Tampereen yliopisto

Otso Rouhiainen

GENERATIIVINEN TEKOÄLY OHJELMOINNIN, OPPIMISEN JA OPETUKSEN APUVÄLINEENÄ

Näkökulmana aloitteleva ohjelmoija

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Tammikuu 2024

TIIVISTELMÄ

Otso Rouhiainen: Generatiivinen tekoäly ohjelmoinnin, oppimisen ja opetuksen apuvälineenä – Näkökulmana aloitteleva ohjelmoija
Kandidaattitutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Tammikuu 2024

Generatiivinen tekoäly on tuonut uusia mahdollisuuksia oppimisen ja opetuksen apuvälineenä. Opetusalalla on kuitenkin herännyt huolia tekoälyn väärinkäytöstä ja sen tekemien virheiden vaikutuksista oppimiseen. Osa kouluista on pyrkinyt ratkaisemaan ongelman kieltämällä sen käytön kokonaan. Tekoälyn salliminen antaa kuitenkin mahdollisuuden tarkastella sen hyötyjä ja aiheuttamia haasteita. Tässä työssä tutkitaan generatiivisen tekoälyn potentiaalia opetuskäytössä, oppimisen tukena ja ohjelmoinnin apuvälineenä. Tämän tutkielman tavoitteena on vastata kysymykseen: ”Onko generatiivisesta tekoälystä hyötyä aloittelevalle ohjelmoinnin opiskelijalle, ja mitä vaikutuksia sillä on ohjelmoinnin perusteiden oppimisen kannalta?”

Työ toteutetaan kirjallisuuskatsauksena. Ensin generatiivinen tekoäly esitellään yleisesti, jonka jälkeen sen roolia tutkitaan opetuksen ja ohjelmoinnin oppimisen välineenä. Työssä pohditaan myös tekoälyn käytön rajoituksia, haasteita ja niiden mahdollisia ratkaisuja. Lopuksi arvioidaan tekoälyn tulevaisuutta ja integrointia osana ohjelmoinnin opetusta.

Ohjelmoinnin perusopetuksessa pyritään usein ratkaisemaan lyhyitä ja vähän koodirivejä vaativia tehtäviä. Tutkielmassa selviää, että generatiivinen tekoäly pystyy tuottamaan oikean ratkaisun suurimpaan osaan näistä tehtävistä sekä selittämään, miten ratkaisu tehtiin. Se mahdollistaa myös personoidun oppimiskokemuksen luomisen, vaihtoehtoisten vastausten tarkastelemisen ja auttaa sekä opettajia että opiskelijoita työskentelemään tehokkaammin. Tekoäly tuottaa kuitenkin ajoittain virheellisiä ja jopa täysin vääriä vastauksia. Onnistunut tekoälyn käyttö saattaakin vaatia useamman tarkentavan komennon antamista ja hienosäätöä.

Aloittelevalle ohjelmoinnin opiskelijalle generatiivinen tekoäly voi olla harhaanjohtava apuväline, sillä opiskelija ei välttämättä tunnista tekoälyn tekemiä virheitä. Tekoälyn käyttö voi myös muodostua ongelmaksi, jos sen tuottamiin vastauksiin luotetaan tai nojataan liikaa. Näitä ongelmia voidaan pyrkiä lieventämään tekoälyn roolittamisella ja käytön rajoittamisella. Apuvälineen oikeaoppinen käyttö osana ohjelmointia vaatii hyviä ohjelmointitaitoja, koodin ymmärtämistä ja tietoa niiden tekemistä virheistä.

Avainsanat: generatiivinen tekoäly, ohjelmointi, opetus, oppiminen, integrointi

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin Originality Check –ohjelmalla.

Sisällysluettelo

Termejä

AI – Tekoäly (Artificial Intelligence)

GAI – Generatiivinen tekoäly (Generative Artificial Intelligence)

LLM – Suuri kielimalli (Large Language Model)

GPT – Open AI:n kehittämä kielimallien perhe (Generative Pre-trained Transformer)

IDE – Koodin kirjoitusympäristö (Integrated Development Environment)

ChatGPT – Open AI:n kehittämä vuorovaikutteinen suuri kielimalli (Open-AI, 2023)

GitHub Copilot – Open AI:n, Microsoftin ja GitHubin kehittämä ohjelmointiin tarkoitettu suuri kielimalli (GitHub, 2023)

1 Johdanto

Generatiivinen tekoäly (GAI – Generative Artificial Intelligence) on tuonut uusia mahdollisuuksia oppimisen ja opetuksen apuvälineenä. Se haastaa perinteisen opetuksen menetelmiä tarjoten välitöntä ja personoitua apua. Opetuksessa herää kuitenkin huolia tekoälyn väärinkäytöstä ja sen tekemien virheiden vaikutuksesta oppimiseen. Osa kouluista on pyrkinyt ratkaisemaan ongelman kieltämällä sen käytön kokonaan. Käytön salliminen antaa kuitenkin mahdollisuuden tarkastella uuden teknologian hyötyjä ja aiheuttamia haasteita.

Yleisesti tiedossa on myös se, että generatiivista tekoälyä on mahdollista käyttää ohjelmoinnissa ja että se pystyy tuottamaan automaattisesti perustason koodia. Erilaisia tekoälymalleja on julkaistu paljon, joista osa on suunniteltu suoraan ohjelmointikäyttöön. Tekoälyn nopea kehitys on ajankohtainen ilmiö, eikä vertaisarvioitua tutkimusta aloittelevan ohjelmoijan tukemisesta ole paljoa, ainakaan suomeksi. Aiheesta on enimmäkseen kannanottoja ja arvioita tulevaisuuden suuntauksista. Mallien suuren määrän takia, tutkielmassa pyritään keskittymään erityisesti malleihin *ChatGPT* (OpenAI, 2023) ja *GitHub Copilot* (GitHub, 2023).

Tässä työssä tutkitaan generatiivisen tekoälyn potentiaalia opetuskäytössä, oppimisen tukena ja ohjelmoinnin apuvälineenä. Tavoitteena on keskittyä aloittelevaan ohjelmoinnin opiskelijaan ja vastata tutkimuskysymykseen: ”Onko generatiivisesta tekoälystä hyötyä aloittelevalle ohjelmoinnin opiskelijalle, ja mitä vaikutuksia sillä on ohjelmoinnin perusteiden oppimisen kannalta?”

Tutkimuksessa selvisi, että generatiivinen tekoäly pystyy tuottamaan automaattisesti syntaktisesti oikeanlaista koodia, mutta se tuottaa ajoittain virheitä. Tekoälymalleista voi olla hyötyä aloittelevalle ohjelmoinnin opiskelijalle, mutta ne saattavat myös johtaa harhaan. Kokemus ja ohjelmointitaito auttaa tunnistamaan tekoälyn tuottaman koodin oikeellisuuden ja parantaa tekoälyn käyttömahdollisuuksia. Mallit voivat kuitenkin tukea ohjelmoinnin perusteiden ja ohjelmointikielien oppimista oikein käytettynä.

Aluksi katsauksessa esitellään generatiivista tekoälyä yleisesti sekä sen käyttöä opetuksessa. Tämän jälkeen siirrytään käsittelemään sen ohjelmointikäyttöä. Sen jälkeen arvioidaan vaikutuksia aloittelevaan ohjelmoijaan ja ohjelmoinnin perusteiden oppimiseen. Työssä pohditaan myös tekoälyn käytön rajoituksia, haasteita ja mahdollisia ratkaisuja. Lopuksi tutkittua materiaalia vertaillaan ja pohditaan sekä annetaan johtopäätökset.

Työn tutkimusmenetelmänä on kirjallisuuskatsaus. Aineistoa on etsitty tekemällä hakuja Google Scholarissa, Andorissa, ACM Digital Libraryssa ja Springer Linkissä. Hakusanoina on käytetty yhdistelmiä erilaisista avainsanoista kuten: ’programming, ’education’, ’learning’, ’generative ai’, ’prompt engineering’, ’novice’, ’tutoring’. Lähteitä on suodatettu ottamalla tutkittavaksi mahdollisimman uutta vertaisarvioitua tutkimusta. Aineiston laatua on arvioitu julkaisufoorumin tasoluokituksen avulla (JuFo,

2023). Lähteistä kaksi on korkeimman tason-, kolme johtavan tason- ja loput perustason tieteellistä tutkimusta.

2 Generatiivinen tekoäly ja opetus

Tässä luvussa esitellään generatiivista tekoälyä sekä sen potentiaalia opetuksessa.

2.1 Generatiivinen tekoäly

Generatiivinen tekoäly viittaa tekniikoihin, jotka pystyvät tuottamaan uutta sisältöä, kuten tekstiä, kuvia tai ääntä. Feuerrigel ja muut (2023) väittävät, että niiden tuottama sisältö on näennäisesti uutta, sillä se perustuu harjoitusdataan, jolla mallit on koulutettu. He uskovat myös, että tekoälyn käytön määrä ja mallien kuten Dall-E 2, GPT-4 ja GitHub Copilot levinneisyys, muuttavat parhaillaan tapoja, joilla ihmiset työskentelevät ja kommunikoivat toisilleen.

Suuri osa GAI:sta perustuu *suuriin kielimalleihin* (LMM – Large Language Model). Feuerrigel ja muut (2023) määrittelevät suuret kielimallit tekstidataa mallintavina neuroverkkoina. Nämä ihmisten aivotoimintaa jäljittelevät neuroverkot koulutetaan itseohjautuviksi suurella määrällä dataa ja tietokantoja. Lopulta ne voidaan *hienosäätää*, eli muokata valittujen tietokokonaisuuksien avulla suorittamaan tehtäviä, kuten kysymyksiin vastaamista tai luonnollisen kielen eli ihmisten kieltä muistuttavan kielen tuottamista.

Euchner (2023) arvioi, että keskusteleva tekoäly voi vaikuttaa ihmismäiseltä ja käyttäjää ymmärtävältä teknologialta. Suuret kielimallit eivät kuitenkaan tulkitse ja ymmärrä kuten ihminen, vaan käyttävät matemaattista mallia ennustaakseen virkkeen tai kappaleen seuraavan sanan. Näihin tekoälyn tuottamiin vastauksiin vaikuttaa koulutusdata ja käyttäjän antama *syöte* eli *komento*.

Euchnerin (2023) mukaan suuret kielimallit ovat vain osa generatiivista tekoälyä. Erilaisten mallien suuren määrän vuoksi, tässä tutkielmassa pyritään keskittymään suuriin kielimalleihin. Näistä kielimalleista tarkemmassa tarkastelussa ovat GitHub Copilot sekä ChatGPT. Kumpikin näistä malleista perustuu Open AI:n kehittämään kielimallien perheeseen, josta käytetään lyhennettä GPT (GPT – Generative Pre-trained Transformer). GPT on hienosäädetty tuottamaan vastauksia käyttäjän antamiin luonnollisen kielen syötteisiin. Kumpikin näistä malleista voi keskustella käyttäjän kanssa. Copilot on ainoastaan ohjelmointikäyttöön tarkoitettu, mutta ChatGPT pystyy keskustelemaan myös ohjelmointiin liittymättömistä asioista (GitHub, 2023; Open-AI, 2023.)

GitHub Copilot ja ChatGPT ovat ohjelmointikäyttöön soveltuvia kielimalleja, niissä ei kuitenkaan ole täysin samat ominaisuudet. Copilot on mahdollista yhdistää suoraan *ohjelmointiympäristöihin* (IDE – Integrated Development Environment), jolloin se tuottaa ehdotuksia, jotka on mahdollista hyväksyä käyttäjän vahvistuksella suoraan osaksi ohjelmaa. Copilot pystyy käyttäjän syötteen lisäksi lukemaan myös tiedostoa, jota käyttäjä on parhaillaan muokkaamassa. ChatGPT:lle vastaavat syötteet tulisi antaa erikseen, sillä sitä ei ole integroitu suoraan osaksi IDE:tä. ChatGPT eroaa Copilotista

myös siten, että mallista on useampia julkaistuja versioita. Suurin osa tämän tutkielman aineistosta käsittelee GPT-3 mallia. Sen päivitetympi versio GPT-4 pystyy tuottamaan tekstin ja koodin lisäksi myös kuvia. Erilaisten mallien välisiä eroja on havainnollistettu taulukossa 1. (GitHub, 2023; Open-AI, 2023.)

Taulukko 1. Erilaisia generatiivisen tekoälyn malleja

Malli	Kehittäjä	Vastaanottaa syötteenä	Tuottaa	Tarkoitus
ChatGPT, GPT-3	Open AI	Tekstiä	Tekstiä, koodia ja taulukoita	Keskustelu käyttäjän kanssa, kysymyksiin vastaaminen ja vastausten korjaus
ChatGPT, GPT-4	Open AI	Tekstiä ja kuvia	Tekstiä, koodia, taulukoita ja kuvia	Keskustelu käyttäjän kanssa, kysymyksiin vastaaminen ja vastausten korjaus
GitHub Copilot	Open AI / GitHub / Microsoft	Tekstiä	Tekstiä ja koodia	Ohjelmistokehitys, koodin arviointi, tuotto ja dokumentointi. Käyttö suoraan ohjelmointiympäristössä
DALL-E	Open AI	Tekstiä ja kuvia	Kuvia	Kuvien muokkaus ja tuottaminen

2.2 Potentiaali opetuksessa

Generatiivisen tekoälyn käyttö opetuksessa tulee todennäköisesti muuttamaan ja uudistamaan tapaa, jolla ihmiset oppivat ja opettavat. Sen potentiaalista huolimatta, osa kouluista on kieltänyt sen käytön kokonaan. Tämä ei kuitenkaan välttämättä ratkaise ongelmaa, sillä mallit ovat sekä opettajien että oppilaiden aktiivisessa käytössä. Alasadin ja Baizin (2023) mukaan oppimistavoitteet ja -suunnitelmat tulee toteuttaa reaalimaailman kontekstissa, jossa teknologialla on yhä suurempi rooli ihmisten käyttäytymisen ja päätöksenteon muokkaajana. Heidän mukaansa olisi hyödyllisempää opettaa oppilaita käyttämään tekoälytyökaluja tarkoituksenmukaisesti sen sijaan, että niiden käyttö kiellettäisiin kokonaan. Tämä tapa voisi lisätä opiskelijoiden kykyä tutkia uusien teknologioiden hyötyjä ja haittoja, samalla kun he harjoittavat taitojaan.

Dai ja muut (2023) uskovat, että mallit kuten ChatGPT ovat tehokas keino parantaa koulutuksen laatua ja muuttaa korkea-asteen koulutusta. He pitävät mallia opiskelijalähtöisenä teknologiana, jolla on potentiaalia uudistaa opetusta, oppimista ja arviointikäytäntöjä. Opiskelijalähtöisyyttä vahvistaa tekoälyn käyttöliittymä.

Yksinkertainen dialogimuotoinen malli varmistaa sen, että keskustelun suuntaa hallitsevat ensisijaisesti käyttäjät, eli tässä tapauksessa opiskelijat. Opiskelijat voivat antaa komentoja ja ohjailla keskustelun suuntaa, eivätkä vain vastaanota passiivisesti ilmoituksia tai tehtäviä. Dai ja muut arvioivat, että mallien hyödyllinen käyttö edellyttää kuitenkin opiskelijoilta valmiutta toteuttaa ja hallita kyselyjä, jolloin heillä on paljon vastuuta omasta oppimisestaan. Tässä tutkielmassa kyselyjen hallinnasta käytetään termiä *syötesuunnittelu*, josta on lisää luvussa 5.

Ohjelmoinnin peruskursseja opetetaan usein suurilla luokkakoilla, joissa tavoitellaan sitä, ettei jokaista opiskelijaa joutuisi tukemaan erikseen. Dai ja muut (2023) arvioivat että opiskelijat voisivat saada tekoälyltä välitöntä apua, eikä heidän tarvitsisi odottaa opettajan saatavuutta. Bull ja Kharrufan (2023) uskovat, että opettaja kuormittuisi tekoälyn avulla vähemmän, ja opiskelijat pystyisivät ratkaisemaan tehtäviä jopa ilman opettajan apua. Opiskelijan välittömän tukemisen lisäksi, tekoäly voisi tukea ja nopeuttaa opettajien työtä. Alasadin ja Baizin (2023) mukaan opettajat voivat käyttää tekoälyä arvioinnin ja opetusmateriaalien muokkaamisen apuna. Meyer ja muut (2023) spekuloivat, että myös kooditutoriaalien, kotitehtävien ja ohjelmointiin liittyvien tenttikysymysten tuottaminen olisi mahdollista. Helppojen tehtävien ulkoistaminen tekoälylle saattaa siis auttaa opettajaa keskittymään muihin opetuksen osa-alueisiin.

GAI voi nopeuttaa myös opiskelijoiden työskentelyä ja tarjota yksilöityä apua. Alasadi ja Baiz (2023) arvioivat, että tekoälyn tuottamat mallivastaukset säästävät aikaa ja vaivaa sekä ovat erityisen hyödyllisiä opiskelijoille, joilla on ongelmia kielen, kirjoittamisen tai ilmaisun kanssa. Johnson (2023) uskoo, että tekoäly pystyy antamaan nopeita vastauksia lähes jokaiseen kysymykseen, ja antamaan niistä lisätietoja. Hän arvioi, että tämä ominaisuus auttaa etenkin opiskelijaa, jolla on ongelmia edetä tehtävän kanssa. Dai ja muut (2023) väittävät, että tekoäly pystyy tunnistamaan hyvin oppilaiden erilaisia tarpeita, tavoitteita ja vaikeusalueita. Tämä auttaisi heidän mukaansa puuttumaan yksittäisten opiskelijoiden oppimisvaikeuksiin- ja ongelmiin.

Bull ja Kharrufan (2023) uskovat, että myös ohjelmistokehityksen ala on jälleen uuden paradigman vaihdoksen partaalla. Heidän mukaansa suurten yritysten, kuten Googlen ja Microsoftin siirtyminen kohti GAI:n integrointia osana ohjelmistokehitystä, tulevat näyttämään mallia myös opetukselle. Integrointi tulisi toteuttaa siten, että tekoälyä opeteltaisiin käyttämään kuten muitakin saatavilla olevia apuvälineitä. Ohjelmoijalla on jo nykyiseltään mahdollisuus hyödyntää hakukoneita, tutoriaalivideoita ja kysymys-vastaus-sivustoja. Kaikkiin lähteisiin on suhtauduttava kriittisesti, ja niiden käyttäminen vaatii tietoa. Vaikka teknologiasta voisi olla hyötyä jo nyt, Bull ja Kharrufan arvioivat, että onnistunut integrointi vaatii siirtymisaikaa, sillä GAI on vielä kehittymässä. Heidän mielestään on vaikeata arvioida sen aiheuttamia reaktioita ja muutoksia. He uskovat kuitenkin, että ohjelmoinnin helpottaminen madaltaa kynnystä siirtyä alalle sekä avustaa ohjelmoijaa ratkaisemaan jatkossa korkeamman tason ongelmia.

3 Generatiivinen tekoäly ja ohjelmointi

Tässä luvussa esitellään, miten generatiivinen tekoäly pystyy auttamaan ohjelmoijaa sekä millä tavoin vuorovaikutus sen kanssa tapahtuu.

3.1 Edut ohjelmoinnissa

Vuorovaikutteinen generatiivinen tekoäly pystyy avustamaan ohjelmoijaa monin eri tavoin. Tarjottu apu voi olla joko koodin automaattista tuottamista, koodin toiminnallisuuden auki selittämistä tai tehtävälle sopivan kirjaston tai koodikielen suosittelemista.

Koodin automaattinen tuottaminen – GAI pystyy tuottamaan koodia automaattisesti usealla eri ohjelmointikielellä. Se on hyvä ratkaisemaan yksinkertaisia tehtäviä, lyhyitä koodin pätkiä ja algoritmeja (Bull & Kharrufa, 2023). Koodin laatu on kuitenkin vaihtelevaa ja riippuu paljon käyttäjän antamien ohjeiden kuvailun laadusta sekä halutun koodin vaikeustasosta ja pituudesta. Koodin toimivuus on usein korkeammalla tasolla silloin, kun tekoälyä pyydetään kirjoittamaan pieniä määriä koodia (Meyer ym., 2023). Tuotettu koodi ei kuitenkaan aina täytä kaikkia käyttäjän tai tämän antaman komennon vaatimuksia, mutta on usein korjattavissa kohtuullisen pienillä muutoksilla joko suoraan koodiin tai muuttamalla tekoälylle syötettyjä komentoja (Moradi Dakhel ym., 2023). Koodin automaattinen tuottaminen johtaa parhaimmillaan nopeampaan ja tehokkaampaan ohjelman kehittämiseen. Prather ja muut (2023) väittävät että Copilot tuottaa useimmiten syntaktisesti oikeata koodia, ja että Finnie-Ansleyn ja muiden mukaan [2022] jo vähemmän jalostettu malli nimeltä Codex, pystyy ratkaisemaan ohjelmoinnin perustehtäviä keskiverto ohjelmoinnin opiskelijaa paremmin.

Koodin selittäminen – Vuorovaikutteiselta tekoälyltä on mahdollista kysyä mitä syötteeksi annettu yhden tai useamman lauseen koodi tekee, jolloin se pyrkii selittämään luonnollisella kielellä, mitä kukin muuttuja, komento tai ohjelman eri vaihe tarkoittaa ja toimii. Sitä voi myös pyytää selittämään miksi koodin osa ei toimi tai antamaan ehdotuksia koodin parantamiseksi. Koodin selittäminen voi auttaa oppimaan uutta ohjelmointikieltä, ymmärtämään syntaksia sekä korjaamaan virheellistä koodia. (Meyer ym., 2023.)

3.2 Käyttötapa

Vuorovaikutteisen generatiivisen tekoälyn kanssa on mahdollista käydä keskustelua. GitHub Copilotin tapauksessa keskustelua käydään siten, että ensin ohjelmoija antaa komennon tai kuvailee funktion toimintoja, jonka jälkeen tekoäly pyrkii tuottamaan mahdollisimman kuvaavan ehdotuksen, jonka käyttäjä voi hyväksyä osaksi ohjelmaansa. Ohjelmoijan on mahdollista selata erilaisia ehdotuksia läpi ja korjata antamaansa syötettä.

Barke ja muut (2023) jakavat ohjelmoijien vuorovaikutukset Copilotin kanssa kahteen eri luokkaan: kiihdytys- ja tutkimusmatkailutilaan. Kiihdytystilassa ohjelmoija tietää jo, miten edetä ja työkalu auttaa pääsemään tavoitteeseen nopeammin. Tutkimusmatkailutilassa ohjelmoija ei ole varma, miten edetä, joten hän käyttää työkalua ehdottamaan erilaisia vaihtoehtoja tai etsiäkseen aloituskohtaa ratkaisulle.

Kiihdytystilassa interaktiot Copilotin kanssa ovat nopeita ja eivät vaadi paljoa tarkastelua ohjelmoijalta ehdotuksen hyväksymiseksi. Tutkimusmatkailu sen sijaan vaatii ehdotusten sekä annettujen komentojen tarkempaa tarkastelua ja arviointia, ja on tämän takia usein hitaampaa. Ohjelmoijat käyttävät kumpaakin tapaa tavoitteiden saavuttamiseksi.

Prather ja muut (2023) tutkivat Copilotin ja aloittelevien ohjelmoinnin opiskelijoiden välisiä vuorovaikutuksia. Tutkimus toteutettiin haastattelemalla opiskelijoita. He tunnistivat opiskelijoiden vuorovaikutuksessa kiihdytys- ja tutkimusmatkailutilan lisäksi myös paimentamiseen ja ajelehtimiseen viittaavaa toimintaa. Paimentamisella tarkoitetaan sitä, että opiskelija käyttää suurimman osan ajasta saadakseen tekoälyn tuottamaan koodia, eikä keskity tuottamaan ratkaisua itse. Keskittyminen pelkästään ratkaisun automaattiseen tuottamiseen ja siihen mitä seuraavaksi pitäisi tehdä, saattaa johtaa aloittelijoita harhaan, ja on mahdollisesti sen merkki, että tekoälyn apua käytetään liikaa. Ajelehtimisella taas tarkoitetaan sitä, että opiskelijat hyväksyvät tekoälyn tuottamat ehdotukset, mutta päätyvät myöhemmin poistamaan koodin, jotta voisivat aloittaa uudestaan. Prather ja muut pohtivat, että tämä saattaa johtua siitä, että osa opiskelijoista suhtautuu epäilevästi tekoälyn antamiin vastauksiin. Haastatteluiden perusteella vaikuttaa kuitenkin siltä, että suurella osalla opiskelijoilla on korkea luotto siihen, että tekoäly tuottaa suurimman osan ajasta hyödyllistä ja oikeata koodia. Tämä saattaa kuitenkin olla ongelmallista, sillä tekoäly tuottaa ajoittain virheitä.

4 Generatiivisen tekoälyn vaikutus ohjelmoinnin oppimiseen

Tässä luvussa esitellään generatiivisen tekoälyn käyttämisen vaikutusta aloittelevaan ohjelmoinnin opiskelijaan sekä sen kykyä tukea ohjelmoinnin perusteiden oppimista. Aloittelijoiden väleillä on eroja sekä apuvälineiden käytön että oppimisen suhteen. Tekoälyllä on perusteiden oppimista tukevia vaikutuksia, mutta se voi olla myös harhaanjohtava apuväline.

4.1 Aloitteleva ohjelmoija

Robins ja muut (2003) arvioivat, että aloittelevien ohjelmoijien merkittävimmät vaikeudet liittyvät ongelmanratkaisuun, tekemisen suunnitteluun ja näiden yhteen tuomiseen osana kokonaisen ohjelman laatimista. Tämän vuoksi ohjelmointikurssien keskiössä on useimmiten käytännön ohjelmointitehtävien ratkaiseminen.

Bull ja Kharrufan (2023) väittävät, että aloittelevien ohjelmoijien voisi olla parempi keskittyä kehittämään tietojenkäsittelyn peruskäsitteiden tuntemusta ennen kuin he yrittävät käyttää tekoälyä apuna. He perustelevat väitettään sillä, että aloittelevalla opiskelijalla ei välttämättä ole kykyä tunnistaa tekoälyn tekemiä virheitä. Heidän mukaansa ohjelmoinnin perusteiden yhteydessä opetetaan koodin tarkistamista ja toisten tuottaman koodin arvioimista, joka auttaa tulkitsemaan tekoälyn tuottamaa koodia.

Aloittelijoiden väleillä on kuitenkin eroja. Robins ja muut (2003) jakavat aloittelevat ohjelmoijat kahteen eri luokkaan: tehokkaisiin ja tehottomiin aloittelijoihin. Tehokkaat aloittelijat oppivat ohjelmoimaan vähäisellä avun määrällä ilman erityisiä vaikeuksia. Tehottomat aloittelijat ovat niitä, jotka eivät opi tai oppivat kohtuuttomien vaikeuksien

sekä henkilökohtaisen avustuksen jälkeen. Heidän mukaansa ohjelmoinnin perusopetusta tulisi ohjata sellaiseksi, että kaikista ohjelmoinnin opiskelijoista pyrittäisiin tekemään tehokkaita aloittelijoita. Tutkielmassa todettiinkin jo aikaisemmin, että GAI pystyy vastaamaan opiskelijoiden kysymyksiin ilman viivettä ja vähentämään opettajien kuormitusta. Tekoäly voisi siis mahdollisesti tuoda myös tehottomille aloittelijoille apua henkilökohtaisella avustuksella.

Prather ja muut (2023) arvioivat, että aloittelevien ja kokeneiden ohjelmoijien tarpeet tekoälylle ovat erilaiset. Kokenut ohjelmoija hakee tehokkuutta, nopeutta, koodin optimointia ja erilaisten toteutustapojen tarkastelemista. Aloittelija taas pyrkii etsimään koodin syntaksin ymmärtämistä. Heidän havainnoissaan erikoista on se, että vaikka monet aloittelijoista luottavat tekoälyn tekevän pääasiassa oikeita ratkaisuja, osa suhtautui siihen varautuneesti, eikä välttämättä hyväksynyt tuotettuja ehdotuksia. Aloittelijoilla oli myös yleisesti vaikeuksia ymmärtää ja käyttää Copilotia. He olivat kuitenkin pääsääntöisesti optimistia sen suhteen, että voisivat hyödyntää tekoälyä jatkossa.

4.2 Vaikutus ohjelmoinnin perusteiden oppimiseen

Tutkielman luvun 2.2 yhteydessä käsiteltiin GAI:n potentiaalia opetuksessa, jossa todettiin, että vuorovaikutteinen tekoäly voi tukea opiskelijoiden oppimista mahdollistaen yksilöityjä vastauksia sekä työskentelyä nopeuttavia etuja. Näiden etujen lisäksi oppilaita on mahdollista tukea myös suoraan ohjelmoinnin apuna. Prather ja muut (2023) väittävät, että GitHub Copilot pystyy tukemaan aloittelevaa ohjelmoijaa. He huomasivat, että oppilaat käyttivät tekoälyä suoraan ongelmanratkaisun apuna, suhtautuen tukeen pääasiassa myönteisesti. Myös se, että oppilaat käyttivät apuvälinettä erityisesti antamaan ideoilleen palautetta, saattaa auttaa tukemaan ohjelmoinnin perusteiden oppimista. Copilot mahdollistaa myös oppimisen kokeilun ja virheen korjaamisen kautta, joka on Pratherin ja muiden mielestä yksi tärkeimpiä tekijöitä ohjelmoinnin perusteiden oppimisessa. He arvioivat, että opiskelijat voivat kokeilla apuvälineen avulla erilaisia vaihtoehtoja ympäristössä, jossa virheet ovat helposti tunnistettavissa ja korjattavissa.

Tekoäly voi myös haastaa perinteistä ohjelmoinnin opetusta. Myös Yilmaz ja Karaoglan Yilmaz (2023) näkevät mahdollisuuksia tukea opiskelijaa GAI:n avulla. He pyrkivät selvittämään tutkimuksessaan, miten ChatGPT:n käyttö vaikuttaa opiskelijoiden ohjelmoinnin tehokkuuteen, motivaatioon ja laskennalliseen ajatteluun. Opiskelijat jaettiin kahteen ryhmään, jossa toisessa käytettiin generatiivista tekoälyä apuna ja toisessa ryhmässä toteutettiin vain perinteisen ohjelmoinnin opetuksen keinoja. Kokeilun ja opiskelijoiden haastattelun seurauksena, tekoälyn käytön todettiin lisäävän merkittävästi oppilaiden laskennallisen ajattelun taitoja. Tekoälyä käyttävät opiskelijat käyttivät vähemmän aikaa koodin kirjoittamiseen, mutta enemmän aikaa omaperäisten kysymysten esittelemiseen, ongelmanratkaisuun, luovaan-, algoritmiseen- ja kriittiseen ajatteluun. Tekoälyn käyttö lisäsi myös opiskelijoiden motivaatiota sekä uskoa siihen, että he saavat tehtävät tehtyä. Tosin, vaikeampien tehtävien kohdalla, tekoälyn käyttö ei lisännyt motivaatiota merkittävästi.

5 Generatiivisen tekoälyn ongelmat

Tässä luvussa esitellään generatiivisen tekoälyn käyttöön liittyviä haasteita ja rajoituksia. Ongelmia aiheuttavat tekoälyn tekemät virheet, liiallinen työkaluun nojaaminen sekä eettiset haasteet. Osaan näistä on kuitenkin mahdollisia ratkaisuja.

5.1 Haasteet

Virheelliset vastaukset – Huijausyrityksiä suurempana ongelmana opetuskäytölle, Meyer ja muut (2023) näkevät suurten kielimallien tavan vastata itsevarmasti täysin vääriä vastauksia, kun ne eivät osaa vastata käyttäjän kysymykseen. Johnson (2023) arvioi virheiden johtuvan kielimallien tavasta yhdistellä eri aiheisiin liittyviä materiaaleja. Mallit voivat tuottaa oikein formatoitua tekstiä väärillä tai olemattomilla lähteillä ja niiden antamat vastaukset voivat olla myös liian pitkiä, epäjohdonmukaisia tai täysin vääriä. Meyer ja muut arvioivat, että ChatGPT on erityisen huono rajatapauksien tulkitsemisessa ja sen vuoksi voi tuottaa koodia, joka rikkoo ohjelman toiminnallisuuden kokonaan. He väittävät, ettei malli myöskään anna aina parasta mahdollista koodiratkaisua ensimmäiseksi, vaan saattaa vaatia useamman kierroksen suorittamista hyvän vastauksen saamiseksi. Tämän vuoksi, he ohjeistavat opettajia kertomaan opiskelijoille, että suuret kielimallit eivät ole niin viisaita, miltä ne vaikuttavat ja että ne tekevät ajoittain virheitä. Heidän mukaansa suuriin kielimalleihin tulisi suhtautua kuten muihinkin internetistä löytyviin tietoihin, eli lähdekritiikkiä käyttäen ja faktoja tarkistaen.

Liiallinen nojaaminen – Ohjelmoinnin apuna käytetään hakukoneita, kysymysvastaus sivustoja sekä älykästä koodintäydennystä. Bullin ja Kharrufanin (2023) mukaan näiden käyttäminen osana ohjelmointia auttaa ohjelmoijia tunnistamaan ja löytämään erilaisia ratkaisuja ongelmiin nopeammin ja vähentää tarvetta muistaa metodeja ja syntaksia ulkoa. Heidän mukaansa GAI:n käyttäminen ei eroa juurikaan näiden työkalujen käytöstä, mutta niiden liiallinen käyttö voi vaikuttaa negatiivisesti oppijoiden lähdekriittisyyteen ja kykyyn ratkaista ongelmia. Tekoälyn liialliselle käyttämiselle voisi kuitenkin olla ratkaisuja. He arvioivat, että käyttöä voitaisiin rajoittaa säätämällä ohjelmointiympäristöön tietty raja avunpyyntöjen määrälle. Opiskelijat eivät siis voisi käyttää tekoälyn apua loputtomiin, vaan joutuisivat valitsemaan, milloin apua hyödyntävät.

Plagiointi – Tekoälyn tuottamat vastaukset voivat olla suoraan internetistä kopioituja, ja siten niiden suora käyttäminen osana tekstiä, olisi plagiointia. Meyerin ja muiden (2023) mukaan tuotettujen vastausten käyttö ei kuitenkaan ole aina plagiointia, sillä suuret kielimallit tuottavat ainoastaan käyttäjän syötteen perusteella. He ehdottavat, että koodin omistaa komentojen antaja eli käyttäjä. Väitettä perustellaan sillä, että tekoäly yhdistelee useampia lähteitä ja useamman tehdyn kierroksen tuloksena tuotettu vastaus alkaa muistuttamaan omaa tekstiä. Asia ei kuitenkaan ole niin yksinkertainen, sillä koodi voi olla lisensoitua. Prather ja muut (2023) mainitsevat, ettei aina ole selvää mistä lähteestä koodi on tuotettu, joten ohjelmoija saattaisi käyttää ohjelmassaan lisensoitua koodia tietämättään siitä. Tämän vuoksi, he arvioivat, että on tärkeätä informoida opiskelijoita myös siitä, miten mallit koulutetaan, jotta he ymmärtäisivät käytön eettiset ja oikeudelliset vaikutukset.

Tietojen luovuttaminen – Prather ja muut (2023) arvioivat, että eettisiä ongelmia voisi aiheutua myös siitä, että tekoälyn käyttö edellyttää tyypillisesti tietojen luovuttamista mallien kehittäjälle. Jos tekoäly olisi koulussa yleisesti käytössä, niin opiskelija saattaisi tuntea itsensä painostetuksi luovuttamaan tietonsa.

Vilppi – Edellisten haasteiden lisäksi, GAI:n käyttö aiheuttaa myös haasteita mahdollisen vilpin suhteen. Johnson (2023) epäilee, että opiskelijat saattavat käyttää tekoälyä tekemään tehtävät puolestaan tai kertomaan tehtäviin suorat vastaukset. Bullin ja Kharrufanin (2023) mukaan tämä on johtanut siihen, että opetusalalla on pyritty tunnistamaan huijausyrityksiä sekä tekemään tehtävänannoista haastavampia. Heidän mukaansa keskittyminen väärinkäyttöjen kitkemiseen jättää tekoälyn tuoman potentiaalin tarkastelematta ja käyttämättä hyödyksi opetuksessa. Meyer ja muut (2023) uskovat, että olisi tärkeämpää keskittyä siihen, miten tekoälyä käytettiin tehtävien tekemiseen, sen sijaan että sen hyödyntämisestä tehtäisiin ongelma. Jos oppilas tuottaisi sisältöä syötteitä miettien, faktoja tarkistaen, sekä tekstiä muokaten, tulisi tekoälyn käyttöä arvioida positiivisessa valossa. Vilpin määrää voitaisiin vähentää myös vaatimalla oppilaita: kirjoittamaan raportti tekoälyn kanssa käydystä keskustelusta, osoittamaan oman ja tekoälyn tuottaman vastauksen väliset erot tai esittämällä tekoälyn tekemiä virheitä ja sitä, miten nämä virheet ratkaistiin tehtävässä.

Datan vinoumat – Johnson (2023) väittää että tekoälyn tuottamat vastaukset voivat olla puolueellisia. Hänen mukaansa datan vinoumat voivat vähentää työkalujen luotettavuutta ja lisätä haitallisia stereotypioita.

5.2 Mahdollisia ratkaisuja

Roolitus – Johnsonin (2023) mukaan yksi ratkaisuista virheelliseen käsitykseen GAI:n kyvyistä voisi olla se, että se näytettäisiin erilaisina keskusteluavatareina. Näillä hahmoilla voisi olla erilaisia rooleja ja tehtäviä, ja ne voisivat toimia simuloituina asiakkaina, työtovereina sekä valmentajina. Hahmo voisi siis muistuttaa enemmän ihmistä, jolloin käyttäjä saattaisi huomata virheet helpommin. Johnson uskoo, että myös se olettamus, että GAI on apuväline, eikä pelkkää faktaa tuottava kone, auttaa pienentämään sen käyttämisen riskejä.

Hienosäätö – Bull ja Kharrufa (2023) arvioivat, että GAI-työkaluja olisi mahdollista hienosäätää osaksi ohjelmointiympäristöjä, siten että niistä olisi hyötyä opiskelijoille pedagogisesta näkökulmasta. Heidän mukaansa monet koodin laatuun ja virheellisiin opetuskäytäntöihin liittyvät ongelmat voitaisiin ratkaista, jos oppilaitokset muokkaisivat malleja tehtävien tekemiseen sopivaksi.

Syötesuunnittelu – GAI:n hyödyllisyys ja käyttömahdollisuudet koodin kirjoittajana riippuvat käyttäjän antamien syötteiden tasosta sekä halutun koodin pituudesta (Meyer ym., 2023). Johnsonin (2023) mukaan generatiivisen tekoälyn käytössä on tärkeätä pyrkiä välttämään ja pienentämään sen heikkouksia. Näitä heikkouksia voidaan vähentää hyvällä syötesuunnittelulla (engl. ’prompt engineering’). Syötesuunnittelulla tarkoitetaan tekoälylle annettujen ohjeiden eli komentojen rakentamista, jotta tekoäly antaisi mahdollisimman hyviä vastauksia. Johnson uskoo syötesuunnittelun olevan

tulevaisuudessa välttämätön taito sekä opettajalle että oppilaalle siinä missä ohjelmointitaidotkin. Hänen mielestään on väistämätöntä, että opiskelijat pyrkivät hyödyntämään generatiivista tekoälyä, joten heitä olisi hyvä opettaa käyttämään sitä tehokkaasti.

6 Yhteenveto ja keskustelu

Tutkielman taustana oli oletus, että generatiivinen tekoäly tuottaa ajoittain virheitä. Tutkimus vahvisti taustaa virheiden tuoton osalta ja selvisi, että tekoälyn oikeaoppiminen käyttö vaatii hyvien syötteiden rakentamista ja suunnittelua, useamman kierroksen suorittamista ja virheiden korjaamista. Tekoäly on parhaimmillaan ohjelmointia nopeuttava tekijä, mutta vaatii ohjelmoijalta kokemusta virheiden tunnistamiseksi. Sitä voidaan käyttää tuottamaan koodia automaattisesti, selittämään koodia ja auttamaan kirjastojen ja koodikielen valinnassa. (Johnson, 2023; Meyer ym., 2023.)

Taustalla oli myös keskustelua tekoälyn käyttämisen kieltämisestä ja siitä, että salliminen mahdollistaa hyötyjen ja haasteiden tarkastelemisen. Kieltoa on kommentoitu useassa lähteessä, ja yleinen uskomus vaikuttaa olevan se, että käyttö pitäisi sallia sekä opetuksessa että opiskelijan käytössä. Tutkimuksen perusteella, vaikuttaa todennäköiseltä, että GAI:ta tullaan integroimaan osaksi opetusta ja ohjelmointia ongelmista huolimatta.

Taustana esiteltiin myös heränneitä huolia GAI:n väärinkäytöstä ja sen vaikutuksista oppimiseen. Tutkimuksessa selvisi, että tekoälyä on mahdollista käyttää väärin, mutta sillä voi olla myös positiivisia vaikutuksia ohjelmoinnin perusteiden oppimisen kannalta. Haasteita aiheuttavat eettiset ongelmat kuten vilppi, plagiarismi, tietojen luovuttaminen ja datan vinoumat. Oppimisen haasteina esiintyi liiallinen työkaluun nojaaminen ja virheiden tunnistamisen vaikeus (Bull & Kharrufa, 2023; Prather ym., 2023). Tekoäly voi kuitenkin lisätä motivaatiota suorittaa tehtäviä, auttaa virheiden korjaamisessa sekä tehostaa opiskelijoiden laskennallista-, algoritmista- ja kriittistä ajattelua (Yilmaz & Karaoglan Yilmaz, 2023).

Kirjallisuuskatsauksen tavoitteena oli selvittää, onko generatiivisesta tekoälystä hyötyä aloittelevalle ohjelmoinnin opiskelijalle ja mitä vaikutuksia sillä on ohjelmoinnin perusteiden oppimisen kannalta. Tutkimuskysymyksen ja aiheen valinnan taustalla on se olettamus, että aloittelevaa ohjelmoijaa ja generatiivisen tekoälyn käytön hyötyjä ei ole ehditty tutkia paljoa käytännössä. Katsauksen perusteella voidaan todeta, että GAI:sta voi olla hyötyä aloittelevalle ohjelmoijalle oikein käytettynä ja se saattaa myös tukea ohjelmoinnin perusteiden oppimista.

Tutkimuksen perusteella selvisi, että aloittelevalla ohjelmoijalla ei ole välttämättä kykyä tunnistaa tekoälyn tuottamia virheitä, jolloin hän saattaisi käyttää virheellistä koodia tai oppia huonoja ohjelmointikäytäntöjä. Osa lähteistä vahvistaa sitä, että aloittelevan ohjelmoijan voisi olla parempi keskittyä tekoälyn käyttämisen sijaan kehittämään ohjelmointitaitojaan yleisesti. (Bull & Kharrufa, 2023; Prather ym., 2023.)

Aloittelijoita ja opiskelijoita on kuitenkin eri tasoisia, ja heillä on erilaisia tapoja käyttää tekoälyä. Pratherin ja muiden mukaan (2023) osa opiskelijoista osaa hyödyntää työkalua suoraan ongelmanratkaisun apuna ja käyttää apuvälinettä antamaan ehdotuksia omille ideoilleen. Osa taas pyrkii saamaan tekoälyn tekemään työn puolestaan, jolloin oppimisen laatu voi kärsiä. GAI:lla on kuitenkin potentiaalia avustaa kaiken tasoisia oppijoita, sillä se mahdollistaa personoitua ja välitöntä tukea, joten sitä voisi siis säätää yksittäiselle opiskelijalle sopivaksi (Dai ym., 2023). Tekoälyn aiheuttamia ongelmia voidaan vähentää myös opettajien tarjoamalla ohjeistuksella ja mallien hienosäätämisellä (Bull & Kharrufa, 2023). Jotta tekoälyä voitaisiin integroida opetuskäyttöön, tarvitsee sekä opettajien että oppilaiden opetella käyttämään työkaluja oikein.

Lähes kaikki kirjallisuuskatsauksessa käytetyt lähteet puhuvat tarpeesta selvittää aiheesta lisää, sekä käytännössä että tutkimustasolla. Aiheesta tulee jatkuvasti uutta tietoa, joten käytännöt voivat muuttua tekoälytyökalujen kehittyessä nopeasti ja tutkittu tieto saattaa siten vanhentua nopeasti. Tiedon vanhenemisen lisäksi pohdintaa aiheuttaa tutkimuksen laatu. Vertaisarvioitua tutkimusta löytyy, mutta julkaisufoorumin (JuFo, 2023) tasoluokituksen perusteella vaikuttaa siltä, ettei korkeimman tason tieteellistä tutkimusta löydy paljoa. Jotta voitaisiin julistaa, että GAI on aloittelevaa ohjelmoijaa hyödyttävä teknologia, tarvitaan lisää laadukasta tietoa, jossa otetaan myös enemmän käytännön kokeiluja osaksi tutkimusta. Tämä työ rajoittuu myös vain koodin ja luonnollisen kielen tuottamiseen, mutta jatkotutkimuksen kohteena voisi olla uudempien mallien mahdollistama äänien ja kuvien tuottaminen osana ohjelmointia ja sen opetusta.

Lähdeluettelo

Alasadi, E. A., & Baiz, C. R. (2023). Generative AI in Education and Research: Opportunities, Concerns, and Solutions. *Journal of Chemical Education*, *100*(8), 2965–2971. https://doi.org/10.1021/acs.jchemed.3c00323

Barke, S., James, M. B., & Polikarpova, N. (2023). Grounded Copilot: How Programmers Interact with Code-Generating Models. *Proceedings of the ACM on Programming Languages*, *7*(OOPSLA1), 85–111. https://doi.org/10.1145/3586030

Bull, C., & Kharrufa, A. (2023). Generative AI Assistants in Software Development Education: A vision for integrating Generative AI into educational practice, not instinctively defending against it. *IEEE Software*, 1–9. https://doi.org/10.1109/MS.2023.3300574

Chat-GPT (2023). Open-AI. https://openai.com/blog/chatgpt. (Haettu 1.12.2023.)

Dai, Y., Liu, A., & Lim, C. P. (2023). Reconceptualizing ChatGPT and generative AI as a student-driven innovation in higher education. *Procedia CIRP*, *119*, 84–90. https://doi.org/10.1016/j.procir.2023.05.002

Euchner, J. (2023). Generative AI. *Research Technology Management*, *66*(3), 71–74. https://doi.org/10.1080/08956308.2023.2188861

Feuerriegel, S., Hartmann, J., Janiesch, C., & Zschech, P. (2023). Generative AI. *Business & Information Systems Engineering*. https://doi.org/10.1007/s12599-023-00834-7

GitHub Copilot (2023). GitHub. https://github.com/features/copilot. (Haettu 1.12.2023.)

Johnson, W. L. (2023). How to Harness Generative AI to Accelerate Human Learning. *International Journal of Artificial Intelligence in Education*. https://doi.org/10.1007/s40593-023-00367-w

JuFo. (2023). Julkaisufoorumi. https://www.tsv.fi/julkaisufoorumi/ (Haettu 24.11.)

Meyer, J. G., Urbanowicz, R. J., Martin, P. C. N., O'Connor, K., Li, R., Peng, P.-C., Bright, T. J., Tatonetti, N., Won, K. J., Gonzalez-Hernandez, G., & Moore, J. H. (2023). ChatGPT and large language models in academia: Opportunities and challenges. *BioData Mining*, *16*(1), 20, s13040-023-00339–9. https://doi.org/10.1186/s13040-023-00339-9

Moradi Dakhel, A., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M. C., & Jiang, Z. M. (2023). GitHub Copilot AI pair programmer: Asset or Liability? *The Journal of Systems and Software*, *203*, 111734-. https://doi.org/10.1016/j.jss.2023.111734

Prather, J., Reeves, B. N., Denny, P., Becker, B. A., Leinonen, J., Luxton-Reilly, A., Powell, G., Finnie-Ansley, J., & Santos, E. A. (2023). "It's Weird That it Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. *ACM Transactions on Computer-Human Interaction*, 3617367. https://doi.org/10.1145/3617367

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, *13*(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Yilmaz, R., & Karaoglan Yilmaz, F. G. (2023). The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Computers and Education. Artificial Intelligence*, *4*, 100147-. https://doi.org/10.1016/j.caeai.2023.100147