Tampereen yliopisto

Valtteri Sutelainen

TEKOÄLYN VAIKUTUS OHJELMOINTITYÖHÖN

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintutkielma
Kesäkuu 2025

TIIVISTELMÄ

Valtteri Sutelainen: Tekoälyn vaikutus ohjelmointityöhön
Kandidaatintutkielma
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaattiohjelma
Kesäkuu 2025

Laajat kielimallit ovat neuroverkkoja, jotka kuuluvat koneoppimisen osa-alueeseen. Nämä ovat koulutettu suurella määrällä tekstidataa, jonka avulla ne vastaavat käyttäjän syötteisiin.

Tekoäly tai tarkemmin laajat kielimallit ovat olleet yleisessä käytössä muutaman vuoden. Monet yrityksetkin ovat alkaneet suosia tekoälyä työvälineenä erilaisiin työtehtäviin. Varsinkin ohjelmoinnissa on jo puhuttu paljon työpaikkojen korvaamisesta tekoälyllä.

Kandidaattityössä tutkitaan kirjallisuuskatsauksena laajojen kielimallien vahvuuksia ja heikkouksia ohjelmoinnissa. Tavoitteena on löytää vastauksia; miten laajoja kielimalleja hyödynnetään tällä hetkellä, miten näihin suhtaudutaan ja mitä parannettavaa niissä on? Lisäksi pohditaan parantaako tekoäly ihmisten tuottavuutta ohjelmointityössä. Tutkielmassa tutkitaan ja käytetään ajankohtaisia artikkeleita sekä kyselyitä kehittäjiltä, liittyen tekoälyyn ja sen käyttöön.

Kehittäjät käyttivät eniten tekoälyä koodin luomiseen, hakemiseen ja virheiden korjaamiseen. Kehittäjien mukaan tekoälyn käyttö on tehostanut heidän ohjelmointityötänsä, mutta tekoäly ei pysty vielä ratkaisemaan ongelmitta vaikeimpia ohjelmointitehtäviä. Tekoälyn ongelmina olivat tietoturva-, lailliset- ja eettiset ongelmat. Näiden ongelmien takia tekoäly ei nopeuta ohjelmointia kaikissa tapauksissa, eikä pysty korvaamaan ohjelmointia kokonaan.

Laajat kielimallit tällä hetkellä eivät tule korvaamaan ihmistyötä ohjelmoinnissa, mutta joitakin aloittelijoiden työpaikkoja voidaan korvata tekoälyllä. Tekoälyä tullaan käyttämään ohjelmoinnissa työkaluna, mutta sen ongelmien vuoksi ammattitietoisuus on ja tulee olemaan tärkeää ohjelmointitöissä. Tekoäly kehittyy nopeasti, jonka takia tutkimukset voivat vanheta lyhyessä ajassa, eikä tekoälyn liiallisen käytön vaikutuksia vielä tiedetä. Tämän takia jatkotutkimusten tekeminen aiheesta on olennaista.

Avainsanat: tekoäly, laajat kielimallit, ohjelmistokehitys, ohjelmointi, ohjelmointityö, kehittäjä

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin Originality Check -ohjelmalla.

TEKOÄLYN KÄYTTÖ OPINNÄYTTEESSÄ

Opinnäytteessäni on käytetty tekoälysovelluksia:

☐ Ei
☒ Kyllä

Ilmoitukseni mukaan olen käyttänyt opinnäytteessäni tutkielmaprosessin aikana seuraavia tekoälysovelluksia: ChatGPT ja Phind

Tekoälysovellusten nimet ja versiot:
Phind versio Phind Instant
ChatGPT versio GPT-4o

Käyttötarkoitus: Käytettiin hahmottamaan mahdollisia aiheeseen liittyviä teemoja ja tarkistamaan, että aiheen rajaus on tarpeeksi erilainen muihin kandidaatintöihin verrattuna. Tehtiin esimerkki, miten tekoäly pystyy generoimaan Python ohjelman käyttäjän syötteen avulla.

Osiot, joissa tekoälyä on käytetty: Luvussa 3.2 Tekoälyn käyttötavat generoitu tekoälyllä Python ohjelma.

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

SISÄLLYSLUETTELO

1.JOHDANTO 1
2.LAAJAT KIELIMALLIT 4
3.TEKOÄLYN HYÖDYNTÄMINEN OHJELMOINNISSA 6
3.1 Suosio 6
3.2 Käyttötavat 6
4.TEKOÄLYN ONGELMAT 8
4.1 Tekoälyn ohjaaminen 8
4.2 Tietoturva 8
4.3 Vaikeiden tehtävien ratkaiseminen 9
4.4 Lailliset ja eettiset 10
4.5 Kattavuus 10
5.TULOKSIEN POHDINTA 11
6.YHTEENVETO 13
LÄHTEET 14

LYHENTEET JA MERKINNÄT

AGI	Yleinen tekoäly (Artificial General Intelligence)
AI	Tekoäly (Artificial intelligence)
ChatGPT	OpenAI:n luoma laaja kielimalli (Generative Pre-trained Transformer)
IDE	Kehitysympäristö (Integrated development environment)
LLM	Laajat kielimallit (Large language model)

1. JOHDANTO

Tekoälyn (*engl. AI - Artificial intelligence*) käyttäminen on lisääntynyt viime vuosien aikana. Mediassa puhutaan aktiivisesti laajoista kielimalleista (*engl. LLM – Large language model*), jotka ovat tekoälyyn liittyvät osajoukko. Nämä ovat saaneet huomiota yleisessä käytössä niiden kykyjen takia. Tekoäly pystyy luomaan koodia ja ratkaisemaan ohjelmointiin liittyviä ongelmia, jonka takia on syntynyt paljon keskustelua, siitä tuleeko tämä korvaamaan tulevaisuudessa ohjelmointityön kokonaan. Ohjelmointityö kehittyy koko ajan ja työtapojen muuttuminen alalla ei ole mitään uutta. Tulevaisuudessa voidaankin olettaa tekoälyn tulevan olemaan työkaluna suuressa roolissa ohjelmistoinsinöörien, kehittäjien ja data-analyytikkojen tehtävissä [1].

Tekoäly tarkoittaa tietokoneohjelmaa, joka koulutetaan haluttuun käyttötarpeisiin. Tekoäly kykenee koulutuksen perusteella oppimaan ratkaisemaan vanhoja ongelmia ja ennalta näkemättömiä ongelmia. Koneoppiminen (*engl. Machine learning*) on yksi tekoälyyn liittyvä osajoukko, joka tarkoittaa tietokoneohjelmaa, jota koulutetaan koulutusdatalla. Koneoppimisessa tietokoneohjelma oppii tunnistamaan oikeat ja väärät vastaukset koulutusdatasta tehtyjen havaintojen avulla. Koneoppimista käytetään esimerkiksi opettamaan tietokoneohjelma tunnistamaan autot kuvasta. [2] Tutkielmassa keskitytään laajojen kielimallien käyttämiseen ohjelmointityössä. Laajat kielimallit perustuvat koneoppimiseen, ja tähän tekoälyn osa-alueeseen. Tekoälyä hyödynnetään paljon myös muissa tapauksissa, kuten robotiikassa, puheen tunnistamisessa, yritysten tuotteiden suositelluissa, lääketieteessä ja pelien pelaamisessa, kuten shakissa [2].

Tässä tutkielmassa keskitytään tutkimaan laajojen kielimallien vaikutusta ohjelmointityöhön. Suosittuihin laajoihin kielimalleihin tällä hetkellä voidaan lukea: GPT-4o, Claude Sonnet 3.5, Gemini 2.0 Flash, Llama 4 ja DeepSeek-V3. Laajoja kielimalleja tulee jatkuvasti uusia ja niitä kehitetään jatkuvasti, joten näihin liittyvää tutkimusta on syytä tehdä usein. Nykypäivänä tunnetuin keskustelubotti on ChatGPT, joka on kerännyt paljon huomiota. Helmikuussa 2025 OpenAI:n kehittämä ChatGPT saavutti 400 miljoona käyttäjää [3].

Aihe on ajankohtainen, sillä laajojen kielimallien käyttäminen ohjelmointityössä on uutta, eikä näiden vaikutusta tai tehokkuutta tiedetä vielä täysin. Tekoälyn uskotaan vaikuttavan tulevaisuudessa suuresti ohjelmointityöhön. Nykypäivänä moni käyttääkin jo laajoja kielimalleja apunaan ohjelmoinnissa. Yritykset ovat alkaneet kiinnittämään enemmän

huomiota tekoälyn käyttämiseen ja sen hyödyntämiseen työssä. Shopifyn toimitusjohtaja Tobi Lutke, vaatiikin työntekijöiden pystyvän todistamaan, ettei työtä voida suorittaa tekoälyn avulla ennen, kuin he saavat lisää työvoimaa tehtävän suorittamiseen [4]. Yritykset ovat alkaneet ottamaan tekoälyä enemmän käyttöön ja kiinnittävät huomiota sen vahvuuksiin.

Keskustelua käydään myös ohjelmointityön kokonaan katoamisesta, mutta tällä hetkellä se ei näytä olevan todennäköistä. Muita samaan aiheeseen liittyviä kandidaattitutkielmia on tehty, kuten Karvisen 2024 kirjoittama kandidaattitutkielma tekoälyn hyödyistä ja haitoista ohjelmistotuotannossa [5]. Karvisen kandidaattitutkielmassa tutkitaan tekoälyn haittoja ja hyötyjä pyrkien vastaamaan miten tekoälyn tulee muuttamaan ohjelmistotuotannon alaa. Tässä tutkitaan asiaa hieman eri näkökulmasta keskittyen enemmän hyödyntämään ajankohtaisia tilastoja ja etsimään vastauksia kysymyksiin; miten ammattikehittäjät suhtautuvat näihin, mitä parannettavaa näissä on, lisäävätkö nämä ihmisten tuottavuutta ohjelmoinnissa ja miten ammattilaiset tällä hetkellä hyödyntävät tekoälyä ohjelmointityössä.

Tekoäly on kehittynyt ja muuttunut paljon viimevuosien aikana. Myös ohjelmointiin liittyvät tekoäly työkalut ovat kehittyneet nopeasti. Tekoälyä integroidaan moniin eri kehitysympäristöihin (*engl. IDE - Integrated development environment*) tavoitteena parantaa kehittäjien tehokkuutta. Kehitysympäristöjä käytetään tekemään ohjelmoinnista mahdollisimman suoraviivaista ja tehokasta. Yksi suosituimpia kehitysympäristöjä on Visual Studio Code, josta on tehty uusia haarautumia kuten Codeium ja Cursor. Näihin on sisäänrakennettu tekoälyagentti auttamaan kehitystyössä. Tekoälyagentti kykenee luomaan tiedostoja, ehdottamaan erilaisia paketteja ladattavaksi ja tekemään paljon muuta ohjelmointiprojekteihin liittyvää työtä. [6] Tekoagentit tarkoitus on helpottaa ohjelmistokehittäjien työtä, sillä tekoälyagentti pystyy helposti hahmottamaan ohjelmointiprojektien kokonaisuutta ja tehdä suuria muutoksia nopeuttaen työtä.

Tutkielmassa käydään läpi laajojen kielimallien toimintaa, niiden erilaisia käyttötapoja, vaikutuksia ohjelmointiin, heikkouksia, riskejä ja ongelmia. Kandidaatintutkielma on tehty kirjallisuuskatsauksena, jossa käsitellään tekoälyn käyttämistä ohjelmoinnissa tutkimusten ja kyselyiden avulla. Lähteitä haettiin seuraavista tieteellisistä tietokannoista, ACM (Digital Library), Science Database (ProQuest), Andor ja IEEE Xplore. Näissä tietokannoissa käytettiin hakutermejä: ”software engineering”, ”developer”, ”software”, ”privacy”, ”llm”, ”large language model”, ”AI”, ”ChatGPT”, ”local” ja ”performance”. Hakutermien avulla muodostettiin erilaisia hakuja, joiden avulla löydettiin lähteitä tieteellisistä tietokannoista. Näiden lisäksi on etsitty lähteitä internetistä Googlella, saavuttaakseen mahdollisimman ajankohtaisia tuloksia tekoälyn suosiosta ja sen käytöstä ohjelmoinnissa.

Luvussa 2 käsitellään laajojen kielimallien toimintaa ja miten nämä luodaan. Luvussa 3 käsitellään, kuinka paljon laajoja kielimalleja käytetään- ja miten näitä hyödynnetään ohjelmoinnissa. Luvussa 4 käydään läpi laajojen kielimallien heikkouksia ja ongelmia. Luvussa 5 pohditaan ja analysoidaan työn tuloksia. Luvussa 6 on yhteenveto koko työstä.

2. LAAJAT KIELIMALLIT

Laajat kielimallit ovat niin sanottuja neuroverkkoja, joita opetetaan suurilla datamäärillä. Laajat kielimallit oppivat datan avulla vastaamaan käyttäjien kysymyksiin todennäköisimmän vastauksen. [7] Esimerkiksi käyttäjä voi kysyä mikä on Suomen pääkaupunki, jolloin laaja kielimalli tunnistaa kysymyksen ja etsii todennäköisimmän vastauksen tähän kysymykseen koulutusdatan avulla. Todennäköisin vastaus on se joka, esiintyy eniten koulutusdatassa. Laajat kielimallit pyrkivät tällä tavalla muodostamaan koulutusdatan pohjalta parhaan mahdollisen vastauksen käyttäjälle.

Laajoja kielimalleja opetetaan koulutusdatan avulla, joka sisältää paljon tekstidataa. Laajat kielimallit vertaavat omaa ennustamaansa arvoa oikeaan arvoon muodostaen havainnon, jonka avulla nämä pystyvät parantamaan parametrejä ja vähentämään eroavaisuuksia omien ja oikeiden vastausten välillä. [2] Laajojen kielimallien kouluttamisessa käytetään ohjaamatonta oppimista (*engl. Unsupervised learning*). Tässä tavassa laajat kielimallit analysoivat tekstidataa, josta ne yrittävät löytää oikean tiedon erilaisien yhtenäisyyksien avulla. Tämä on toimiva tapa, sillä koulutusdataan ei tarvitse manuaalisesti käydä merkitsemässä oikeita vastauksia. [8]

Laajat kielimallit sisältävät satoja miljardeja parametreja. Nämä ovat ennalta koulutettuja suurella määrällä tekstidataa mahdollistaen laadukkaiden vastausten saamisen pienellä syötteellä. [9] Koska laajojen kielimallien kouluttamiseen käytetään ohjaamatonta oppimista, on mahdollista käyttää suuria tekstimääriä mallien kouluttamiseen. Tekstidataa laajoille kielimalleille kerätään yleensä avoimista lähteistä, mutta tämä voi tuottaa ongelmia. Tähän liittyen on syntynyt keskustelua siitä, kuka laillisesti omistaa tekoälyn generoiman sisällön. Tätä aihetta käsitellään myöhemmin luvussa 4.4.

Tekoäly, joka voisi korvata ihmisten tekemän ohjelmointityön kokonaan on Yleinen tekoäly (*engl. Artificial General Intelligence - AGI*). Tämä tekoäly pystyisi oppimaan suorittamaan kaikkia asioita mitä ihmisetkin osaavat tehdä [2]. Yleistä tekoälyä ei ole vielä kehitetty ja laajoilla kielimalleilla on siis suuri ero tähän. Laajat kielimallit eivät ajattele samalla tavalla kuin ihmiset, vaan käyttävät olemassa olevaa materiaalia, muodostamalla niistä todennäköisimmän vastauksen. Tämän takia nämä eivät ole täydellinen ratkaisu eri alojen työntekijöiden korvaamiseen.

Tällä hetkellä laajoille kielimalleille löytyy monia eri käyttötapoja. Niitä voidaan käyttää luomaan kuvia, tekstejä, koodia, yhteenvetoja ja ennustuksia datan avulla. Käytännössä lääketieteessä voidaan pystyä tunnistamaan sairauksien lopputuloksia paremmin tai fi-

nanssialalla luoda yhteenvetoja yritysten tuloksista. [8] Laajoja kielimalleja voidaan hyödyntää monella eri tavalla ja monella eri alalla. Laajojen kielimallien hyödyntämistä ohjelmointityössä käsitellään tarkemmin seuraavassa luvussa.

3. TEKOÄLYN HYÖDYNTÄMINEN OHJELMOINNISSA

Tässä luvussa käydään erilaisten tutkimusten ja kyselyiden pohjalta asioita liittyen tekoälyn käyttämiseen. Luvussa yhdistetään kehittäjiltä kyselyistä kerättyjä vastauksia tutkimuspohjaisiin tuloksiin ja vastataan kysymyksiin; miten ammattikehittäjät suhtautuvat tekoälyn käyttämiseen, miten tätä hyödynnetään ja lisääkö tämä kehittäjien tuottavuutta. Luvussa 3.1 käsitellään tekoälyn suosiota. Luvussa 3.2 käydään läpi erilaisia ohjelmointiin liittyviä tekoälyn käyttötapoja, sisältäen esimerkin tämän käytöstä.

3.1 Suosio

Tekoälyn käyttö ohjelmoinnissa on selvästi suuressa suosiossa. Tämä voidaan havaita Stack Overflow'n kyselyssä, johon oli vastannut 46049 ammattikehittäjää [10]. Heistä 63,2 % vastasi käyttäneen tekoälyä työssänsä. Tekoälyn menestys voidaan myös havaita internetissä, sosiaalisessa mediassa ja uutisissa, joissa jatkuvasti keskustellaan aiheesta. Johdannossa todettu ChatGPT:n käyttäjämäärä myös kuvastaa laajojen kielimallien yleisyyttä ohjelmoinnissa.

3.2 Käyttötavat

Kehittäjät käyttävät tekoälyä ohjelmoinnissa eniten koodin tuottamiseen. Kyselyn [10] tuloksessa 35978 vastaajasta 82 % sanoi käyttävänsä tekoälyä tähän. Toiseksi yleisin käyttötapa on vastausten hakeminen, jota käyttää 67,5 % ja kolmanneksi suosituin 56,7 % on virheiden korjaaminen ja avun saantiin. 40,1 % vastaajista sanoi käyttävänsä tekoälyä koodin dokumentaatioon. Ammattikehittäjistä 23581 (82,7 %) henkilöä vastasi tekoälyn nopeuttavan heidän työtänsä. [10] Myös muissa tutkimuksissa kehittäjät ovat havainneet tekoälyn parantavan heidän tuottavuuttansa ohjelmoinnissa [11], [12].

Alla olevassa kuvassa 1 on annettu esimerkki siitä, miten tekoälyä voidaan käyttää ohjelmoinnissa koodin luontiin ja sen dokumentointiin. Tekoäly myös pystyy muokkaamaan ja selittämään koodia käyttäjän tarpeiden mukaisesti. ChatGPT-4o mallille syötettiin englanniksi seuraava syöte "Create python script which reads every line of a text file, sorts them into alphabetical order and writes them into a new text file. Also add descriptive documentation to the code." Vastaukseksi saatiin Python ohjelma, joka suorittaa kysytyn tehtävän. Laajat kielimallit yleensä kommentoivat koodia jo valmiiksi, mutta dokumentaatiota kysyessä saatiin enemmän tietoa ohjelman käyttämisestä ja sen toiminnasta. Ohjelma testattiin täysin toimivaksi.

Sort Text Lines

```python
"""
This script reads lines from an input text file, sorts them alphabetically,
and writes the sorted lines to a new output file.

Usage:
    1. Place the input text file in the same directory as this script.
    2. Update the 'input_filename' variable with the name of the input file.
    3. Run the script to generate the sorted output file.
"""

def sort_file_lines(input_filename, output_filename):
    """
    Reads lines from the input file, sorts them alphabetically,
    and writes the sorted lines to the output file.

    :param input_filename: str, name of the input file containing unsorted lines
    :param output_filename: str, name of the output file to store sorted lines
    """
    try:
        # Open the input file and read lines
        with open(input_filename, 'r', encoding='utf-8') as infile:
            lines = infile.readlines()

        # Sort the lines alphabetically
        sorted_lines = sorted(line.strip() for line in lines)  # strip() removes leading/trailing spaces

        # Write the sorted lines to the output file
        with open(output_filename, 'w', encoding='utf-8') as outfile:
            outfile.write("\n".join(sorted_lines) + "\n")

        print(f"Successfully sorted the lines and saved to '{output_filename}'.")
    except FileNotFoundError:
        print(f"Error: The file '{input_filename}' was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")

# Example usage
input_filename = "input.txt"  # Change this to your input file name
output_filename = "sorted_output.txt"  # Change this to your desired output file name
sort_file_lines(input_filename, output_filename)
```

Kuva 1. *ChatGPT-4o mallilla generoitu Python ohjelma*

Tehokkaimmiksi tavoiksi käyttää tekoälyä koettiin ohjelmistokehityksen hallinta ja optimointi, sekä uusien ominaisuuksien toteuttaminen. Näihin tapoihin tarvittiin keskimäärin vähemmän ihmisten syötteitä tekoälylle, jotta saatiin haluttu lopputulos. Hitaimpia olivat dokumentaation generoiminen, ympäristön oikeiden asetusten määrittäminen ja koodin laadunhallinta [13]. Kuitenkin kyselyn [10] mukaan 35978 kehittäjästä 40,1 % dokumentoi koodia tekoälyn avulla. Tätä tekoälytyökalujen tekijät voisivat varmasti keskittyä parantamaan, sillä tähän löytyy kysyntää. Oikean kehitysympäristön käyttö vaikuttaa paljon, kuinka nopeasti tekoälyä voidaan hyödyntää. Esimerkiksi johdannossa mainittu Cursor-kehitysympäristö sisältää tekoälyn integroituna tähän alustaan, mahdollistaen nopeamman tekoälyn hyödyntämisen.

4. TEKOÄLYN ONGELMAT

Tekoälyn käytössä esiintyy ongelmia, joten sitä ei voida pitää täydellisenä ratkaisuna ohjelmointiin. Erityisesti vaativissa ja arkaluontoisissa tapauksissa tekoälyn käytön kanssa pitää olla varovainen. Esimerkiksi terveydenhuoltoon tai lentokoneiden ohjaukseen liittyvissä tehtävissä tekoälyyn ei voida luottaa täysin. Tekoälyn generoiman koodin laatu voi olla heikkoa ja virheet näissä tapauksissa voivat olla hengenvaarallisia. [1] Monet yritykset eivät myöskään uskalla luottaa tekoälylle tärkeitä tehtäviä, sillä se voi tuottaa yritykselle tappioita toimimalla väärin. Luvussa 4.1 käsitellään tekoälyn ohjaamiseen liittyviä ongelmia. Luvussa 4.2 käydään tekoälyn luomia tietoturva haasteita. Luvussa 4.3 tekoälyn suoriutumista monimutkaisten tehtävien ratkaisemisesta. Luvussa 4.4 laillisia ja eettisiä ongelmia ja luvussa 4.5 tekoälyn kattavuutta.

4.1 Tekoälyn ohjaaminen

Tekoälyn tuloksien ohjaaminen on merkittävä haaste tekoälyn käytössä. Kehittäjillä on vaikeuksia tietää mitkä syötteet vaikuttavat tekoälyn tulokseen. Tämän seurauksena on vaikea ohjata tekoälyä luomaan haluttu vastaus. [11] Tekoäly voi helposti ymmärtää syötteen merkityksen väärin, siksi joissakin tapauksissa voi olla nopeampaa kirjoittaa koodi itse, kuin yrittää saada tekoälyn avulla haluttu lopputulos. Tekoälyn tuloksien ohjaamista voidaan helpottaa opettelemalla kehotesuunnittelua (*engl. prompt engineering*). Kehotesuunnittelussa syötteitä optimoidaan antaen laajalle kielimallille tarvittavat kontekstit, ohjeet ja esimerkit aiheeseen liittyen saadakseen haluttu vastaus. Tällä tavalla saadaan hyödynnettyä laajoja kielimalleja paremmin. [14]

Tekoäly voi myös alkaa ”hallusinoida”, jolloin tekoäly tuottaa tosiasiallisia epäjohdonmukaisuuksia ja ristiriitaisuuksia [15]. Hallusinointi saa tekoälyn antamaan siis vääriä vastauksia, jotka vaikuttavat oikeilta. Tämän takia kaikki tekoälyn antamat tulokset ovat vaikeita tulkita vääriksi tai oikeiksi. Vastausten tarkistamiseen tarvitaan asiantuntemusta ja tarkkaavaisuutta.

4.2 Tietoturva

Tietoturva on merkittävä haaste laajoille kielimalleille, jonka takia kaikki eivät voi hyödyntää niitä täysin. Laajat kielimallit kuten GPT-4o ovat yleensä yritysten omistamia, joten niihin syötetty data on heidän käsissään. Ohjelmointityössä, jossa käsitellään arkaluontoista tietoa, ei voida käyttää kielimalleja, kuten GPT-4o:ta tietojen vuotamisen riskin takia.

Koodin laatu vaikuttaa myös paljon tietoturvaan. Tutkimuksessa [16] tutkittiin, kuinka tietoturvallista koodia tekoäly on tuottanut erilaisiin projekteihin GitHubissa. Tekoälyn luomaa koodia kerättiin ja nämä analysoitiin tietoturvallisuuden perusteella. Apuna käytettiin staattisia analyysityökaluja, joilla luokiteltiin koodi *Common Weakness Enumeration kategorian* mukaisesti. *Common Weakness Enumeration* kategoria on yhteisön luoma lista yleisistä ohjelmisto heikkouksista, jotka voivat johtaa ohjelmistojen haavoittuvuuteen [17]. Tutkimuksessa 30 % tekoälyn generoimasta koodista sisälsi tietoturva heikkouksia.

Koodin laatu on tietoturvan kannalta ja myös ylipäätänsä ohjelmiston ylläpidon kannalta tärkeää. Suuret heikkoudet ohjelmistossa voivat vaarantaa koko ohjelmiston toiminnan ja aiheuttaa suuria kustannuksia. Tekoälyn kasvava käyttö avaakin uusia ongelmia ratkaistavaksi tietoturva-alalle.

4.3 Vaikeiden tehtävien ratkaiseminen

Tekoälyn suoriutuminen haastavissa ongelmissa voidaan havaita testissä, jossa vertailtiin ihmisten ja tekoälyn luomaa koodia. Tehtävinä käytettiin leetcode.com verkkosivulta löydettyjä tehtäviä. Sivulta löytyy noin 3000 tehtävää luokiteltuna tasoihin helppo, keskitaso ja vaikea. Laajoja kielimalleja käytettiin 7 eri mallia ja ohjelmointikieliksi valittiin C++ (C), Java (J) ja Python (P). Tehtäviä oli yhteensä 6, jotka jaettiin tasaisesti vaikeusasteittain. Kaikkiin tehtäviin yritettiin saada ratkaisu kaikilla kolmella ohjelmointikielellä. Tulokset löytyvät alempana olevasta taulukosta 1. Taulukossa väärät ratkaisut ovat merkitty (-), oikeat ratkaisut (✓) ja käännösvirheet merkitty (ϵ). [18]

Taulukko 1. *Tulokset tekoälyjen ja ihmisen suorituksista Leetcode tehtäviin [18]*

	Easy			Medium			Hard			Total
	#1	#2	#3	#4	#5	#6				
	J | P | C	J | P | C	J | P | C	J | P | C	J | P | C	J | P | C	J | P | C	J | P | C	J | P | C | ∑
ChatGPT (GPT-3.5)	— | — | ✓	— | ✓ | ✓	— | — | —	— | — | —	— | — | —	— | ✓ | —	0 | 2 | 2 | 4
Bing AI (GPT-4.0)	— | ✓ | ✓	✓ | ✓ | ✓	— | ✓ | ✓	— | ϵ | —	— | ϵ | —	— | — | —	1 | 3 | 3 | 7
GH Copilot (GPT-3.0)	✓ | — | ✓	✓ | ✓ | ✓	✓ | — | ✓	✓ | — | ✓	— | — | —	— | — | —	**4 | 1 | 4 | 9**
StarCoder	— | — | —	✓ | — | —	— | — | —	— | — | —	— | — | —	— | — | —	1 | 0 | 0 | 1
CodeWhisperer	— | — | —	— | — | —	— | — | —	— | — | —	ϵ | — | —	— | — | —	0 | 0 | 0 | 0
Code Llama (Llama 2)	— | — | —	✓ | ✓ | ✓	✓ | — | —	— | — | —	— | ϵ | —	— | — | —	2 | 1 | 1 | 4
InstructCodeT5+	— | — | —	— | ✓ | —	— | — | —	— | — | —	— | — | —	— | — | —	0 | 1 | 0 | 1
human	✓ | ✓ | ✓	✓ | ✓ | ✓	✓ | ✓ | ✓	✓ | ✓ | ✓	✓ | ✓ | ✓	✓ | ✓ | ✓	6 | 6 | 6 | 18

Tutkimuksen tuloksien perusteella havaittiin, että tekoäly pystyi ratkaisemaan helppoja ja keskitason tehtäviä, mutta vaikeisiin tehtäviin vain ChatGPT (GPT-3.5) onnistui saamaan yhden ratkaisun Pythonilla. [18]

Stack Overflow'n kyselyssä kuinka hyvin tekoäly pystyy käsittelemään monimutkaisia tehtäviä 28645 vastaajasta 44,8 % sanoi sen olevan huono, 34,4 % vastasi sen olevan hyvä ja loput olivat neutraaleja [10]. Ihmiset ovat myös havainneet, että monimutkaisempien ja vähemmän tunnettujen algoritmien toteuttamisessa tekoäly ei onnistu hyvin [11].

Myös muissa tutkimuksissa tekoälyn on todettu olevan heikko monimutkaisemmissa tehtävissä, tuottaen käyttökelvottomia- tai vääriä vastauksia [19]. Näiden perusteella tekoälyn ratkaisukykyyn vaikuttaa ongelman yleisyys ja kompleksisuus. Tekoälyn ohjelmointikyvyt heikentyvät ongelman haasteellisuuden kasvaessa

Tekoäly pystyy ratkaisemaan helpompia ja tunnettuja ohjelmointitehtäviä, mutta vaikeustasoa nostaessa syntyy haasteita tekoälylle. Kehittäjien mielestä tekoäly ei ole tehokas monimutkaisemmissa ongelmissa. Tekoäly ei vielä kykene ratkaisemaan kaikkia ongelmia. Varsinkin jos ongelma ei ole ennalta tunnettu tai helposti tunnistettava, täytyy kehittäjän itse osata ratkaista ongelma. Pitää myös huomioida, että leetcode.com verkkosivun tehtävät ovat julkisia tehtäviä internetissä, jolloin tekoälyä voidaan olla koulutettu niiden ratkaisuilla jo valmiiksi. Näissä tehtävissä tekoäly oletettavasti suoriutuu paremmin.

4.4 Lailliset ja eettiset

Lailliset ja eettiset ongelmat ovat myös merkittäviä haasteita laajojen kielimallien käytössä. Laajojen kielimallien harjoittamiseen käytetään paljon dataa, joka sisältää julkista sekä tekijänoikeudellista koodia. Tämä herättää kysymyksen sen laillisuudesta ja kuka oikeastaan omistaa tekoälyn luoman koodin. [1] Yrityksiä onkin jo haastettu oikeuteen aiheeseen liittyen. Meta-yrityksen väitetään käyttäneen tekijänoikeudella suojattua materiaalia Llama-kielimallin harjoittamiseen [20]. Yritysten pitääkin olla tarkkana, miten he aikovat kouluttaa ja käyttää tekoälyä, välttääkseen tekijänoikeuden rikkomista. Valtiot tulevat todennäköisesti luomaan erilaisia lakeja laajojen kielimallien käyttämiseen ja niiden kouluttamiseen.

4.5 Kattavuus

Laajat kielimallit tarvitsevat paljon dataa, jotta näitä voidaan kouluttaa kuten luvussa 2 käsiteltiin. Koulutuskelpoisen datan määrä rajoittaa laajojen kielimallien kouluttamista. Tämän takia ne eivät kykene antamaan laadukkaita vastauksia kaikkiin osa-alueisiin ja tulevat olemaan jäljessä uusiin asioihin, joihin koulutusdataa ei ole paljoa saatavilla. Joidenkin osa-alueiden asiantuntemus voi myös olla tekoälyn kykyjen täysin ulkopuolella, sen monimutkaisuuden tai vähäisen koulutusdatan takia. Laajat kielimallit eivät siis kykene kattamaan kaikkea ja ennalta tuntemattomat asiat tuottavat vaikeuksia niille.

5. TULOKSIEN POHDINTA

Tekoälyä mainostetaan todella paljon ja varsinkin yritykset, jotka valmistavat näitä niin haluavat uskoa tekoälyn olevan ratkaiseva tekijä tulevaisuuden ohjelmointiin. Kehittäjät suhtautuivat positiivisesti tekoälyn käyttämiseen ja suurin osa heistä jo hyödyntää tekoälyä työssänsä. Tekoälyn havaittiin nopeuttavan monen kehittäjän työtä monissa eri tutkimuksissa [10], [11], [12]. Kehittäjät, jotka kokivat paljon tuottavuutta ja luottivat tekoälyyn, kokivat myös suurempaa uhkaa heidän työnsä korvaamisesta tekoälyllä [12]. Stack Overflow'n [10] kyselyssä myös huomattiin, että ammattikehittäjät eivät koe tekoälyä yhtä suurena uhkana, kuin ohjelmointia opettelevat. Tämä todennäköisesti johtuu siitä, että he eivät vielä tiedä tarpeeksi mitä kehittäjien työ täsmälleen sisältää. Se selittää myös miksi he uskovat enemmän tekoälyn korvaavan kokonaan heidän työnsä. Aloittelevat kehittäjät keskittyvät myös paljon yksinkertaisempiin ongelmiin, kuin kokeneemmat ammattilaiset. Tekoälyä mainostetaan niillä ongelmilla, joita se pystyy ratkaisemaan. Tästä syystä tekoälyn kyvyt vaikuttavat paremmilta aloitteleville kehittäjille.

Tekoälyn kyvyt generoida toimivia ohjelmia ja tuottaa vastauksia käyttäjille nopeasti vaikuttavat todella suurilta, mutta osa tekoälyn käyttämisestä korvaakin vain koodin hakemisen manuaalisesti internetistä. Ohjelmoinnissa on aina etsitty jatkuvasti vastauksia internetistä samankaltaisesti, kuten nykyään tekoälyllä etsitään. Tekoäly kuitenkin nopeutti monen työtä, sillä internetistä juuri oikean koodin, syntaksin tai ratkaisun löytäminen ohjelmoinnin ongelmaan voi kestää kauankin, kun taas tekoäly pystyy antamaan nopeasti tarkennettua palautetta juuri kehittäjän omaan koodiin. Johdannossa mainitut tekoälyagentit ovat yleistyneet kehitysympäristöissä, joiden avulla kehittäjät pystyvät käyttämään tekoälyä muokkaamaan koko ohjelmointiprojektia. Tekoälyagentit voivat kuitenkin olla vaarallisia, sillä tekoälyagentin tehdessä paljon muutoksia on vaikeampaa havaita tämän tekemät mahdolliset virheet.

Vaikka tekoäly ei pysty ratkaisemaan täysin vaikeimpia tehtäviä, niin joskus näistä on kuitenkin apua tehtävän ratkaisemisessa [18]. Nämä ovat varmasti merkittäviä syitä, miksi tekoäly tehostaa monen kehittäjän ohjelmointityötä. Kuitenkin tekoälyn vastausten analysointi ja oikoluku vie paljon aikaa, jolloin joissakin tapauksissa voisi olla nopeampi työskennellä ilman tekoälyä.

Liiallinen tekoälyn käyttö voi myös vaikuttaa negatiivisesti ohjelmointitaitoihin heikentämällä näitä. Välttääkseen taitojen heikentymistä kehittäjien täytyy harjoittaa ja pitää yllä teknistä osaamistansa [21]. Laajoja kielimalleja ei ole vielä käytetty tarpeeksi pitkää, jotta tiedettäisiin heikentääkö liiallinen käyttö kehittäjien kykyjä ohjelmoida. Kuitenkin aiheesta

keskustellaan paljon ja monet uskovatkin tekoälyn liiallisen käytön olevan riski omien taitojen heikentymiseen. Varsinkin ohjelmointia opetteleville henkilöille ohjelmoinnin perusteet voivat helposti jäädä tietämättä turvautumalla koko ajan tekoälyn käyttämiseen ja tämän vastauksien suoraan kopioimiseen. Tekoäly on myös avannut ohjelmointia paljon uusille ihmisille, jonka seurauksena on myös syntynyt uusi ilmiö englanniksi nimeltä ”vibe coding” tarkoittaen ohjelmien tekemistä tekoälyn avulla kirjoittamatta koodia itse. Ohjelmaa kuvaillaan yksinkertaisilla sanoilla laajoille kielimalleille, yrittäen saada mahdollisimman hyvä tulos. Tämä siirtääkin kehittäjien roolia enemmän koodin kirjoittamisesta ohjaamiseen, testaamiseen ja tekoälyn koodin parantamiseen. [22]

Laajoilla kielimalleilla ongelmia olivat tietoturva koodin heikon laadun takia [16], arkaluonteisten tietojen käsitteleminen muiden yritysten laajoilla kielimalleilla ja laillisuus datan käytöstä laajojen kielimallien kouluttamiseen [1]. On mahdollista kehittää yksityisiä laajoja kielimalleja yrityksille, joiden avulla voidaan luoda juuri oikeanlainen laaja kielimalli. Nämä kuitenkin vaativat suuren määrän dataa ja laskentatehoa, jolla mallia koulutettaisiin. Mallien luominen on kallista, jolloin halvempi ratkaisu voisi olla palkata ammattilainen suorittamaan sama työ. Yksityisten laajojen kielimallien luomisella kuitenkin voitaisiin hyödyntää tekoälyä yrityksissä, joissa käsitellään arkaluonteista tietoa ja voitaisiin, myös mahdollisesti ratkaista tietoturva- ja laillisia ongelmia. Tulevaisuudessa laajoja kielimalleja varmasti kehitetään ja viedään eteenpäin ratkaisten näitä ongelmia.

Kirjallisuuskatsauksen tuloksien perusteella tekoäly ei kuitenkaan tulisi korvaamaan ohjelmointityötä kokonaan vaan toimisi apuna, kuten monet muutkin työkalut, joita käytetään ohjelmoinnissa. Tekoälyn generoiman koodin laatu oli sen verran heikkoa, että siihen tarvitaan ammattilainen valvomaan ja työskentelemään tekoälyn kanssa. Monet tutkimukset myös päätyivät samaan johtopäätökseen, että tekoälyn suoriutuminen haastavista tehtävistä oli heikompaa, turvaten asiantuntemusta vieläkin enemmän. Tekoäly ei siis näytä tulevan korvaamaan töitä lähitulevaisuudessa, joissa tarvitaan ihmisten päätöskykyä [7]. Tällä hetkellä asiantuntemus tulee olemaan tärkeässä roolissa, sillä tekoäly tekee paljon virheitä, eikä pysty ratkaisemaan kaikkia monimutkaisempia ongelmia. Laajojen kielimallien kuitenkin todettiin olevan tehokkaita työkaluja. Monet kuitenkin uskoivat, että tekoälyllä tulevaisuudessa on mahdollista korvata joitakin aloittelijoiden kehittäjien työpaikkoja [12]. Yritykset kuitenkin usein haluavat kouluttaa uusia kehittäjiä erikoistumaan heidän työtehtäviinsä, jonka takia kaikkia aloittelijoiden kehittäjien työpaikkoja ei korvattaisi. Kaikkia ohjelmoinnin työpaikkoja ei siis tultaisi korvaamaan tekoälyllä tällä hetkellä.

6. YHTEENVETO

Tässä työssä käsiteltiin, miten tekoälyä eli tarkemmin laajoja kielimalleja käytetään ohjelmoinnissa ja näiden vaikutuksia ohjelmointityöhön. Laajat kielimallit ovat ohjelmointityökaluina kasvanut suosiossa, ja tuloksien perusteella nämä tulevat vieläkin suositummaksi ohjelmointityössä. Yritykset ja kehittäjät käyttävät näitä aktiivisesti tehostaakseen tuottavuuttansa.

Laajat kielimallit perustuvat koneoppimiseen, joka on tekoälyn osajoukko. Näitä opetetaan suurilla määrillä dataa. Laajat kielimallit eivät ole sama kuin yleinen tekoäly, eikä laajat kielimallit ajattele itse, vaan käyttävät valmiina olevaa tekstidataa eri lähteistä vastauksien muodostamiseen. Tämän avulla ne pystyvät antamaan laadukkaita vastauksia moneen eri tilanteeseen.

Tekoälyä voidaan käyttää monin eri tavoin ohjelmointityössä. Tekoälyn on koettu nopeuttavan työn tehokkuutta, sekä helpottavan työtä. Suosituimmiksi käyttötavoiksi lukeutuivat koodingeneroiminen ja vastauksien hakeminen, joka tuntui auttavan työnkulussa. Tutkielmassa havaittiin myös, että kokeneet ammattikehittäjät tunsivat vähemmän uhkaa tekoälyn korvaavan heidän työnsä, kuin aloittelevat kehittäjät.

Suurimpia heikkouksia tekoälyllä olivat tietoturvaongelmat, laadukkaan koodin kirjoittaminen, monimutkaisempien ongelmien ratkaiseminen, oikeanlaisten tuloksien saaminen tiettyihin ongelmiin ja koodin omistamiseen liittyvät oikeudelliset asiat. Näiden ongelmien takia, vaikka on paljon puhetta kokonaan kehittäjien korvaamisesta tekoälyllä, ei näin ainakaan lähitulevaisuudessa näytä olevan mahdollista.

Ohjelmointityö sisältää paljon asioita, joita tekoäly ei pysty itsenäisesti tekemään. Tekoälyllä onkin vielä suuria haasteita, ratkaistavana ennen kuin se voisi korvata ohjelmoinnin kokonaan. Vaikka tällä hetkellä tämä ei korvaa ohjelmointityötä niin tekoälyä, käytetään jatkuvasti ohjelmointityössä apuna. Näiden käytöstä työkaluina oli positiivisia mielipiteitä, ja kehittäjät käyttivät näitä ominaisuuksia usein työssänsä. Tekoäly kuitenkin kehittyy eteenpäin koko ajan, joten on syytä tehdä jatkotutkimusta, miten tekoäly vaikuttaa ohjelmointiin, kehittäjien taitoihin ja ohjelmistojen laatuun.

LÄHTEET

[1] S. Greengard, ”AI Rewrites Coding”, *Commun. ACM*, vsk. 66, nro 4, ss. 12–14, huhtikuu 2023, doi: 10.1145/3583083

[2] S. J. Russell, *Artificial Intelligence: a modern approach*, Fourth edition., Global edition. teoksessa Pearson series in artificial intelligence. Harlow, United Kingdom: Pearson, 2022.

[3] ”OpenAI’s weekly active users surpass 400 million”, *Reuters*, 20. helmikuuta 2025. Saatavissa: https://www.reuters.com/technology/artificial-intelligence/openais-weekly-active-users-surpass-400-million-2025-02-20/. [Viitattu: 7. maaliskuuta 2025]

[4] A. Palmer, ”Shopify CEO says staffers need to prove jobs can’t be done by AI before asking for more headcount”, *CNBC*, 7. huhtikuuta 2025. Saatavissa: https://www.cnbc.com/2025/04/07/shopify-ceo-prove-ai-cant-do-jobs-before-asking-for-more-headcount.html. [Viitattu: 9. huhtikuuta 2025]

[5] R. Karvinen, ”Tekoälyn hyödyt ja haitat ohjelmistotuotannossa”, 2024, Saatavissa: https://urn.fi/URN:NBN:fi:tuni-202405186009. [Viitattu: 26. maaliskuuta 2025]

[6] V. Terragni, A. Vella, P. Roop, ja K. Blincoe, ”The Future of AI-Driven Software Engineering”, *ACM Trans Softw Eng Methodol*, tammikuu 2025, doi: 10.1145/3715003. Saatavissa: https://dl.acm.org/doi/10.1145/3715003. [Viitattu: 13. helmikuuta 2025]

[7] S. Johnson-Yu ja S. Shah, ”You Don’t Know Jack About AI: And ChatGPT probably doesn’t either”, *Queue*, vsk. 22, nro 5, s. Pages 70:103-Pages 70:116, joulukuu 2024, doi: 10.1145/3703128

[8] ”What are Large Language Models? | NVIDIA Glossary”, *NVIDIA*. Saatavissa: https://www.nvidia.com/en-us/glossary/large-language-models/. [Viitattu: 17. huhtikuuta 2025]

[9] Z. Chu, Q. Ai, Y. Tu, H. Li, ja Y. Liu, ”Automatic Large Language Model Evaluation via Peer Review”, teoksessa *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, Boise ID USA: ACM, lokakuu 2024, ss. 384–393. doi: 10.1145/3627673.3679677. Saatavissa: https://dl.acm.org/doi/10.1145/3627673.3679677. [Viitattu: 7. maaliskuuta 2025]

[10] ”AI | 2024 Stack Overflow Developer Survey”. Saatavissa: https://survey.stackoverflow.co/2024/ai. [Viitattu: 24. tammikuuta 2025]

[11] J. T. Liang, C. Yang, ja B. A. Myers, ”A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges”, New York, NY, USA: ACM, 2024, ss. 1–13. doi: 10.1145/3597503.3608128

[12] M. A. Kuhail, S. S. Mathew, A. Khalil, J. Berengueres, ja S. J. H. Shah, ”“Will I be replaced?” Assessing ChatGPT’s effect on software development and programmer perceptions of AI tools”, *Sci. Comput. Program.*, vsk. 235, s. 103111, heinäkuu 2024, doi: 10.1016/j.scico.2024.103111

[13] A. I. Champa, M. F. Rabbi, C. Nachuma, ja M. F. Zibran, ”ChatGPT in Action: Analyzing Its Use in Software Development”, teoksessa *Proceedings of the 21st International Conference on Mining Software Repositories*, teoksessa MSR ’24. New York, NY, USA: Association for Computing Machinery, heinäkuu 2024, ss. 182–186. doi: 10.1145/3643991.3645077. Saatavissa: https://dl.acm.org/doi/10.1145/3643991.3645077. [Viitattu: 14. helmikuuta 2025]

[14] ”Prompt Engineering for AI Guide”, *Google Cloud*. Saatavissa: https://cloud.google.com/discover/what-is-prompt-engineering. [Viitattu: 9. kesäkuuta 2025]

[15] T. R. McIntosh, T. Liu, T. Susnjak, P. Watters, A. Ng, ja M. N. Halgamuge, ”A Culturally Sensitive Test to Evaluate Nuanced GPT Hallucination”, *IEEE Trans. Artif. Intell.*, vsk. 5, nro 6, ss. 2739–2751, kesäkuu 2024, doi: 10.1109/TAI.2023.3332837

[16] Y. Fu *ym.*, ”Security Weaknesses of Copilot-Generated Code in GitHub Projects: An Empirical Study”, *ACM Trans Softw Eng Methodol*, helmikuu 2025, doi: 10.1145/3716848. Saatavissa: https://dl.acm.org/doi/10.1145/3716848. [Viitattu: 6. maaliskuuta 2025]

[17] ”CWE - About CWE”. Saatavissa: https://cwe.mitre.org/about/index.html. [Viitattu: 9. kesäkuuta 2025]

[18] B. Idrisov ja T. Schlippe, ”Program Code Generation with Generative AIs”, *Algorithms*, vsk. 17, nro 2, s. 62, 2024, doi: 10.3390/a17020062

[19] M. Uandykova, L. Baitenova, G. Mukhamejanova, A. Yeleukulova, ja T. Mirkassimova, ”Java coding using artificial intelligence”, *Front. Comput. Sci.*, vsk. 6, joulukuu 2024, doi: 10.3389/fcomp.2024.1473870. Saatavissa: https://www.frontiersin.org/journals/computer-science/articles/10.3389/fcomp.2024.1473870/full. [Viitattu: 27. maaliskuuta 2025]

[20] K. Knibbs, ”Meta Secretly Trained Its AI on a Notorious Piracy Database, Newly Unredacted Court Docs Reveal”, *Wired*. Saatavissa: https://www.wired.com/story/new-documents-unredacted-meta-copyright-ai-lawsuit/. [Viitattu: 24. maaliskuuta 2025]

[21] ”AI in Software Development | IBM”, 7. lokakuuta 2024. Saatavissa: https://www.ibm.com/think/topics/ai-in-software-development. [Viitattu: 30. huhtikuuta 2025]

[22] ”What is Vibe Coding”, *GeeksforGeeks*, 11:53:00+00:00. Saatavissa: https://www.geeksforgeeks.org/what-is-vibe-coding/. [Viitattu: 28. maaliskuuta 2025]