellainen aloituskehote, jolla havaitut ongelmat voitaisiin välttää luomatta uusia tilalle. Tekoälyn ennustamattomuus on kuitenkin todellinen huolenaihe. Kuten Copilot pyydettäessä kertoo, sen antamat vastaukset eivät ole aina samanlaisia, vaan voivat muuttua viestiketjun aiempien vastausten perusteella. Viestiketjun mukaan vaihdellen – vaikkapa useita viestejä peräkkäin peitettäessä tai purettaessa – ja kattavasti testattukin ohjelma voi ja luultavasti antaa sellaisia odottamattomia tuloksia, joihin kaikkiin paraskaan aloituskehote ei voi varautua.

Voi myös pohtia, olisiko peittäminen ja purkaminen on erotettava toisistaan jollakin tavalla, jolloin sovellus vaivihkaa vaihtaisi sisäistä aloituskehotettaan ja aloittaisi uuden keskustelun uudella konteksti-ikkunalla. Aiheellinen kysymys niin ikään on, kuinka

tällainen lopulta toteutettaisiin käyttömukavuuden kärsimättä. Konteksti-ikkunan koko ei luultavasti ollut syynä testissä havaittuihin epäjohdonmukaisuuksiin, mutta täysin varma tästä ei voi olla. Konteksti-ikkunan kokoa voidaan kasvattaa testissä käytetystä, mikäli sen pienuus osoittautuisi ongelmaksi, ja sen ylittyminen voidaan estää jonkinlaisella maksimimerkkimäärällä ja tarvittaessa pakotetulla uudelleenkäynnistyksellä.

FinGPT:n tai mahdollisesti Poron muokkaaminen toteuttamaan peittämistä ja myöhempi sovellusmuotoinen toteutus ovat edelleen lopullisesti ratkaisemattomia aiheita. Tässä ongelmana on myös kielimallin koon ja mobiililaitteiden suorituskyvyn ristiriita. Kuten luvussa 4.4.3 mainittiin, transformereiden mittava koko yleensä estää niiden hyödyntämisen mobiililaitteilla. Copilot ja ChatGPT toimivat toistaiseksi yksinomaan pilvipalveluina, mutta tällainen rajoite ei ole mahdollinen taktisessa viestinnässä, eikä kaupallinen, ulkopuolisen toimijan hallinnoima kokonaisuus muutenkaan ole hyväksyttävä valinta tähän käyttöön.

Voidaan arvioida, että Copilot suoriutui pienimuotoisesta peittämisestä ja purkamisesta varsin hyvin. Ennen testausta oli pohdintaa, että kehoitteet voisi joutua antamaan englanniksi, mutta huoli osoittautui turhaksi: Copilot osasi hyödyntää yleiskielellä annettuja kehoitteita ja korjata toimintaansa niiden pohjalta. Sen suoritus ei ollut virheetön, mutta se oli lupaava.

6. JOHTOPÄÄTÖKSET

Tässä tutkielmassa on tarkasteltu steganografiaa, peittämistä sanankorvauksella, syväoppimista, tekoälyä ja neuroverkkoja päätyen lopuksi suuriin kielimalleihin ja niiden mahdollisuuksiin. Lisäksi on toteutettu syvällistä kielen tuntemusta ja teknistä osaamista vaatinut sääntöpohjainen ohjelma, jolla luonnollista suomen kieltä voi analyysin pohjalta taivuttaa monipuolisesti. Toteutetulla toiminnallisuudella voi jo nyt peittää ja purkaa suomenkielistä viestintää peitesanastoja käyttäen, ja sitä voi myös hyödyntää opetusaineiston laatimisessa tekoälylle. Samoin on esitetty vaihtoehtoja toteuttaa vastaava toiminnallisuus tekoälypohjaisesti opettamalla suurta kielimallia omalla aineistolla, perusteltu tällaisen tutkimuksen toteutettavuus ja vertailtu tarjolla olevia vaihtoehtoja, vaikka diplomityön laajuus ei sallikaan eri vaihtoehtojen täysimittaista toteutusta ja vertailua.

Tekoälypohjaista peittämistä testattiin käyttäen OpenAI:n GPT-kielimalleihin perustuvaa Microsoft Copilot -ohjelmaa. Chattibottina toimivalle ohjelmistolle annettiin saman keskustelun aikana vaikeustasoltaan kasvavia tehtäviä, joista osan se selvitti moitteetta, osan korjasi palautteesta ja osan korjasi usean palautekerran jälkeen. Vaikuttavaa oli botin selviytyminen yleiskielisistä ja vaihtelevistakin suomenkielisistä kehoitteista. Tekoälyn suoritus ei ollut täydellinen, mutta siinä oli lupausta. Osaltaan tällainen yleiskäyttöinen, monikielinen ohjelma kuitenkin vahvasti tutkielmassa esitetyjä ajatuksia tekoälyn mahdollisuuksista steganografiseen viestintään.

Tutkielman ote on ollut läpi tutkimuksen kokonaisuuksiin ja suuruusluokkiin keskittyvä. Käsittelyn ulkopuolelle on jätetty sinänsä tärkeitä aiheita, kuten luonnollisen kielen käsittelyyn liittyvää metriikkaa ja arviointikriteerejä sekä kaavojen matemaattista käsittelyä. Näitä aiheita olisi helposti käsitellyt oman päälukunsa verran kumpaakin, mutta tällainen käsittely ei olisi palvellut kokonaiskuvan muodostamista tarjolla olevista tekoälyn mahdollisuuksista. Pelin luonteeseen valitettavasti kuuluu, että tekoälyä koskeva tieto on äärimmäisen nopeasti päivittyvää ja vanhenevaa. Historia ei luultavasti ole kovin armelias nyt muodostetulle kokonaiskuvalle edes puolenkaan vuoden jälkeen, saati useamman vuoden.

Myös steganografian luonteeseen liittyy monia kysymyksiä, joita olisi kiinnostavaa tarkastella lähemmin. Etenkin psykologinen näkökulma olisi mielenkiintoinen: tutkimus siitä, mikä tekee piiloviestistä vaikeasti havaittavan ja kuinka viestitään piilossa näkyvillä muutenkin kuin näkö- ja kuuloaistin heikkouksia hyödyntämällä. Toisaalta minkä tahansa

muun kuin konetarkkailijan läsnäolon olettaminen saattaa olla jo vanhakantaista, mutta kysymys on joka tapauksessa kutkuttava.

Kaupallisella kielimallilla tehty testi toi lisää huomioita vielä toteutusta odottavaan suurten kielimallien opettamiseen omalla aineistolla. Seuraava vaihe jatkotutkimukselle olisi toteuttaa prototyyppi suomalaisen kielimallin opettamisesta valituilla kielimallin säätötekniikoilla ja tarkastella, millaisia tuloksia niillä saadaan käytännössä.

Nimenomaisesti tämän tutkielman käsittelystä jääviä jatkotutkimus- ja jatkokehitysaiheita on useita. Näihin kuuluvat lopullisen sovelluksen viimeistely nyt toteutettuja toiminnallisuuksia hyödyntäen sekä merkityksellinen peitesanastojen laatiminen ja tämän laadinnan automatisointi. Näihin kuuluvat säätötekniikan tai -tekniikoiden käytännön testaaminen suomenkieliseen peittämiseen ja tehtyjen oletusten validointi tai uudelleenarviointi. Koska kielimallien säätötekniikat kehittyvät nopeasti, mahdollisen jatkotutkimuksen tullessa ajankohtaiseksi nyt esiteltyjen tekniikoiden tarkastelu ja päivittäminen osittain tai kokonaan lienee ensimmäinen askel kielimallien hallintaan.

Harkinnan arvoinen on myös pohdinta mahdollisen salauksen lisäämisestä steganografiseen viestiin ja erilaisten salausalgoritmien tai stegoavainten käyttö. Tällainen toiminnallisuus veisi toteutuksen lähemmäs kryptografian aihealuetta, ja huomioiden tieteenalojen jokseenkin läheisen suhteen, hyppy ei kenties olisi kovin kaukaa haettu.

Erityisen kiintoisa kehitysajatus, jota tutkielmassa ei ole käsitelty lainkaan, on peittämisen toteuttaminen täysin toisessa muodossa olevalla viestillä. Sen sijaan, että sanankorvauksella ilmaistaan viesti ”tuntematon ajoneuvo matkalla kenttäsairaalaan” piiloviestillä ”taivaansininen hippiainen matkalla kanton”, korvattaisiinkin viestin sisältö täysin toisenlaisella rakenteella, kuten ”laineilla vene liplattaa”. Oletus on, että neuroverkkojen ja generatiivisen tekoälyn avulla tällainen toteutus voisi olla mahdollinen, mutta kirjallisuudesta ei löytynyt viitteitä siitä, että vastaavaa tutkimusta olisi tehty. Tutkimus ja neuroverkkojen kykyjen koettelu kehittyi kuitenkin niin ennustamattomasti, että tilanne voi muuttua nopeasti. Vielä pidemmälle vietynä tätä ajatusta voisi kehittää siten, että peittämistä voisi toteuttaa mille tahansa tekstille sanankorvaustaulukoiden ulkopuolella joidenkin kielimalliin liitettyjen, ulkopuolisille tuntemattomien ehtojen aktivoituessa.

Työn ohessa on tehty sivutoimista sanastotyötä ja kielenhuoltoa tallentamalla ja valikoidulla vain hajanaisesti saatavilla olevaa suomenkielistä ammattisanastoa, osin luomalla sellaista itse. Olisi tärkeää, että tekoälyä, syväoppimista ja steganografiaa käsittelevää terminologiaa koottaisiin ja suomen sanastotyötä tehtäisiin myös normatiivisesti laajemmalla asiantuntijakollegiolla ja -katselmoinnilla. Tieteellisiä julkaisuja hallitsee englantia, mutta tämän ei pitäisi olla este suomenkielisen tieteellisen sanaston kehittämiseksi ja

aktiiviselle käyttämiselle. Hyvä tieteellinen kieli vaatii aktiivista panostusta ja sanaston ylläpitämistä.

Steganografian merkittävyys todennäköisesti nousee jatkossa. Kätkemisen kulta-aikana saatetaan pitää toisen maailmansodan aikaista viestintää, mutta 2000-luvulla nousee entistä hienovaraisempia, innovatiivisempia ja vaikeasti havaittavampia steganografisia keinoja, jotka luultavasti yhdistelevät kryptografiaa ja steganografiaa uudella tavalla. Jokainen steganografian edistysaskel on samalla väistämätön kimmoke myös entistä paremman ja kattavamman steganalyysin kehittämiseen. Tutkimusaihe ei ole vielä valmis.

LÄHTEET

Aamuset (2024). Uusi kielimalli Viking julkaistu – ensimmäinen avoin malli pohjoismaisille kielille. Julkaistu 19.5.2024. Saatavissa [haettu 5.8.2024]: <https://aamuset.fi/artikkeli/6328823>.

Agarap, A. F. M. (2019). Deep Learning Using Rectified Linear Units (ReLU). 7 sivua. Saatavissa [haettu 11.11.2024]: <https://arxiv.org/abs/1803.08375>.

Agarwal, M. (2013). Text Steganographic Approaches: A Comparison. International Journal of Network Security & Its Applications (IJNSA), vol. 5, no. 1, January 2013. 16 sivua. Saatavissa [haettu 22.9.2024]: <https://arxiv.org/abs/1302.2718>.

Ahuja, V. (2023). Few-Shot Learning: Comparison with Traditional Machine Learning Algorithm and Deep Learning. Saatavissa [haettu 30.7.2024]: <https://medium.com/@vipula.a.e/few-shot-learning-comparison-with-traditional-machine-learning-and-deep-learning-6c6495a66b63>.

Ahvanooey, M. T., Li, Q., Hou, J., Rajput, A. R. & Chen, Y. (2019). Modern Text Hiding, Text Steganalysis, and Applications: A Comparative Analysis. Entropy (Basel), April 1, 21(4):355. 31 sivua. Julkaistu 1.4.2019. Saatavissa [haettu 4.10.2024]: <https://pubmed.ncbi.nlm.nih.gov/33267069/>.

AlphaStar (2019). AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. Julkaistu 24.1.2019. Saatavissa [haettu 6.7.2024]: <https://deepmind.google/discover/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii/>.

AlSabhany, A. A., Ali, A. H., Ridzuan, F., Azni, A. H. & Mokhtar, M. R. (2020). Digital Audio Steganography: Systematic Review, Classification, and Analysis of the Current State of the Art. Computer Science Review volume 38, November 2020, 100316. 27 sivua. Saatavissa [haettu 18.9.2024]: <https://www.sciencedirect.com/science/article/pii/S1574013720304160>.

Anderson, R. J. & Petitcolas, F. A. P. (1998). On the Limits of Steganography. IEEE Journal on Selected Areas in Communications, vol. 16, no. 4, May 1998. 8 sivua. Saatavissa [haettu 26.9.2024]: <https://www.cl.cam.ac.uk/~rja14/Papers/jsac98-limsteg.pdf>.

ASCII (2024). ASCII Table: Reference to ASCII Table of Windows-1252. Saatavissa [haettu 14.10.2024]: <https://www.ascii-code.com/>.

Bennett, K. (2004). Linguistic Steganography: Survey, Analysis, and Robustness Concerns for Hiding Information in Text. CERIAS Tech Report 2004-13, Purdue University, Indiana, United States. 30 sivua. Julkaistu 12.5.2004. Saatavissa [haettu 4.10.2024]: https://www.cerias.purdue.edu/apps/reports_and_papers/view/2697.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. & Amodei, D. (2020). Language Models Are Few-Shot Learners. 75 sivua. Julkaistu 28.5.2020, päivitetty 22.7.2020. Saatavissa [haettu 9.3.2024]: <https://arxiv.org/abs/2005.14165>.

Broz, M. (2024). How Many Pictures Are There in 2024? Photutorial Statistics. Julkaistu 12.9.2024. Saatavissa [haettu 3.11.2024]: <https://photutorial.com/photos-statistics/>.

Camenzind, M., Göbel, N., Eberhard-Moscicka, A. K., Knobel, S. E. J., Hegi, H., Single, M., Kaufmann, B. C., Schumacher, R., Nyffeler, T., Nef, T. & Müri, R. M. (2024). The Phenomenology of Pareidolia in Healthy Subjects and Patients with Left- or Right-Hemispheric Stroke. *Heliyon*, 2024-03, vol.10 (5), s. e27414-e27414, article e27414. 8 sivua. Saatavissa [haettu 1.10.2024]: [https://www.cell.com/heliyon/fulltext/S2405-8440\(24\)03445-5](https://www.cell.com/heliyon/fulltext/S2405-8440(24)03445-5).

Chapman, M. & Davida, G. (1997). Hiding the Hidden: A Software System for Concealing Ciphertext as Innocuous Text. *Information and Communications Security. First International Conference, ICIS'97, Beijing, China, November 11–14, 1997. Lecture Notes in Computer Science, 1997*, s. 335–345. 11 sivua. Saatavissa [haettu 17.10.2024]: <https://link.springer.com/chapter/10.1007/BFb0028489>.

Chapman, M., Davida, G. & Rennhard, M. (2001). A Practical and Effective Approach to Large-Scale Automated Linguistic Steganography. *Information Security, 2001*, vol. 2200, s.156–165.12 sivua. Saatavissa [haettu 17.10.2024]: https://www.researchgate.net/publication/220905193_A_Practical_and_Effective_Approach_to_Large-Scale_Automated_Linguistic_Steganography.

Cheddad, A., Condell, J., Curran, K. & Mc Kevitt, P. (2010). Digital Image Steganography: Survey and Analysis of Current Methods. *Signal Processing 90* (2010), s. 727–752. 26 sivua. Saatavissa [haettu 31.10.2024]: <https://www.sciencedirect.com/science/article/pii/S0165168409003648>.

Chen, Z., Huang, L., Yu, Z., Yang, W., Li, L., Zheng, X. & Zhao, X. (2008). Linguistic Steganography Detection Using Statistical Characteristics of Correlations between Words. *Information Hiding*, s. 224–235. 12 sivua. Saatavissa [haettu 16.10.2024]: https://link.springer.com/chapter/10.1007/978-3-540-88961-8_16.

Chhabra, P. & Goyal, S. (2023). A Thorough Review on Deep Learning Neural Network. *2023 International Conference on Artificial Intelligence and Smart Communication (AISC)*. 7 sivua. Julkaistu 3.4.2023. Saatavissa [haettu 30.4.2024]: <https://ieeexplore.ieee.org/document/10085166>.

Chiusano, F. (2022). Two Minutes NLP — Perplexity Explained with Simple Probabilities. Saatavissa [haettu 29.10.2024]: <https://medium.com/nlplanet/two-minutes-nlp-perplexity-explained-with-simple-probabilities-6cdc46884584>.

Christian, G. (2022a). Reason Behind Instagram Compression Being Added to the Images You Upload. Julkaistu 3.8.2022. Saatavissa [haettu 28.10.2024]: <https://www.aiseesoft.com/resource/instagram-compression.html>.

Christian, G. (2022b). Understand Why Facebook Compression Is Automatic to the Uploaded Images. Julkaistu 12.8.2022. Saatavissa [haettu 28.10.2024]: <https://www.aiseesoft.com/resource/facebook-image-compression.html>.

Cunningham, P. & Delany, S. J. (2021). Underestimation Bias and Underfitting in Machine Learning. *Trustworthy AI - Integrating Learning, Optimization and Reasoning, 2021*, vol.12641, s. 20–31. 12 sivua. Saatavissa [haettu 2.7.2024]: <https://arxiv.org/abs/2005.09052>.

- DAIR Institute (2024). Prompt Engineering Guide. Päivitetty 26.6.2024. Saatavissa [haettu 15.8.2024]: <https://www.promptingguide.ai/techniques/fewshot>.
- Domsch, S. (2013). Signal to Noise – Engaging with the Information Society from Pope to 'Year Zero'. *Anglistik*, Volume 24, Issue 1 (2013), s. 165–176. 12 sivua. Saatavissa [haettu 30.9.2024]: <https://angl.winter-verlag.de/article/angl/2013/1/15>.
- Dugas, D. (2024). The GPT-3 Architecture, on a Napkin. Saatavissa [haettu 26.11.2024]: https://dugas.ch/artificial_curiosity/GPT_architecture.html.
- Dumitrescu, S. & Wu, X. (2005). A New Framework of LSB Steganalysis of Digital Media. *IEEE Transactions on Signal Processing*, vol. 53, no. 10, October 2005, s. 3936–3947. 12 sivua. Saatavissa [haettu 31.10.2024]: <https://ieeexplore.ieee.org/abstract/document/1511008>.
- Eliot, L. (2023). Solving the Mystery of How ChatGPT and Generative AI Can Surprisingly Pick Up Foreign Languages, Says AI Ethics and AI Law. *Forbes*. Julkaistu 19.4.2023. Saatavissa [haettu 26.11.2024]: <https://www.forbes.com/sites/lan-ceeliot/2023/04/19/solving-the-mystery-of-how-chatgpt-and-generative-ai-can-surprisingly-pick-up-foreign-languages-says-ai-ethics-and-ai-law/>.
- Euroopan keskuspankki (2024). Eurosetelin turvatekijät. Saatavissa [haettu 16.9.2024]: <https://www.ecb.europa.eu/euro/banknotes/current/security/html/index.fi.html#additionalFeatures>.
- Fu, Z., Lam, W., Yu, Q., Anthony, M.-C. S., Hu, S., Liu, Z. & Collier, N. (2023). Decoder-Only or Encoder-Decoder? Interpreting Language Model as a Regularized Encoder-Decoder. 22 sivua. Julkaistu 8.4.2023. Saatavissa [haettu 28.11.2024]: <https://arxiv.org/abs/2304.04052>.
- Genkina, D. (2024). AI Prompt Engineering Is Dead. Julkaistu 6.3.2024. Saatavissa [haettu 15.8.2024]: <https://spectrum.ieee.org/prompt-engineering-is-dead>.
- Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, Massachusetts, United States. 773 sivua. Saatavissa [haettu 27.3.2024]: <https://www.deeplearningbook.org/>.
- Grammarly (2024). GPT-3 vs. GPT-4: What's the Difference? Päivitetty 18.7.2024. Saatavissa [haettu 15.8.2024]: <https://www.grammarly.com/blog/ai/gpt-3-vs-gpt-4/>.
- Hakulinen, A., Vilkuna, M., Korhonen, R., Koivisto, V., Heinonen, T. R. & Alho, I. (2004). *Iso suomen kielioppi*. 1698 sivua. Suomalaisen kirjallisuuden seuran toimituksia 950, Helsinki. ISBN 951-746-557-2.
- Hallamaa, T. (2023a). EU:n tekoälyasetus on paisunut tuotesääntelystä ihmisoikeuksien suojelijaksi – suurta huomiota saanut asetus etenee tällä viikolla. *Yle*. Julkaistu 13.6.2023. Saatavissa [haettu 9.5.2024]: <https://yle.fi/a/74-20036203>.
- Hallamaa, T. (2023b). FinGPT3 on suurin puhtaasti suomenkielinen kielimalli, eikä suurempaa ole hetkeen tulossa. *Yle* 10.5.2023. Saatavissa [haettu 5.8.2024]: <https://yle.fi/a/74-20030871>.
- Hendrycks, D. & Gimpel, K. (2016). Gaussian Error Linear Units (GELUs). 10 sivua. Julkaistu 27.6.2016, päivitetty 6.6.2023 (v5). Saatavissa [haettu 5.4.2024]: <https://arxiv.org/abs/1606.08415>.

- Heyman, S. (2015). Photos, Photos Everywhere. *The New York Times*. Julkaistu 29.7.2015. Saatavissa [haettu 3.11.2024]: <https://www.nytimes.com/2015/07/23/arts/international/photos-photos-everywhere.html>.
- Hong, Z. (2023). Attention Mechanisms in Deep Learning: Enhancing Model Performance. Julkaistu 13.10.2023. Saatavissa [haettu 7.7.2024]: <https://medium.com/@zhonghong9998/attention-mechanisms-in-deep-learning-enhancing-model-performance-32a91006092a>.
- Hugging Face (2024). Summary of the Tokenizers. Saatavissa [haettu 21.11.2024]: https://huggingface.co/docs/transformers/tokenizer_summary.
- IBM (2024a). What Is Artificial Intelligence (AI)? Saatavissa [haettu 29.3.2024]: <https://www.ibm.com/topics/artificial-intelligence>.
- IBM (2024b). What Is Underfitting? Saatavissa [haettu 3.7.2024]: <https://www.ibm.com/topics/underfitting>.
- Jain, A., Maleki, A. & Saade, N. (2024). Methods for Adapting Large Language Models. Julkaistu 7.8.2024. Saatavissa [haettu 29.8.2024]: <https://ai.meta.com/blog/adapting-large-language-models-llms/>.
- Janakiram, MSV (2024). The Building Blocks of LLMs: Vectors, Tokens and Embeddings. Saatavissa [haettu 6.8.2024]: <https://thenewstack.io/the-building-blocks-of-llms-vectors-tokens-and-embeddings>.
- Jenifer, J. M., Ratna, S. R., Loreto, J. B. S. & Gethsy, D. M. (2018). A Survey on Different Video Steganography Techniques. 2nd International Conference on Trends in Electronics and Informatics (ICOEI), 2018, s. 627–632. 6 sivua. Saatavissa [haettu 17.9.2024]: <https://ieeexplore.ieee.org/document/8553847>.
- Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Chen, D., Chan, H. S., Dai, W., Madotto, A. & Fung, P. (2022). Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, 2023-03, vol. 55 (12), article 248. 39 sivua. Julkaistu 8.2.2022, päivitetty 19.2.2024 (v6). Saatavissa [haettu 9.7.2024]: <https://arxiv.org/abs/2202.03629>.
- Kallio, S. (2024). Mitä on generatiivinen tekoäly – GenAI Opas (2024). Päivitetty 4.6.2024. Saatavissa [haettu 28.7.2024]: <https://santerikallio.com/genai-opas/>.
- Karampidis, K., Kavallieratou, E. & Papadourakis, G. (2018). A Review of Image Steganalysis Techniques for Digital Forensics. *Journal of Information Security and Applications* 40 (2018), s. 217–235. 19 sivua. Saatavissa [haettu 18.10.2024]: <https://www.sciencedirect.com/science/article/pii/S2214212617300777>.
- Kaur, H. & Rani, J. (2016). A Survey on Different Techniques of Steganography. *MATEC Web of Conferences*, 2016. 6 sivua. Saatavissa [haettu 4.10.2024]: <https://doi.org/10.1051/mateconf/20165702003>.
- Ke, Y., Zhang, M.-q., Liu, J., Su, T.-t. & Yang, X.-y. (2019). Generative Steganography with Kerckhoffs' Principle. *Multimedia Tools and Applications*, 2019-05, vol. 78 (10), s. 13805–13818. 14 sivua. Saatavissa [haettu 3.11.2024]: <https://link.springer.com/article/10.1007/s11042-018-6640-y>.

- Kielitoimiston ohjepankki (2023). Kohtelias kehotus tai pyyntö: liitepartikkelit -han, -pa ja -s. Saatavissa [haettu 9.8.2024]: <https://kielitoimistonohjepankki.fi/ohje/kohtelias-kehotus-tai-pyynto-liitepartikkelit-han-pa-ja-s-sammutathan-valot-nukkukaapas-nyt/>.
- Korpihalkola, J. (2024). Syväoppiminen. Kurssimateriaali. Jyväskylän ammattikorkeakoulu JAMK. Saatavissa [haettu 29.3.2024]: <https://syvaoppiminen-ttow1400-3001.pages.labranet.jamk.fi/syvaoppiminen-materiaalit/>.
- Kotus (2024a). Astevaihtelutyypit. Kotimaisten kielten keskus. Saatavissa [haettu 28.2.2024]: <https://kaino.kotus.fi/sanat/nykysuomi/astevaihtelutyypit.php>.
- Kotus (2024b). Taivutustyypit. Kotimaisten kielten keskus. Saatavissa [haettu 28.2.2024]: <https://kaino.kotus.fi/sanat/nykysuomi/taivutustyypit.php>.
- Kudo, T. & Richardson, J. (2018). SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing. 6 sivua. Saatavissa [haettu 26.11.2024]: <https://arxiv.org/abs/1808.06226>.
- Laatikainen, T. (2024). Miksi tätä ei ole ennen tehty? Suomalainen keksintö automatisoi tuhansien työt ja tekee työvuorolistat tekoälyn avulla. Tekniikka & Talous 11/2024. Julkaistu 19.3.2024. Saatavissa [haettu 24.3.2024]: <https://www.tekniikkatalous.fi/uutiset/miksi-tata-ei-ole-ennen-tehty-suomalainen-keksinto-automatisoi-tuhansien-tyot-ja-tekee-tyovuorolistat-tekoalyn-avulla/5d31125d-2cd2-419a-a007-e0b378bb6eed>.
- Latif, S., Zaidi, A., Cuayáhuil, H., Shamshad, F., Shoukat, M. & Qadir, J. (2023). Transformers in Speech Processing: A Survey. Saatavissa [haettu 1.12.2024]: <https://arxiv.org/abs/2303.11607>.
- Lee, M. (2023). GELU Activation Function in Deep Learning: A Comprehensive Mathematical Analysis and Performance. 19 sivua. Saatavissa [haettu 5.4.2024]: <https://arxiv.org/abs/2305.12073>.
- Lehtinen, R. (2013). Perussanat, johdokset ja yhdyssanat. Kielikello 3/2013. Saatavissa [haettu 20.3.2024]: <https://kielikello.fi/perussanat-johdokset-ja-yhdyssanat/>.
- Lempinen, J. (2024). Kielimalleihin liittyy kohtuuttomia odotuksia erityisesti suomen kielellä – näin ne oikeasti toimivat. Yle. Julkaistu 17.1.2024. Saatavissa [haettu 21.3.2024]: <https://yle.fi/aihe/a/20-10006122>.
- Lester, B., Al-Rfou, R. & Constant, N. (2021). The Power of Scale for Parameter-Efficient Prompt Tuning. 15 sivua. Julkaistu 18.4.2021, päivitetty 2.9.2021. Saatavissa [haettu 2.2.2024]: <https://arxiv.org/abs/2104.08691>.
- Li, X. L. & Liang, P. (2021). Prefix-Tuning: Optimizing Continuous Prompts for Generation. 15 sivua. Julkaistu 1.1.2021. Saatavissa [haettu 11.3.2024]: <https://arxiv.org/abs/2101.00190>.
- Li, Y., Miao, N., Ma, L., Shuang, F. & Huang, X. (2023). Transformer for Object Detection: Review and Benchmark. Engineering Applications of Artificial Intelligence, volume 126, part C, November 2023, 107021. 16 sivua. Saatavissa [haettu 25.11.2024]: <https://www.sciencedirect.com/science/article/pii/S0952197623012058>.
- Liikenne- ja viestintäministeriö (2004). Tekstiviestejä lähetettiin vuonna 2003 enemmän kuin koskaan – myös muut mobiilipalvelut kasvattivat vahvasti suosiotaan. Tiedote 22.4.2004. Saatavissa [haettu 3.11.2024]: <https://lvm.fi/-/tekstiviesteja-lahetettiin->

vuonna-2003-enemman-kuin-koskaan-myos-muut-mobiilipalvelut-kasvattivat-vahvasti-suosiotaan-753775.

Liu, X., Ji, K., Fu, Y., Tam, W., Du, Z., Yang, Z. & Tang, J. (2022). P-Tuning: Prompt Tuning Can Be Comparable to Fine-Tuning across Scales and Tasks. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 8 sivua. Saatavissa [haettu 26.3.2024]: <https://aclanthology.org/2022.acl-short.8/>.

Liu, Y., Liu, S., Wang, Y., Zhao, H. & Liu, S. (2019). Video Steganography: A Review. Neurocomputing, Volume 335, March 28, 2019, s. 238–250. 13 sivua. Saatavissa [haettu 18.9.2024]: <https://www.sciencedirect.com/science/article/abs/pii/S0925231218312608>.

Lockwood, R. & Curran, K. (2017). Text Based Steganography. International Journal of Information Privacy, Security and Integrity, 3(2), s. 134–153. Ulster University, United Kingdom. 21 sivua. Saatavissa [haettu 4.10.2024]: <https://www.inderscienceonline.com/doi/abs/10.1504/IJPSI.2017.088700>.

Lubacz, J., Mazurczyk, W. & Szczypiorski, K. (2014). Principles and Overview of Network Steganography. IEEE Communications Magazine, 2014-05, vol. 52 (5), s. 225–229. 5 sivua. Saatavissa [haettu 16.9.2024]: <https://ieeexplore.ieee.org/document/6815916>.

Lui, B. (2024). Prompt Engineering is Dead? Now What? Saatavissa [haettu 15.8.2024]: <https://medium.com/@obbyjust/prompt-engineering-is-dead-now-what-48d4d1f967fa>.

Luukkonen, R., Komulainen, V., Luoma, J., Eskelinen, A., Kanerva, J., Kupari, H.-M., Ginter, F., Laippala, V., Muennighoff, N., Piktus, A., Wang, T., Tazi, N., Le Scao, T., Wolf, T., Suominen, O., Sairanen, S., Merioksa, M., Heinonen, J., Vahtola, A., Antao, S. & Pyysalo, S. (2023). FinGPT: Large Generative Models for a Small Language. 17 sivua. Julkaistu 3.11.2023. Saatavissa [haettu 14.11.2023]: <https://arxiv.org/abs/2311.05640>.

Maas, A. L., Hannun, A. Y. & Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, United States, 2013. 6 sivua. Saatavissa [haettu 1.12.2024]: https://ai.stanford.edu/%7Eamaas/papers/relu_hybrid_icml2013_final.pdf.

Majeed, M. A., Sulaiman, R., Shukur, Z. & Hasan, M. K. (2021). A Review on Text Steganography Techniques. Mathematics (Basel), 2021-11, vol. 9 (21), 2829. 28 sivua. Saatavissa [haettu 2.10.2024]: <https://www.mdpi.com/2227-7390/9/21/2829>.

Marcelline, M. (2023). Cybercriminals Using ChatGPT to Build Hacking Tools, Write Code. Julkaistu 8.1.2023. Saatavissa [haettu 6.8.2024]: <https://uk.pcmag.com/security/144802/cybercriminals-using-chatgpt-to-build-hacking-tools-write-code>.

Marr, B. (2024). Why You Should Be Polite to ChatGPT and Other AIs. Forbes. Julkaistu 5.11.2024. Saatavissa [haettu 28.11.2024]: <https://www.forbes.com/sites/bernardmarr/2024/11/05/why-you-should-be-polite-to-chatgpt-and-other-ais/>.

Martineau, K. (2023). What Is Generative AI? IBM Research. Julkaistu 20.4.2023. Saatavissa [haettu 9.8.2024]: <https://research.ibm.com/blog/what-is-generative-ai>.

McGuinness, P. (2023). GPT-4 Details Revealed. Julkaistu 12.6.2023. Saatavissa [haettu 15.8.2024]: <https://patmcguinness.substack.com/p/gpt-4-details-revealed>.

Meng, L., Jiang, X. & Sun, T. (2024). A Review of Coverless Steganography. *Neuro-computing (Amsterdam)*, vol. 566 (2024), article 126945. 22 sivua. Saatavissa [haettu 2.10.2024]: <https://www.sciencedirect.com/science/article/pii/S0925231223010688>.

Meng, P., Hang, L., Yang, W., Chen, Z. & Zheng, H. (2009). Linguistic Steganography Detection Algorithm Using Statistical Language Model. 2009 International Conference on Information Technology and Computer Science. 4 sivua. Saatavissa [haettu 16.10.2024]: <https://ieeexplore.ieee.org/document/5190297>.

Metzger, S. (2022). A Beginner's Guide to Tokens, Vectors, and Embeddings in NLP. Julkaistu 20.12.2022. Saatavissa [haettu 25.11.2024]: <https://medium.com/@saschametzger/what-are-tokens-vectors-and-embeddings-how-do-you-create-them-e2a3e698e037>.

Microsoft (2024). Turn Off Image Compression. Saatavissa [haettu 28.10.2024]: <https://support.microsoft.com/en-us/office/turn-off-picture-compression-81a6b603-0266-4451-b08e-fc1bf58da658>.

Microsoft Learn (2024). Zero-Shot and Few-Shot Learning. Julkaistu 20.5.2024. Saatavissa [haettu 4.9.2024]: <https://learn.microsoft.com/en-us/dotnet/ai/conceptual/zero-shot-learning>.

Monika, A. & Eswari, R. (2023). An Ensemble Based Stegware Detection System for Information Hiding Malware Attacks. *Journal of Ambient Intelligence and Humanized Computing* (2023) 14, s. 4401–4417. 18 sivua. Saatavissa [haettu 17.9.2024]: <https://link.springer.com/article/10.1007/s12652-023-04559-z>.

Murphy, K. P. (2022). *Probabilistic Machine Learning: An Introduction*. MIT Press, Cambridge, Massachusetts, United States. ISBN: 9780262046824. 858 sivua. Saatavissa [haettu 5.7.2024]: <https://probml.github.io/pml-book/book1.html>.

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N. & Mian, A. (2023). A Comprehensive Overview of Large Language Models. 46 sivua. Julkaistu 12.7.2023, päivitetty 9.4.2024 (v9). Saatavissa [haettu 31.7.2024]: <https://arxiv.org/abs/2307.06435>.

Nexla (2024). Prompt Tuning vs. Fine-Tuning: Differences, Best Practices and Use Cases. Saatavissa [haettu 28.8.2024]: <https://nexla.com/ai-infrastructure/prompt-tuning-vs-fine-tuning/>.

Nguyen, X. S. (2024). Easier to Understand: What Is a Transformer? How Does GPT Work? Julkaistu 11.2.2024. Saatavissa [haettu 28.11.2024]: <https://blog.ngxson.com/easier-to-understand-what-is-transformer>.

Nielsen, D. S., Enevoldsen, K. & Schneider-Kamp, P. (2024). Encoder vs Decoder: Comparative Analysis of Encoder and Decoder Language Models on Multilingual NLU Tasks. 14 sivua. Julkaistu 19.6.2024. Saatavissa [haettu 28.11.2024]: <https://arxiv.org/abs/2406.13469>.

Nykysuomen sanalista. Kotimaisten kielten keskus. Päivitetty 20.4.2023. Saatavissa [haettu 13.3.2024]: <https://kaino.kotus.fi/lataa/nykysuomensanalista2022.csv>.

OpenAI (2023). Our Approach to AI Safety. Julkaistu 5.4.2023. Saatavissa [haettu 6.8.2024]: <https://openai.com/index/our-approach-to-ai-safety/>.

OpenAI (2024). Prompt Engineering Guide. Saatavissa [haettu 15.8.2024]: <https://platform.openai.com/docs/guides/prompt-engineering>.

Othman, E. S. (2020). Hide and Seek: A White Paper in Steganography. *Journal of the ACS Advances in Computer Science*, article 5, volume 11, issue 1, 2020, s. 39–70. 31 sivua. Saatavissa [haettu 14.9.2024]: https://journals.ekb.eg/article_176270.html.

Oza, S. (2019). Stegware aka Steganography Malware – Malware of the Month, July 2019. *Spanning*. Saatavissa [haettu 17.9.2024]: <https://spanning.com/blog/stegware-steganography-malware-malware-of-the-month/>.

Peng, B., Narayanan, S. & Papadimitriou, C. (2024). On Limitations of the Transformer Architecture. Julkaistu 13.2.2024. 14 sivua. Saatavissa [haettu 7.7.2024]: <https://arxiv.org/html/2402.08164v1>.

Petrov, A., La Malfa, E., Torr, P. H. S. & Bibi, A. (2023). Language Model Tokenizers Introduce Unfairness between Languages. 15 sivua. Julkaistu 17.5.2023, päivitetty 20.10.2023 (v2). Saatavissa [haettu 30.7.2024]: <https://arxiv.org/abs/2305.15425>.

Por, L. Y. & Delina, B. (2008). Information Hiding: A New Approach in Text Steganography. *Proceedings of the 7th WSEAS International Conference on Applied Computer & Applied Computational Science (ACACOS '08)*, Hangzhou, China, s. 689–695. 7 sivua. Saatavissa [haettu 2.10.2024]: https://www.academia.edu/1491402/Information_Hiding_A_New_Approach_in_Text_Steganography.

Rahman, A. (2021). Overfitting & Underfitting Solutions. Kaggle. Saatavissa [haettu 4.7.2024]: <https://www.kaggle.com/discussions/general/222637>.

Ramachandran, P., Zoph, B. & Le, Q. V. (2017). Searching for Activation Functions. 13 sivua. Julkaistu 16.10.2017, päivitetty 27.10.2017 (v2). Saatavissa [haettu 11.11.2024]: <https://arxiv.org/abs/1710.05941>.

Restack (2024). Tokenization with GPT-3 Tokenizer. Päivitetty 14.11.2024. Saatavissa [haettu 21.11.2024]: <https://www.restack.io/p/tokenization-knowledge-gpt-3-tokenizer-cat-ai>.

Russell, S. & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach*, 3rd Edition, 1151 sivua. Pearson Education Limited. e-book ISBN 9781292153971. Saatavissa [haettu 29.3.2024]: <https://ebookcentral.proquest.com/lib/tampere/detail.action?docID=5483443>.

Sainath, T. N., Vinyals, O., Senior, A. & Sak, H. (2015). Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, Queensland, Australia, s. 4580–4584. 5 sivua. Saatavissa [haettu 30.3.2024]: <https://ieeexplore.ieee.org/document/7178838>.

Salonen, J. & Okkolin, T. (2008). *Opas pelastustoimen tehtävien hälytysohjeen laatimiseksi*. Pelastusopiston julkaisu, 1/2008. 69 sivua. Saatavissa [haettu 4.11.2024]: http://info.smedu.fi/kirjasto/Sarja_D/D1_2008.pdf.

Schneider, J., Meske, C. & Kuss, P. (2024). *Foundation Models: A New Paradigm for Artificial Intelligence*. *Business & Information Systems Engineering*, 2024-01. 11 sivua.

Julkaistu 29.1.2024. Saatavissa [haettu 9.4.2024]: <https://link.springer.com/article/10.1007/s12599-024-00851-0>.

Schuster, M. & Nakajima, K. (2012). Japanese and Korean Voice Search. 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), s. 5149–5152. 4 sivua. Saatavissa [haettu 26.11.2024]: <https://ieeexplore.ieee.org/abstract/document/6289079>.

Sennrich, R., Haddow, B. & Birch, A. (2015). Neural Machine Translation of Rare Words with Subword Units. 11 sivua. Saatavissa [haettu 26.11.2024]: <https://arxiv.org/abs/1508.07909>.

Shakya, U. (2020). Steganography: Hiding Information inside Pictures. Julkaistu 3.6.2020. Saatavissa [haettu 1.10.2024]: <https://www.programiz.com/blog/steganography-hiding-information-inside-pictures/>.

Sharp, T. (2001). An Implementation of Key-Based Digital Signal Steganography. Information Hiding 2001, vol. 2137, s. 13–26. 14 sivua. Saatavissa [haettu 3.11.2024]: https://link.springer.com/chapter/10.1007/3-540-45496-9_2.

Shen, J., Ji, H. & Han, J. (2020). Near-Imperceptible Neural Linguistic Steganography via Self-Adjusting Arithmetic Coding. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, November 16–20, s. 303–313. 11 sivua. Saatavissa [haettu 2.11.2024]: <https://aclanthology.org/2020.emnlp-main.22/>.

Shirali-Shahreza, M. & Shirali-Shahreza, M. H. (2007). Text Steganography in SMS. 2007 International Conference on Convergence Information Technology, s. 2260–2265. 6 sivua. Saatavissa [haettu 25.9.2024]: <https://ieeexplore.ieee.org/document/4420590>.

SIMARGL (2020). Stegware – the Latest Trend in Cybercrime. SIMARGL, Secure Intelligent Methods for Advanced Recognition of Malware and Stegomalware Consortium. Julkaistu 14.2.2020. Saatavissa [haettu 14.5.2023]: <https://simargl.eu/blog/technical/stegware-the-latest-trend-in-cybercrime>.

Singh, K. U. & Singhal, A. (2017). Video Steganography Techniques: A Survey. International Journal on Recent and Innovation Trends in Computing and Communication, volume 5, issue 5, May 2017, s. 687–695. 9 sivua. Saatavissa [haettu 21.9.2024]: <https://ijritcc.org/index.php/ijritcc/article/view/587>.

Sinha, A. (2024). P-Tuning vs Prefix-Tuning vs Prompt-Tuning. Julkaistu 19.2.2024. Saatavissa [haettu 29.8.2024]: <https://ankitasinha0811.medium.com/ways-to-improve-an-llm-performance-while-keeping-the-llm-weights-frozen-7f67d740c5e2>.

Smith, C. S. (2023). What Large Models Cost You – There Is No Free AI Lunch. Forbes. Julkaistu 8.9.2023. Saatavissa [haettu 9.5.2024]: <https://www.forbes.com/sites/craigsmith/2023/09/08/what-large-models-cost-you--there-is-no-free-ai-lunch/>.

Solanki, K., Sarkar, A. & Manjunath, B. S. (2007). YASS: Yet Another Steganographic Scheme That Resists Blind Steganalysis. Information Hiding 2007, Lecture Notes in Computer Science, vol. 4567. Springer, Berlin, Heidelberg, Germany. 16 sivua. Saatavissa [haettu 28.10.2024]: https://link.springer.com/chapter/10.1007/978-3-540-77370-2_2.

Sotilaan käsikirja (2019). Puolustusvoimat. ISBN 978-951-25-3051-9. 323 sivua. Saatavissa: <https://puolustusvoimat.fi/asiointi/aineistot/ohjesaannot-ja-oppaat>.

Stöffelbauer, A. (2023). How Large Language Models Work: From Zero to ChatGPT. Julkaistu 24.10.2023. Saatavissa [haettu 5.8.2024]: <https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f>.

Taistelijan opas (2013). Maavoimien esikunta. ISBN 978-951-25-2485-3. 273 sivua. Saatavissa [haettu 14.3.2024]: <https://puolustusvoimat.fi/asiointi/aineistot/ohjesaannot-ja-oppaat>.

Taunk, D. (2020). L1 vs L2 Regularization: The Intuitive Difference. Saatavissa [haettu 15.11.2024]: <https://medium.com/analytics-vidhya/l1-vs-l2-regularization-which-is-better-d01068e6658c>.

Teubner, T., Flath, C. M., Weinhardt, C., van der Aalst, W. & Hinz, O. (2023). Welcome to the Era of ChatGPT et al.: The Prospects of Large Language Models. Business & Information Systems Engineering, 2023-04, vol. 65 (2), s. 95–101. 7 sivua. Saatavissa [haettu 25.7.2024]: https://libkey.io/libraries/2233/articles/564463260/full-text-file?utm_source=api_1105.

Thangavel, T. (2023). Limitations of Transformer Architecture. Julkaistu 14.9.2023. Saatavissa [haettu 6.7.2024]: <https://medium.com/@thirupathi.thangavel/limitations-of-transformer-architecture-4e6118cbf5a4>.

Tidmarsh, D. (2023). A Guide to Steganography: Meaning, Types, Tools, & Techniques. Julkaistu 12.4.2023. Saatavissa [haettu 3.11.2024]: <https://www.eccouncil.org/cybersecurity-exchange/ethical-hacking/what-is-steganography-guide-meaning-types-tools/>.

Toloka (2024). Prefix Tuning vs. Fine-Tuning and Other PEFT Methods. Julkaistu 17.4.2024. Saatavissa [haettu 21.8.2024]: <https://toloka.ai/blog/prefix-tuning-vs-fine-tuning/#prefix-funing-vs-prompt-tuning>.

Tuominen, H. (2024). Tekoälyn perusteita ja sovelluksia. Päivitetty 4.8.2024. Saatavissa [haettu 8.8.2024]: <https://tim.jyu.fi/view/kurssit/tie/tiep1000/tekoalyn-sovellukset/kirja#vb4lax4qaXvj>.

Turun yliopisto (2024). Euroopan avoin kielimalli Poro: eurooppalaisen tekoälyn ja kielten monimuotoisuuden virstanpylväs. Julkaistu 21.2.2024. Saatavissa [haettu 5.8.2024]: <https://www.utu.fi/fi/ajankohtaista/uutinen/euroopan-avoin-kielimalli-poro-eurooppalaisen-tekoalyn-ja-kielten>.

Valyaeva, A. (2023). People Are Creating an Average of 34 Million Images per Day. Statistics for 2024. Everypixel Journal. Julkaistu 15.8.2024. Saatavissa [haettu 30.11.2024]: <https://journal.everypixel.com/ai-image-statistics>.

Vartiainen, H., Tedre, M., Jormanainen, I., Kahila, J., Valtonen, T. & Toivonen, T. (2021). Tekoäly, koneoppiminen ja teknologinen murros: Kohti datatoimijuutta ja tulevaisuuden design-taitoja. Itä-Suomen yliopisto. Ainedidaktiikka 5(2) (2021), s. 103–120. 18 sivua. Julkaistu 31.12.2021. Saatavissa [haettu 29.3.2024]: <https://journal.fi/ainedidaktiikka/article/view/90776>.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017). Attention Is All You Need. 15 sivua. Julkaistu 12.6.2017, päivitetty 2.8.2023 (v7). Saatavissa [haettu 13.3.2024]: <https://arxiv.org/abs/1706.03762>.

Viebrock, S. (2024). Advanced ChatGPT: Context Window Limits and Prompt Engineering. Saatavissa [haettu 28.11.2024]: <https://www.o8.agency/blog/ai/advanced-chatgpt-context-windows-and-prompt-engineering>.

Vylomova, E., Rimell, L., Cohn, T. & Baldwin, T. (2015). Take and Took, Gaggle and Goose, Book and Read: Evaluating the Utility of Vector Differences for Lexical Relation Learning. 12 sivua. Julkaistu 5.9.2015, päivitetty 13.8.2016 (v4). Saatavissa [haettu 7.8.2024]: <https://arxiv.org/abs/1509.01692>.

Wang, H., Yang, Z., Yang, J., Chen, C. & Huang, Y. (2023). Linguistic Steganalysis in Few-Shot Scenario. *IEEE Transactions on Information Forensics and Security*, 2023-01, vol. 18, s. 4870–4882. 13 sivua. Saatavissa [haettu 9.10.2024]: <https://ieeexplore.ieee.org/document/10190738>.

Warkentin, M., Bekkering, E. & Schmidt, M. B. (2008). Steganography: Forensic, Security, and Legal Issues. *Journal of Digital Forensics, Security and Law*, Vol. 3(2), s. 17–34. 19 sivua. Saatavissa [haettu 29.8.2024]: <https://commons.erau.edu/jdfsl/vol3/iss2/2/>.

Wayner, P. (1992). Mimic Functions. *Cryptologia*, 16:3, s. 193–214. 23 sivua. Saatavissa [haettu 4.10.2024]: <https://doi.org/10.1080/0161-119291866883>.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. & Zhou, D. (2022b). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 43 sivua. Julkaistu 28.1.2022, päivitetty 10.1.2023 (v6). Saatavissa [haettu 15.8.2024]: <https://arxiv.org/abs/2201.11903>.

Wei, P., Li, S., Zhang, X., Luo, G., Qian, Z. & Zhou, Q. (2022a). Generative Steganography Network. *Proceedings of the 30th ACM International Conference on Multimedia (MM '22)*, October 10–14, 2022, Lisbon, Portugal. 9 sivua. Julkaistu 28.7.2022, päivitetty 14.8.2022 (v3). Saatavissa [haettu 1.12.2024]: <https://arxiv.org/abs/2207.13867>.

Wood, N. (2023). Steganography Breakthrough Could Be a Gift for Hackers. *Telecoms*. Julkaistu 7.3.2024. Saatavissa [haettu 1.12.2024]: <https://www.telecoms.com/security/steganography-breakthrough-could-be-a-gift-for-hackers>.

Yang, H. & Cao, X. (2010). Linguistic Steganalysis Based on Meta Features and Immune Mechanism. *Chinese Journal of Electronics*, vol. 19, no. 4, s. 661–666. 6 sivua. Julkaistu lokakuu 2010. Saatavissa [haettu 17.10.2024]: <https://ieeexplore.ieee.org/document/10168756>.

Yang, W.-C. & Chen, L.-H. (2008). A Novel Steganography Method via Various Animation Effects in PowerPoint Files. *2008 International Conference on Machine Learning and Cybernetics*, vol. 6, Kunming, China, July 12–15, s. 3102–3107. 6 sivua. Saatavissa [haettu 17.9.2024]: <https://ieeexplore.ieee.org/document/4620941>.

Yang, Y., Zha, L., Zhang, Z. & Wen, J. (2022). An Overview of Text Steganalysis. *The International Conference on Image, Vision and Intelligent Systems (ICIVIS 2021)*, s. 933–943. 11 sivua. Julkaistu marraskuu 2022. Saatavissa [haettu 28.10.2024]: https://www.researchgate.net/publication/358980605_An_Overview_of_Text_Steganalysis.

Yang, Z.-L., Guo, X.-Q., Chen, Z.-M., Huang, Y.-F. & Zhang, Y.-J. (2019). RNN-Stega: Linguistic Steganography Based on Recurrent Neural Networks. *IEEE Transactions on Information Forensics and Security*, 2019-05, vol. 14 (5), s.1280–1295, article 8470163. 16 sivua. Saatavissa [haettu 2.11.2024]: <https://ieeexplore.ieee.org/document/8470163>.

Yhdysvaltain oikeusministeriö (2023). Former GE Power Engineer Sentenced for Conspiracy to Commit Economic Espionage. U.S. Department of Justice, Office of Public Affairs. Lehdistöiedote 3.1.2023. Saatavissa [haettu 17.9.2024]: <https://www.justice.gov/opa/pr/former-ge-power-engineer-sentenced-conspiracy-commit-economic-espionage>.

Yin, Z., Wang, H., Horio, K., Kawahara, D. & Sekine, S. (2024). Should We Respect LLMs? A Cross-Lingual Study on the Influence of Prompt Politeness on LLM Performance. *Proceedings of the Second Workshop on Social Influence in Conversations (SICon 2024)*, Miami, Florida, United States, s. 9–35. Association for Computational Linguistics. 27 sivua. Saatavissa [haettu 28.11.2024]: <https://aclanthology.org/2024.sicon-1.2/>.

Ziegler, Z. M., Deng, Y. & Rush, A. M. (2019). *Neural Linguistic Steganography*. 9 sivua. Saatavissa [haettu 22.9.2024]: <https://arxiv.org/abs/1909.01496>.

LIITE: PEITTÄMINEN MICROSOFT COPILOTIN AVULLA

#	Microsoft Copilotin (ChatGPT) vastaus	Käyttäjän kehote
		<p>Copilot, tässä on lista sanoista ja niiden korvaajista puolipisteellä (;) erotettuna:</p> <p>ajoneuvo;käpylintu drooni;hippiäinen kenttäsaaraala;kanto sotilas:käki tukikohta;juurakko tuntematon;tulipunainen</p> <p>Korvaa seuraavassa virkkeessä sanat, joille on listassa annettu jokin korvaava sana. Säilytä alkuperäisen sanan kielioppi:</p> <p>Näin tieää monta ajoneuvoa matkalla kenttäsaaraalaan. (1)</p>
1	<p>Tässä on virke korvatuilla sanoilla:</p> <p>“Näin tieää monta käpylintua matkalla kanton.”</p> <p>Onko jotain muuta, missä voisin auttaa?</p>	<p>Tee sama myös näille virkkeille:</p> <p>Tuntemattomia sotilaita on matkalla tukikohtaan. Lähettäkää droonit.</p>
2	<p>Tässä ovat virkkeet korvatuilla sanoilla:</p> <p>“Tulipunaisia käkiä on matkalla juurakkoon. Lähettäkää hippiäiset.”</p> <p>Onko vielä muita virkeitä, joihin tarvitsisit apua?</p>	<p>Tee sama tälle virkkeelle:</p> <p>tuntematon ajoneuvo matkalla kenttäsaaraalaan, lähettäkää droonit</p>
3	<p>Tässä on virke korvatuilla sanoilla:</p> <p>“Tulipunainen käpylintu matkalla kanton, lähettäkää hippiäiset.”</p>	

Microsoft Copilotin (ChatGPT) vastaus

Käyttäjän kehote

Tarvitsetko apua vielä jonkin muun kanssa? (2)

Copilot, tässä on uusi lista, käytä tätä. Nyt rivin ensimmäinen on sana, joka halutaan korvata, ja sen korvaajat on erotettu puolipisteellä (;).

ajoneuvo;käpylintu;hippiäinen;käki
 drooni;hippiäinen;juurakko;kanto kenttä-
 sairaala;kanto;kanto;käpylintu soti-
 las;käki;käki;hippiäinen tukikohta;juu-
 rakko;käpylintu;juurakko tuntematon;tuli-
 punainen;taivaansininen;ruohonvihreä

Copilot, toimi seuraavasti. Korvaa seuraavassa virkkeessä (3)

4 Ymmärretty! Tässä on virke korvatuilla sanoilla uuden listan mukaan:

"Tulipunainen käpylintu matkalla kantoon, lähettäkää hippiäiset."

Onko jotain muuta, missä voisint auttaa?

Copilot, korvaa seuraavassa virkkeessä sanat uusilla versioilla siten, että lopputulokseksi muodostuu kolme uutta virkettä. Toisin sanoen ensimmäisessä versiossa käytä korvaamiseen ensimmäisen puolipisteen jälkeisiä sanoja, toisessa versiossa toisen puolipisteen jälkeisiä sanoja ja kolmannessa versiossa kolmannen puolipisteen jälkeisiä sanoja.

Korvattava virke:

tuntematon ajoneuvo matkalla kenttäsairaalaan, lähettäkää droonit

5 Tässä ovat kolme uutta virkettä korvatuilla sanoilla:

1. Tulipunainen käpylintu matkalla kantoon, lähettäkää hippiäiset.


2. Taivaansininen hippiäinen matkalla kantoon, lähettäkää juurakko.

3. Ruohonvihreä käki matkalla käpylintuun, lähettäkää kanto.

#	Microsoft Copilotin (ChatGPT) vastaus	Käyttäjän kehote
	Toivottavasti tämä auttaa! Onko jotain muuta, missä voisin olla avuksi?	Erinomaista, Copilot! Teit kuitenkin pienen virheen: sana "droonit" on monikossa. Näin ollen sen korvaavan sanankin pitäisi olla monikossa. Lause 1 on oikein, mutta voisitko korjata lauseet 2 ja 3?
6	Totta, kiitos huomiosta! Tässä ovat korjatut lauseet: 1. Tulipunainen käpylintu matkalla kantoon, lähettäkää hippiset. 2. Taivaansininen hippiäinen matkalla kantoon, lähettäkää juurakot. 3. Ruohonvihreä käki matkalla käpylintuun, lähettäkää kannot.	
	Toivottavasti nämä ovat nyt oikein! Onko muuta, missä voisin auttaa?	Erinomaista, Copilot! Lauseet ovat oikein.
7	Kiitos! Olen iloinen, että lauseet ovat nyt oikein. Jos tarvitset apua muissa tehtävissä tai sinulla on muita kysymyksiä, olen täällä auttamassa! 😊	Copilot, tehdään vielä testi. Sama tehtävänanto kuin aiemmin. Laadi kolme versiota seuraavasta virkkeestä: käpylintu lentelee taivaalla, viekää tuntemattomat sotilaat kenttäsaaralaan
8	Tässä ovat kolme versiota annettusta virkkeestä: 1. Hippiäinen lentelee taivaalla, viekää tulipunaiset kädet kanton. 2. Käki lentelee taivaalla, viekää taivaansiniset kädet kanton. 3. Käki lentelee taivaalla, viekää ruohonvihreät hippiset käpylintuun.	
	Toivottavasti nämä ovat oikein! Tarvitsetko apua vielä jossain muussa?	

#	Microsoft Copilotin (ChatGPT) vastaus	Käyttäjän kehote
9	<p>Tässä ovat korjatut virkkeet pienellä kirjoitettuna ja ilman loppupistettä:</p> <ol style="list-style-type: none"> 1. hippiäinen lentelee taivaalla, viekää tulipunaiset käet kantoon 2. käki lentelee taivaalla, viekää taivaansiniset käet kantoon 3. käki lentelee taivaalla, viekää ruohonvihreät hippiäiset käpylintuun <p>Toivottavasti nämä ovat nyt oikein! Onko muuta, missä voisin auttaa?</p>	<p>Copilot, tämä testi oli vaikea, ja siinä oli pari asiaa korjattavaksi. Ensimmäinen korjaus: Antaisitko virkkeet pienellä kirjoitettuna ja ilman loppupistettä, kuten alkuperäinenkin virke oli?</p> <p>Kiitos, Copilot. Sitten vielä se vaikea korjaus: antamassani sanalistassa vain ensimmäinen sana korvataan, jos sitä esiintyy. Jos tekstissä esiintyy jokin korvaavista sanoista eli puolipisteellä erotetuista sanoista, sinun ei tarvitse korvata niitä. Osaisitko muuttaa virkeitä tämän tiedon pohjalta?</p>
10	<p>Ymmärretty! Tässä ovat virkkeet korjattuina siten, että vain ensimmäinen sana korvataan, jos se esiintyy:</p> <ol style="list-style-type: none"> 1. hippiäinen lentelee taivaalla, viekää tuntemattomat sotilaat kantoon 2. käki lentelee taivaalla, viekää tuntemattomat sotilaat kantoon 3. käki lentelee taivaalla, viekää tuntemattomat sotilaat käpylintuun <p>Toivottavasti nämä ovat nyt oikein! Onko muuta, missä voisin auttaa?</p>	<p>Vielä ei mennyt oikein, yritetään uudelleen.</p> <p>Alkuperäinen lause:</p>

# Microsoft Copilotin (ChatGPT) vastaus	Käyttäjän kehote
<p>11 Ymmärretty! Tässä ovat virkkeet korjattuina siten, että vain tarvittavat sanat korvataan:</p> <p>1. käpylintu lentelee taivaalla, viekää tulipunaiset käet kantoon</p> <p>2. käpylintu lentelee taivaalla, viekää taivaansiniset käet kantoon</p> <p>3. käpylintu lentelee taivaalla, viekää ruohonvihreät hippiäiset käpylintuun</p> <p>Toivottavasti nämä ovat nyt oikein! Onko muuta, missä voisin auttaa?</p>	<p>käpylintu lentelee taivaalla, viekää tunte mattomat sotilaat kenttäsaaraalaan</p> <p>Korvaava lause 1. oikein:</p> <p>käpylintu lentelee taivaalla, viekää tulipunaiset käet kantoon</p> <p>Huomaatko? Säilytä sana "käpylintu" sellaisenaan, koska sitä ei tarvitse korvata.</p> <p>Erinomaista, Copilot! Kiitos paljon.</p>
<p>12 Ole hyvä! Olen iloinen, että pystyin auttamaan. Jos sinulla on lisää kysymyksiä tai tarvitset apua muissa asioissa, olen täällä auttamassa!</p> <p>😊</p>	<p>Copilot, tehdään vielä testi. Nyt sinun tehtäväsi on palauttaa virkkeessä alkuperäiseen muotoonsa. Antamassani virkkeessä sanat on siis jo korvattu toisilla. Käytetään samaa listaa kuin aiemmin, mutta nyt olen lisännyt listaan otsikkorivin:</p>
<p>13 Olen valmis kokeilemaan! Anna minulle virke, jonka haluat palauttaa alkuperäiseen muotoonsa, ja</p>	<p>alkuperäinen sana;salaus1;salaus2;salaus3</p> <p>Jos kerron sinulle, että käytössä on salaus2 ja pyydän palauttamaan alkuperäisen virkkeen virkkeestä "käpylintuja matkalla tulipunaiseen juurakkoon", oikea vastaus olisi "ajoneuvoja matkalla tuntemattomaan tukikohtaan". Oletko valmis kokeilemaan?</p>

#	Microsoft Copilotin (ChatGPT) vastaus	Käyttäjän kehote
	kerro, mikä salaus on käytössä. Voimme aloittaa heti!	Palauta alkuperäiseen muotoonsa seuraava virke. Käytössä on salaus3:
		ruohonvihreitä käpylintuja havaittu, lähettäkää kannot tutkimaan
14	Tässä on virke palautettuna alkuperäiseen muotoonsa käyttäen salaus3:	
	tuntemattomia ajoneuvoja havaittu, lähettäkää dronit tutkimaan	
	Onko jotain muuta, missä voisin auttaa?	
15	Kiitos! Olen iloinen, että se meni oikein. Jos sinulla on lisää kysymyksiä tai tarvitset apua muissa asioissa, olen täällä auttamassa! 	Erinomaista, Copilot! Se oli oikein
		Tehdään vielä toinen testi. Sama tehtävänanto kuin äsken eli annan sinulle virkkeen, jossa sanoja on korvattu toisilla, ja sinun tehtäväsi on palauttaa alkuperäinen virke. Käytössä on salaus2. Palautettava virke:
		tukikohtia ja käkiä havaittu, lähettäkää hippiäiset tutkimaan ja juurakot varmistamaan
16	Tässä on virke palautettuna alkuperäiseen muotoonsa käyttäen salaus2:	
	tukikohtia ja sotilaita havaittu, lähettäkää dronit tutkimaan ja kenttäsaairaalat varmistamaan	
	Toivottavasti tämä on oikein! Onko jotain muuta, missä voisin auttaa?	
		Ennen pilkkua oleva osuus on oikein, mutta loppuosa ei. Vaihdot vahingossa salaus1:een, mutta sinun pitää käyttää vain salaus2 koko virkkeelle. Osaisitko korjata?
17	Totta, pahoittelut virheestä. Tässä on korjattu virke käyttäen vain salaus2:	
	tukikohtia ja käkiä havaittu, lähettäkää hippiäiset tutkimaan ja kannot varmistamaan	

Microsoft Copilotin (ChatGPT) vastaus

Käyttäjän kehote

Toivottavasti tämä on nyt oikein!
Onko muuta, missä voisın auttaa?

Vielä ei mennyt oikein. Annan tehtävänannon uudestaan. Palauta annettu virke alkuperäiseen muotoonsa. Käytössä on salaus2.

Palautettava virke:

tukikohtia ja käkiä havaittu, lähettäkää hippiaiset tutkimaan ja juurakot varmistamaan

- 18 Ymmärretty! Tässä on virke palautettuna alkuperäiseen muotoonsa käyttäen salaus2:

tukikohtia ja sotilaita havaittu, lähettäkää droonit tutkimaan ja kenttäsairaalat varmistamaan

Toivottavasti tämä on nyt oikein!
Onko muuta, missä voisın auttaa?

Tässä on vielä virhe. salaus2 mukaan sana "ajoneuvo" korvataan sanalla "hippiäinen" ja sana "drooni" korvataan sanalla "juurakko". Nyt lauseen jälkimmäisessä osassa "lähettäkää hippiaiset tutkimaan ja juurakot varmistamaan" on sanat "hippiaiset" ja "juurakot" palautettava alkuperäisiksi sanoiksi. Tekisitkö tämän?

- 19 Ymmärretty! Tässä on virke palautettuna alkuperäiseen muotoonsa käyttäen salaus2:

tukikohtia ja sotilaita havaittu, lähettäkää ajoneuvot tutkimaan ja droonit varmistamaan

Toivottavasti tämä on nyt oikein!
Onko muuta, missä voisın auttaa?

Täysin oikein! Korjasit hienosti, Copilot.

- 20 Kiitos paljon! Olen iloinen, että se meni oikein. Jos sinulla on lisää kysymyksiä tai tarvitset apua muissa asioissa, olen täällä auttamassa! 😊

(1) Tehtävänannossa kirjoitusvirhe. Sanan *tiellä* sijaan lukee *tieää*.

(2) Copilotia ei korjattu siitä, että virke aloitettiin isolla kirjaimella, kuten oikeakielisesti kuuluukin.

(3) Tehtävänanto katkeaa kesken.