

SAVONIA

ammattikorkeakoulu

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

GENERATIIVISET TEKOÄLYTYÖKALUT OHJELMISTOKEHITYKSESSÄ

TEKIJÄ Santtu-Pekka Vartiainen

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä Santtu-Pekka Vartiainen	
Työn nimi Generatiiviset tekoälytyökalut ohjelmistokehityksessä	
Päiväys 14.1.2024	Sivumäärä/Liitteet 41 + 3
Toimeksiantaja / Yhteistyökumppani Ponsse Oyj	
Tiivistelmä <p>Tekoälyn edistys on ollut huimaa viime vuosien aikana ja tekoäly on levinnyt osaksi monenlaisia laitteita ja sovelluksia. Myös ohjelmistokehityksen alalle on ilmestynyt tekoälytyökaluja helpottamaan ja tehostamaan ohjelmistokehittäjien työtä. Tällaiset tekoälytyökalut pystyvät generoimaan koodia määrittelyn perusteella, opastamaan ohjelmistokehittäjää ongelmien kanssa ja tarjoamaan automaattista koodintäydennystä koodia kirjoitettaessa. Opinnäytetyön toimeksiantajalla, Ponsse Oyj:llä, ollaan kiinnostuneita tekoälytyökaluista ja niiden tarjoamista mahdollisuuksista tehostaa ohjelmistokehitysprosesseja. Opinnäytetyössä perehdytään tekoälyn historiaan ja teoriaan, esitellään koodin generoinnin käytetyimmät tekoälytyökalut, tutustutaan tieteellisten tutkimusten tuloksiin kyseisten työkalun hyödyistä ja toteutetaan käytännön kokeilu Github Copilot -työkalulla.</p> <p>Kokeilu toteutettiin toimeksiantajan projektissa valikoidulla testiryhmällä kuukauden mittaisella ajanjaksolla. Testiryhmän jäsenille kerrottiin työkalun ominaisuuksista ja he pääsivät tutustumaan työkalun käyttöön opastusvideoiden avulla. Testiryhmän jäsenet työskentelivät projektissa eri rooleissa: frontend-, backend-, DevOps- sekä testiautomaatiokehittäjinä. Kokeiluajan jälkeen testiryhmän jäsenille lähetettiin kysely, jossa selvitettiin, kuinka käyttäjät kokivat työkalun hyödyllisyyden, mitä työkalussa pidettiin toimivana ja mitä siinä tulisi eniten parantaa.</p> <p>Opinnäytetyön tuloksena voidaan todeta Github Copilot -tekoälytyökalun käytön hyödyttävän ohjelmistokehitystä. Github Copilot nopeuttaa ohjelmointia sekä tieteellisen tutkimuksen, että käytännön kyselytutkimuksen perusteella. Käytännön kyselytutkimuksen perusteella Github Copilotin käytön arvioidaan nopeuttavan ohjelmointia ja parantavan ohjelmistokehittäjän tuottavuutta 20–40 %. Mitä tulee Github Copilot -työkalun generoiman koodin laatuun, niin kyselytutkimus osoittaa, että generoitua koodia voidaan pitää toimivana ja sekä kyselytutkimuksen, että tieteellisen tutkimuksen perusteella voidaan osoittaa, että generoitua koodia voidaan pitää yksinkertaisena ja ymmärrettävänä.</p>	
Avainsanat Tekoäly, Koneoppiminen, Neuroverkot, Generatiivinen tekoäly, Ohjelmistokehitys, Ohjelmointi	

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology	
Author Santtu-Pekka Vartiainen	
Title of Thesis Generative AI Tools in Software Development	
Date 14.1.2024	Pages/Appendices 41 + 3
Client Organisation / Partner Ponsse Oyj	
<p>Abstract</p> <p>Advances in artificial intelligence (AI) have progressed rapidly in recent years, and this technology has spread to a wide range of devices and applications. In the field of software development, AI tools have emerged to simplify and increase efficiency in the work of developers. These tools can generate code based on specifications, guide developers through problems, and provide automatic code completion while coding. The focus of this thesis was on AI tools and their potential to enhance software development processes for a client, Ponsse Oyj. The thesis covered the historical and theoretical background of AI and highlighted the most frequently utilized AI tools for code generation. The study also showcased scientific research findings on the advantages of such tools and included a practical experiment involving Github Copilot.</p> <p>The experiment was conducted within the client's project, with a designated test group for a duration of one month. The test group members were educated about the tool's features and got familiar with it via instructional videos. The members of the test group had varied roles in the project, such as frontend, backend, DevOps, and test automation developers. Following the completion of the trial period, a survey was distributed among the test group participants to collect feedback on the perceived effectiveness of the tool, as well as identifying any areas that were successful and those in need of improvement.</p> <p>As a result, it was concluded that the Github Copilot AI tool can benefit software development by speeding up programming and improving software developer productivity. Scientific research findings and the practical survey supported this claim. Based on the survey it was estimated that the use of Github Copilot can increase programming speed and productivity by 20-40 %. Regarding the quality of the code generated by the Github Copilot tool, the survey showed that the generated code can be considered functional, and both the survey and the research findings showed that the generated code can be considered simple and understandable.</p>	
<p>Keywords</p> <p>Artificial Intelligence, Machine learning, Neural networks, Generative Artificial Intelligence, Software Development, Programming</p>	

SISÄLTÖ

1	JOHDANTO	6
2	TEORIA	8
2.1	Historia	8
2.2	Tekoölyn määrittely.....	10
2.3	Kapea tekoöly ja yleinen tekoöly	11
2.4	Koneoppiminen.....	12
2.4.1	Syväoppiminen	13
2.4.2	Neuroverkot	14
2.5	Luonnollisen kielen käsittely.....	15
2.5.1	Kielimallit	16
2.6	Generatiivinen tekoöly	17
3	GENERATIIVISET TEKOÄLYTYÖKALUT OHJELMISTOKEHITYKSESSÄ	19
3.1	IntelliCode.....	20
3.2	GitHub Copilot	21
3.3	Tabnine.....	22
3.4	Amazon CodeWhisperer	23
4	TIETEELLINEN TUTKIMUS.....	25
4.1	Stack Overflow Labs 2023 Developer Survey	25
4.2	The Impact of AI on Developer Productivity: Evidence from GitHub Copilot - Sida Peng, Eirini Kalliamvakou, Peter Cihon & Mert Demirer	26
4.3	Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models – Priyan Vaithilingam, Tianyi Zhang & Elena Glassman	27
4.4	An Empirical Evaluation of GitHub Copilot’s Code Suggestions - Nhan Nguyen & Sarah Nadi	28
4.5	GitHub Copilot AI pair programmer: Asset or Liability? - Arghavan Moradi Dakhel, Vahid Majdinasab, Amin Nikanjam, Foutse Khomh, Michel C. Desmarais & Zhen Ming Jiang.....	30
4.6	Tulosten yhteenveto.....	30
5	KÄYTÄNNÖN KOKEILU	32
5.1	Kokeilun lähtökohdat ja määrittely	32
5.2	Kokeilujakson odotukset	32
5.3	Kyselytutkimuksen suunnittelu	33
5.4	Tulokset ja johtopäätökset	34
6	POHDINTA.....	37

LÄHTEET	39
LIITE 1: KYSELY GITHUB COPILOT -TYÖKALUN KÄYTTÖKOKEMUKSISTA.....	42

KUVALUETTELO

KUVA 1. Tekoälyn ryhmitellyt määritelmät	10
KUVA 2. Tekoälyn, koneoppimisen ja syväoppimisen suhde (Kelleher, 2020, s. 15)	13
KUVA 3. Neuroverkon rakenne (Stanford University: Nilsson, Nils J., 2009, s. 94)	14
KUVA 4. Koodin generoinnin osuus koko generatiivisen tekoälyn sovellusympäristöstä (Marshall, 2023)	19
KUVA 5. Suosituimmat ohjelmistokehittäjien käyttämät tekoälytyökalut (Stack Overflow, 2023)	20
KUVA 6. IntelliCode-tekoälytyökalun koodintäydennyssuosituksset koodieditorissa (Microsoft, 2023)	20
KUVA 7. Github Copilotin Chat -ominaisuus käytössä koodauksessa (Rodriguez, 2023)	21
KUVA 8. Tabninen koodintäydennys funktion deklaraation perusteella (Tabnine, 2023)	22
KUVA 9. Amazon CodeWhisperer ilmaisee generoidun koodin yhteneväisyydet avoimen lähdekoodin kanssa ja kertoo sen mahdollisesta lisenssistä (Amazon Web Services, 2023).	23
KUVA 10. Tutkimuksen suoritusten suoritusajat (Vaithilingam;Zhang;& Glassman, 2022).....	27
KUVA 11. Tutkimuksen suoritusten lopputulos ja onnistumisaste (Nguyen & Nadi, 2022).....	29
KUVA 12. Kyselyn monivalintakysymysten tulokset	34

1 JOHDANTO

Tekoäly, tietokoneella tuotettua älykkyyttä. Mitä on tämä älykkyyys? Onko olemassa määritelmää sille mitä älykkyyys on? Ainakin olemme onnistuneet luomaan tekoälyä, mutta kuinka älykästä tämä tekoäly on. Tekoäly voittaa ihmisen shakissa ja Go-lautapelissä, mutta nämä edustavat kapeaa alaa älykkyydestä. Me ihmisinä miellämme älykkyyden laajemmaksi ja yleisemmäksi ominaisuudeksi, kuin tietyn taidon hallitsemiseksi. Voiko tekoäly saavuttaa tällaisen yleisen älykkyyden tason ja ajatella itsenäisesti? ChatGPT on tällä hetkellä varmasti paras kandidaatti yleisen tekoälyn saavuttajaksi, mutta ei ole vielä sitä saavuttanut (huom. vielä). ChatGPT:n ja muiden vastaavien tuotteiden läpimurto 2020-luvulla on siivittänyt tekoälykehityksen tekoälyhuumaa uudelle kehityksen aallolle. Tekoäly ei ole kuitenkaan uusi keksintö, vaan sen juuret lepäävät 1955 vuoden New Hampshiressa. Nyt voimme vain arvailla kuinka tekoäly tulee muuttamaan elämäämme ja työtehtäviämme. Jos tekoäly pystyy generoimaan toimivaa koodia, tarvitaanko ohjelmistokehittäjiä enää? Toisaalta mikään ei ole yhtä pysyvää kuin muutos. Kaikesta tästä koostuu tämä opinnäytetyö.

Opinnäytetyön tarkoituksena on perehtyä tekoälyyn ja sen tuomiin mahdollisuuksiin ohjelmistokehityksen alalla. Työn toimeksiantaja haluaa toteuttaa kokeilun, jossa otettaisiin käyttöön tekoälytyökalu osaksi ohjelmistokehitystä. Tätä varten tulisi selvittää mikä työkalu olisi kannattavaa valita kokeiltavaksi ja onko tällaisilla työkaluilla todistetusti vaikutusta tehokkuuteen ja tuottavuuteen. Tämä vaatii perehtymistä tarjolla oleviin työkaluihin ja tieteelliseen tutkimukseen, millaisia tutkimuksia työkaluista on tehty ja millaisia tuloksia niistä on saatu. Kun lupaava työkalu on löytynyt, toteutetaan käytännön tutkimus kyseisellä työkalulla ja koostetaan tuloksista johtopäätökset kokeilulle. Olisi mielenkiintoista myös vertailla käytännön tutkimuksessa esille nousseita havaintoja tieteellisen tutkimuksen havaintoihin, että onko niissä yhteneväisyyttä. Tulosten perusteella toimeksiantaja voi sitten arvioida kannattaako työkalua lähteä ottamaan käyttöön laajemmalti yrityksessä.

Työn tavoitteena olisi löytää lupaavin tekoälytyökalu osaksi ohjelmistokehitystä työtä ja toteuttaa käytännön kokeilu kyseisellä työkalulla. Valitun työkalun tulisi hyödyttää ohjelmistokehitystyötä, esimerkiksi tehostamalla ohjelmointia, ja sen tulisi olla tietoturvallinen sekä luotettava. Tällaisten ohjelmistokehitykseen suunnattujen tekoälytyökalujen käytöstä ja tuloksista olisi hyvä löytää aikaisempaa tieteellistä tutkimusta. Tieteellisten tutkimusten tulokset antaisivat viitteitä siitä mitä voisi odottaa käytännön kokeilun tuloksilta. Löydettyjä tuloksia voitaisiin myös verrata sitten käytännön kokeilussa löydettyihin tuloksiin.

Aluksi opinnäytetyössä tutustutaan tekoälyn historiaan ja käydään läpi aihealueen kannalta tärkeimmät tekoälyyn liittyvät teoriat. Tämän jälkeen työssä esitellään muutama käytetyin ohjelmistokehityksen tekoälytyökalu ja minkälaisia ominaisuuksia työkaluissa on. Koska Github Copilot on markkinoiden käytetyin tekoälytyökalu ja vain kyseisen työkalun käytön tuottavuudesta löytyy tieteellisiä tutkimuksia, työn seuraava osa pitää sisällään tiivistelmät ja kootut tulokset Github Copilotta käsittelevistä tieteellisistä tutkimuksista. Viimeisenä osiona opinnäytetyössä on kyselytutkimuksena toteutettu käytännön kokeilu Github Copilot -työkalun käytöstä ja verrataan kyselytutkimuksen tuloksia aikaisemmin esiteltujen tutkimusten tuloksiin ja tehdään johtopäätökset.

Opinnäytetyön toimeksiantajana toimii Ponsse Oyj. Ponsse on vieremäläinen puunkorjukseen keskittyviä ratkaisuja tuottava yritys. Ponssen palveluihin kuuluvat metsäkoneiden valmistus, huolto ja varaosat sekä digitaaliset palvelut. Ponsse on maailman suurimpia metsäkonevalmistajia ja tavaralajimenetelmää hyödyntävän puunkorjuun edelläkävijä. Nykyaikainen puunkorjuu tuottaa jatkuvasti dataa, jota pystytään digitaalisten palveluiden avulla valjastamaan tuottavuuden ja tehokkuuden parantamiseksi, asiakkaiden hyväksi.

2 TEORIA

2.1 Historia

Ensimmäisenä modernin kaltaisena tietokoneena voidaan pitää ENIAC nimistä ohjelmoitavaa elektronista digitaalietokoneetta, joka kehitettiin vuonna 1945 (The Metaphysics Research Lab, Department of Philosophy, Stanford University, 2006). Kuten yleensä, myös tietokone syntyi sotateollisuuden tuotoksena ja ENIAC kehitettiin laskemaan tykistön ballistisia laskutoimituksia. Tästä kymmenen vuotta eteenpäin ja sana ”tekoäly” lausutaan ensimmäisen kerran Dartmouth Collegessa järjestetyssä ”älykkäiden koneiden” työpajassa (Stanford University: Nilsson, Nils J., 2009, s. 77). Työpajan tarkoitus oli määritellä oppiminen ja muu älykkyyden perusta niin että voidaan saada kone simuloimaan sitä. Tätä noin kymmenen oppineen kuuden viikon pituista aivoriittä pidetään tekoälykehityksen alkupisteenä. Tekoälykehityksen alkuaikoina tekoälyn älykkyyden mittana pidettiin shakkia, sen pelaamiseen tarvittavan loogisen ja strategisen päättelykyvyn takia. Tällöin ajateltiin, että tekoäly tulee voittamaan ihmisen shakissa seuraavan 10 vuoden aikana (Council of Europe, 2023). Tähän saavutukseen kului kuitenkin seuraavat 40 vuotta, mutta lopulta IBM:n kehittämä Deep Blue voitti hallitsevan maailmanmestarin Garry Kasparovin shakissa vuonna 1997.

Tekoälyn saavutettua virstanpylvään voittaessaan maailmanmestarin shakissa, tutkijat uskoivat saman saavutuksen tapahtuvan Go:ssa vasta sukupolven päästä (Alpaydin, 2021). Shakkilauta on 8 x 8 kokoinen ruudukko. Shakissa mahdollisten siirtojen määrä on 10^{123} (käytetään myös lukua 10^{120} lähteestä riippuen), joka tunnetaan myös Shannonin numerona kehittäjänsä Claude Shannonin mukaan (Shannon oli yksi Dartmouthin työpajaan osallistuneista vuonna 1956). Shannon julkaisi vuonna 1949 ensimmäisenä tutkimuksen, jossa esiteltiin, kuinka tietokone voidaan ohjelmoida pelaamaan shakkia. Go laudan koko on 19 x 19 ruudukko, joka ei tarkoita, että monimutkaisuus ja mahdollisten siirtojen määrä shakkiin verrattuna kaksin- tai kolminkertaistuisi, vaan ne ennemminkin räjähtävät. Go:ssa mahdollisten siirtojen määrä on 10^{360} , eli luvun perässä on 240 nollaa enemmän kuin shakissa. Vuonna 2016, vain 19 vuotta shakin maailmanmestarin kukistamisesta, Google DeepMindin AlphaGo tekoälysovellus voitti hallitsevan Go:n maailmanmestarin Lee Sedolin ja on ollut siitä lähtien voittamaton, jonka vuoksi myös Sedol lopetti ammattilaisuransa (Alpaydin, 2021). Mikä mahdollisti sen, että AlphaGo pystyi voittamaan Go:n maailmanmestarin? AlphaGo hyödyntää neuroverkkoalgoritmeja, koneoppimista ja tehokasta tietokoneoppia oppiakseen ja arvioidakseen parasta mahdollista siirtoa. Kaikki kolme ovat perinteisiä osia nykyisten tekoälyä käyttävien ratkaisujen takana. (Koch, 2016)

Tekoälyn historiallisessa kehityksessä puhutaan tekoälyn talvista ajanjaksoina, jolloin kehityksen tielle on ilmennyt este tai tekoälyn kehityksestä on vähennetty resursseja, jolloin kehitys on hidastunut. Tämä on aiheuttanut sen, että kehitys on tapahtunut aalloissa, kun este on pystytty rikkomaan muun teknologisen kehityksen tai keksinnön mahdollistamana. Elämme tällä hetkellä vuonna 2023 tekoälyhuuman aikaa, jonka voidaan nähdä alkaneen vuoden 2020 tienoilla kun generatiivisia tekoälysovelluksia ilmestyi laajan yleisön käytettäväksi lyhyellä aikajänteellä. Näistä erityisesti OpenAI:n ChatGPT herätti kiinnostusta ja siivitti tietä sille mitä kaikkea tekoäly voikaan tulevaisuudessa tehdä. Se mikä laukaisi tekoälyhuuman ja miksi useita tekoälysovelluksia julkaistiin lähes samaan aikaan,

on monen asian summa. Neuroverkkojen suunnittelussa saavutettu läpimurto, tietokoneiden laskentatehon nopea kehitys sekä käytettävissä olevan datan määrän kasvu mahdollistivat tekoälysovellusten julkaisut, mutta kun ensimmäinen tekoälysovellus ilmestyi markkinoille, pelko paitsi jäämisestä sai isot teknologiayritykset julkaisemaan omat tekoälyhankkeensa nopealla aikataululla. (Chow & Perrigo, 2023)

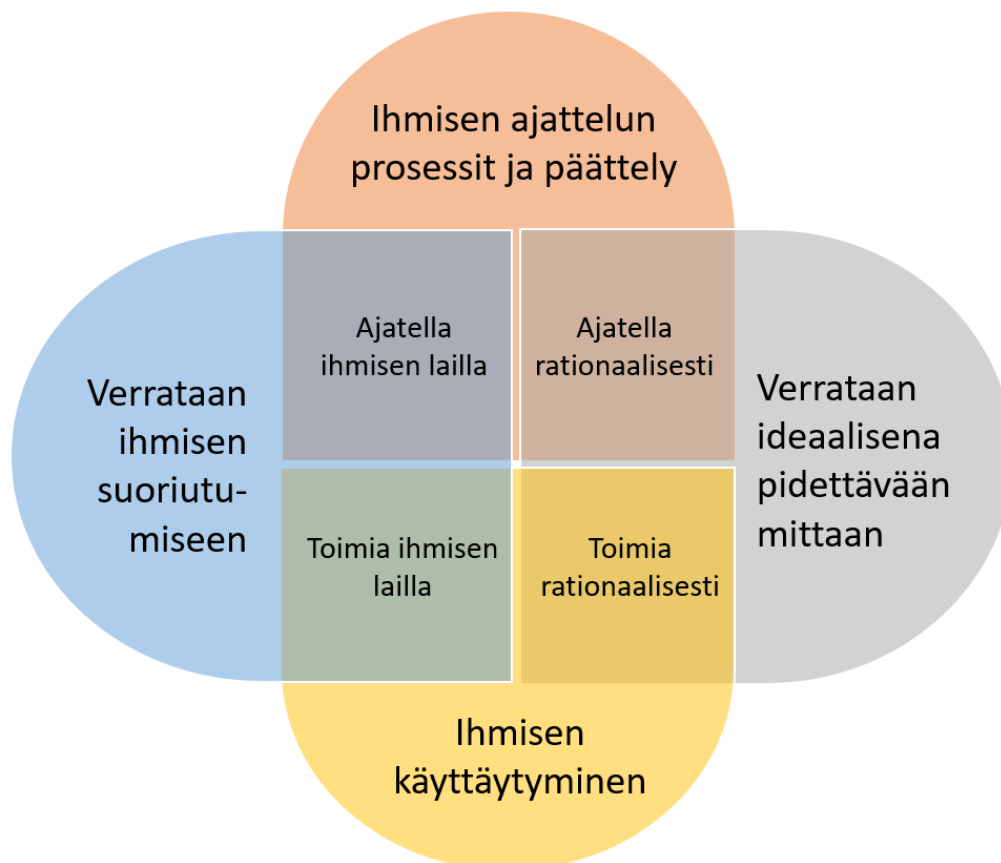
Jo ennen viime vuosien uusia tekoälyn käyttökohteita olemme käyttäneet tekoälyä esimerkiksi sähköpostisuodattimissa, palveluiden suosituksissa ja ennustavassa tekstinsyötössä. Nykyiset tekoälysovellukset pystyvät keskustelemaan ihmisen kanssa, luomaan avainsanojen avulla kuvia tai kirjoittamaan ohjelmistokoodia määritelmän perusteella, mutta tämä on luultavasti vasta alkusoittoa sille kaikelle mihin tekoäly pystyy ja mihin kaikkeen tekoälyä tullaan tulevaisuudessa käyttämään.

Yksi aikamme suurista tietotekniikan alan vaikuttajista, Bill Gates, on todennut:

”Tekoälyn kehittyminen on yhtä keskeistä kuin mikroprosessorin, henkilökohtaisen tietokoneen, internetin ja matkapuhelimen synnyn. Se tulee muuttamaan kuinka ihmiset työskentelevät, oppivat, matkustavat, saavat terveydenhuoltoa ja kommunikoivat toisilleen. Kokonaisia toimialoja tulee uudelleen järjestymään sen ympärille. Yritykset tulevat erottumaan joukosta sen perusteella, kuinka hyvin ne hyödyntävät sitä.” (Gates, 2023)

Elämme jännittäviä aikoja tekoälykehityksen saralla ja voimme jo nähdä tekoälyn tuovan suuren murroksen osaksi elämäämme. Viestinnän professori ja tulevaisuudentutkija Osmo A. Wiion kehittämien tulevaisuuslakien mukaisesti me yliarvioimme uuden teknologian vaikutuksen lähitulevaisuuteen, mutta aliarvioimme sen kaukaiseen tulevaisuuteen, joten suuret tulevaisuuden näkymät tekoälyn osalta eivät tule tapahtumaan hetkessä (Wiio, 2008). Toisaalta ne luultavamminkin tulevat tapahtumaan ja muuttavat elämäämme yli odotuksiemme.

2.2 Tekoälyn määrittely



KUVA 1. Tekoälyn ryhmitellyt määritelmät

Pietikäinen ja Silvén (2019, s. 17) jakavat tekoälyn määritelmät neljään eri luokkaan ryhmiteltyinä Russelin ja Norvigin tekoälyn oppikirjan *Artificial Intelligence* (2010) mukaisesti. Nämä neljä määritelmää ovat: 1) Ajatella ihmisen lailla 2) Ajatella rationaalisesti 3) Toimia ihmisen lailla ja 4) Toimia rationaalisesti. Kyseinen ryhmittely on esitetty kuvassa 1. Tekoälyn määritelmässä tekoälyä verrataan ihmiseen ja rationaalisuuteen. Kuvan yläpuolen tekoälyn määritelmät pohjautuvat ajatteluun ja alapuolen määritelmät toimintaan. Kuvan vasemman puolen määritelmässä tekoälyä verrataan ihmisen suoriutumiseen ja oikean puolen määritelmässä tekoälyä verrataan rationaaliseksi pidettävään mittaan.

Tekoälyn ihmismäisyyden testaamista mielti jo tietokoneiden historian alkuaikoina tietokoneen teoreettisen mallin ja ensimmäisten ohjelmoitavien digitaalisten tietokoneiden kehittäjä matemaatikko Alan Turing vuonna 1950 julkaistussa artikkelissaan *Computing Machinery and Intelligence* (Turing, 1950), jossa hän esitteli kokeen nimeltä matkimispeli (*Imitation Game*). Myöhemmin kokeesta alettiin käyttää nimitystä Turingin testi kehittäjänsä mukaan. Kokeessa ihminen laitetaan keskustelemaan toisen osapuolen kanssa tekstipohjaisesti. Jos ihminen ei pysty erottamaan keskusteleeko hän toisen ihmisen vai tietokoneen kanssa, voidaan tietokoneen tekoälyn sanoa toimivan ihmisen lailla ja olevan älykäs.

Turingin tarkempi määritelmä kokeen läpäisylle oli, että jos keskiverto haastattelijalla ei ole yli 70 % mahdollisuutta erottaa vastaajaa tekoälyksi viiden minuutin keskustelun perusteella, tekoäly on läpäissyt kokeen (The Metaphysics Research Lab, Department of Philosophy, Stanford University,

2021). Useamman tekoälyn väitetään jo kiistellysti läpäisseen Turingin testin ja testiä kohtaan on esitetty kritiikkiä ja parannusehdotuksia. Testin läpäisy vaatii tekoälyltä luonnollisen kielen muodostamista, tietämystä, päättelyä ja oppimista. (Pietikäinen & Silvén, 2019, ss. 17-18)

Tekoälyn neljästä määritelmästä sekä Pietikäinen ja Sivén (2019) että Russell ja Norvig (2010) nostavat rationaalisen toimimisen määritelmän muita tärkeämmäksi. Rationaalisen toimimisen määritelmä tukeutuu rationaalisen agentin käsitteeseen. Agentti voi olla ihminen, robotti, kone, ohjelma yms., joka aistii toimintaympäristöään. Tällainen rationaalinen agentti pystyy toimimaan autonomisesti, mukautumaan muutoksiin ja sitä ohjaa pyrkimys saavuttaa paras mahdollinen tulos sen hetkisen tiedon perusteella. Tällainen toiminta ei pois sulje Turingin testin läpäisemiseksi vaadittavia taitoja (luonnollisen kielen muodostaminen, tietämys, päättely ja oppiminen). Rationaalisella toimimisella voidaan nähdä olevan selviä etuja muihin tekoälyn määritelmiin nähden ja se on käyttökelpoisempi erilaisten sovellusten kehittämiseen, kun ihmisen käyttäytymiseen ja ajatteluun pyrkivät määritelmät. (Pietikäinen & Silvén, 2019, ss. 18-20)

2.3 Kapea tekoäly ja yleinen tekoäly

Artificial Intelligence (AI) kääntyy suomeksi keinotekoinen älykkyys tai keinoäly, mutta vakiintunein käänös on tekoäly. Kun termiä käytetään, niin puhutaan asiasta joka omaa älykkyyttä, mutta joka ei ole ihmisälyä, vaan siihen rinnastettavaa (Kolari & Kallio, 2023, s. 14). Tällainen ihmismäinen älykkyys koneilla on tekoälyä, mutta kaikki koneiden älykkyys ei ole tekoälyä. Tämä johtuu siitä, että suomen kieleen kerkesi aikaisemmin vakiintua käsite älylaite, *smart device*, jota on alettu laajasti käyttää kaikissa tuotteissa, joihin saadaan yhteys tietoverkon kautta. Esimerkkeinä mainittakoon älylamppu, joka saadaan kytkettyä päälle tai vaihdettua väriä puhelimen välityksellä, tai älykoti, jossa robotti-imuri hoitaa imuroinnin automaattisesti, kiukaan saa lämpenemään etäyhteydellä tai valvontakamera ottaa kuvan ovikellon soittajasta. Vaikka nämä ominaisuudet tuntuvat meistä ”älykkäiltä”, ovat ne lopulta hyvin kaukana ihmisälykkyyydestä ja tekoälystä. Siksi koneiden älystä puhuttaessa pitää olla tarkkana puhutaanko älykkäistä laitteista vai tekoälystä.

Tekoälyn määritelmä jaetaan yleisesti kahteen luokkaan; kapeaan tai suppeaan tekoälyyn ja vahvaan tai yleiseen tekoälyyn (*Artificial General Intelligence – AGI*). Kaikki tekoäly mitä olemme tähän mennessä onnistuneet luomaan, on kapeaa tekoälyä. Kapea tekoäly pystyy suorittamaan vain sille määriteltyä tehtävää. Tästä kyseisestä tehtävästä tekoäly pystyy tosin suoriutumaan paremmin kuin ihminen. Tekoäly ei väsy, sen keskittyminen ei herpaannu, se ei erehdy ja se toimii ihmistä monin kerroin nopeammin. Se mitä kapealta tekoälyltä puuttuu, on itsenäisyys. (Järvinen, 2023, ss. 50-51; Kolari & Kallio, 2023, s. 22)

Yleistä tekoälyä ei ole vielä olemassa. Yleisen tekoälyn saavuttamiseksi, tulisi tekoälylle kehittyä laajempi ymmärrys sekä itsenäinen ajattelu ja toiminta. Toisin sanoen, kun tekoäly saavuttaa ihmisen älykkyuden tason, on tekoälystä tullut yleistä tekoälyä. Se mitä näemme scifi-elokuvissa ja jonka uhkakuvista luemme eri medioista, on supertekoälyä (*Artificial Superintelligence – ASI*). Supertekoäly on ihmistä älykkäämpi kaikilla osa-alueilla ja omaa lisäksi oman tahdon ja tietoisuuden. Pelko ja uhkakuvat tekoälykehityksen vaaroista ovat herättäneet huolta laajalti, ja huolen jakavat myös tieteellisen tutkimuksen ja teknologia-alan älyköt, kuten Bill Gates, Stephen Hawking ja Steve Wozniak.

Vuonna 2023 yli tuhat tutkijaa ja liike-elämän edustajaa allekirjoitti avoimen kirjeen tekoälyn kouluttamisen pysäyttämiseksi siitä muodostuneen ihmiskunnan uhkan vuoksi. Myös Oxfordin yliopiston ja ruotsalaisen Global Challenges -säätiön tutkimuksessa tekoälyn arvioitiin olevan todennäköisin syy maailmanlopun aiheuttajaksi. Jopa tunnetuin tekoälytutkimuskeskus OpenAI on julkisesti ilmaissut huolensa yhteisen sääntelyn puuttumisesta, jotta supertekoälyn kehitystä voitaisiin säännellä ja mahdollisiin tulevaisuuden riskeihin pystyttäisiin vaikuttamaan vielä kun on mahdollista (OpenAI: Altman, Sam; Brockman, Greg; Sutskever, Ilya, 2023). (Järvinen, 2023, ss. 305-309)

2.4 Koneoppiminen

Perinteisessä järjestelmässä lopputulos ei voi olla ohjelmoijaansa viisaampi. Ohjelmoija on kirjoittanut järjestelmään kaiken "älykkyyden" eli algoritmit. Algoritmit ovat sarja toimenpiteitä, jotka toteutetaan määritellyn tehtävän. Hyvänä vertauskuvana voidaan pitää suklaakakkureseptiä; seuraamalla ohjeita vaihe vaiheelta sinulla on lopputuloksena suklaakakku. Ohjelmoija varmistaa järjestelmän toimivuuden erilaisilla syötteillä, ja koska hän tietää mitä niiden tulisi olla, koska on kirjoittanut algoritmit, vertaa lopputulosta odotuksiin. Jos lopputulokset vastaavat odotuksia, järjestelmä toimii. Tekoäly on kuitenkin enemmän kuin yksittäisten ohjeiden toteuttamista. (Alpaydin, 2021, ss. 8-9, 30-31)

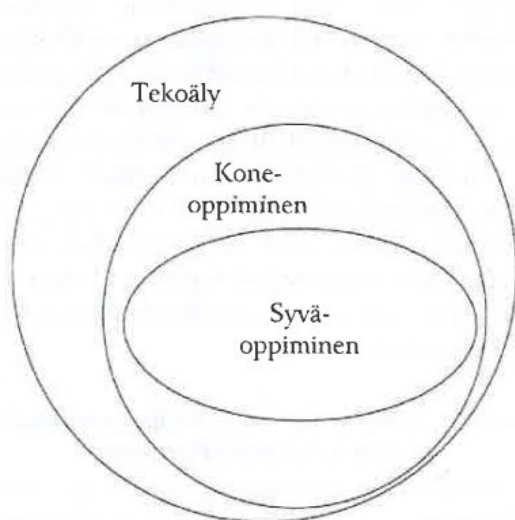
Koneoppimisessa tekoäly voi hoitaa algoritmien kirjoittamisen vaiheen. Ohjelmoija antaa tekoälylle syötteen ja lopputulokset, ja antaa tekoälyn selvittää millä toimenpiteillä tähän päästään. Yleensä ohjelmoijalla on etukäteistietoa aihealueesta ja hän valitsee lupaavimman mallin, jota tekoäly alkaa hienosäätämään, mutta on mahdollista antaa tekoälyn toimia täysin itsenäisesti. Tekoäly suorittaa datalla erilaisia ajoja uudestaan ja uudestaan ja oppii mallintamaan toimenpiteet. Tätä jatketaan senkin jälkeen, kun toimenpiteet ja malli lopputulokseen on löydetty, jotta mallista tulisi yhä tarkempi ja malli oppisi välttämään virheensä ja toistamaan onnistumisensa. Mitä laajempi ja tarkempi data, sitä tarkempi ja käytettävämpi malli saavutetaan. Tällaista oppimisprosessia kutsutaan ohjatuksi koneoppimiseksi. Ohjaamattomassa koneoppimisessa puolestaan tavoitteena on löytää syötteiden säännönmukaisuudet käytössä olevan syöteaineiston avulla tietämättä niiden ulostuloja. Kolmas koneoppimisen osa-alue on vahvistusoppiminen, jossa mallin suoritusta arvioidaan suorituksen jälkeen ja mallin on opittava, kuinka sen tulee toimia ympäristössään saavuttaakseen palkinnon. Koneoppimisen avulla syntyneiden tekoälyjärjestelmien lopputulos ei ole riippuvainen ohjelmoijansa viisaudesta, vaan toivotusti ylittää sen. Tästä voidaan hyvin vetää johtopäätös, että koneoppimisessa lopputuloksen "älykkyyden" määrää datan laadukkuus. (Alpaydin, 2021, ss. 10-12, 56-57, 141; Kelleher, 2020, s. 34)

Tiedon louhintana tarkemmin tunnetussa menetelmässä pyritään koneoppimisen avulla luomaan ennustavia malleja, joista meillä ei ole entuudestaan tietoa tai ymmärrystä. Jos dataa on kertynyt tai sitä pystytään keräämään riittävästi, niin tätä dataa pystytään käyttämään hyväksi koneoppimisessa ja tuottamaan tällaisia malleja, jotka selittävät sääntöjä ja mallintavat asioita. Näin koneoppimisen kautta luotu ennustava malli, ei välttämättä anna kaiken kattavaa totuutta, mutta sitä hyödyntämällä päästään kuitenkin totuuden jäljille. Tällaista mallia pystytään hyödyntämään esimerkiksi kulu-
tustavaroiden myynnin tai terveydenhuollon syy-yhteyksien ennustamisessa tai mallintamisessa.

Tällä hetkellä emme voi kuin arvailla mihin kaikkeen tiedon louhintaa tulemme tulevaisuudessa käyttämään. Keräämme jatkuvasti yhä enemmän dataa yhä laajemmilta elämän osa-alueilta, joista voimme luoda yhä tarkempia oppivia ja ennustavia malleja analysoitavaksi. (Alpaydin, 2021, ss. 28-29)

”Mikään ei ole pysyvää, paitsi muutos” on toteama, jonka alkuperäiseksi esittäjäksi esitetään filosofi Herakleitos, joka eli noin 500 eaa., mutta toteama pitää edelleen paikkansa nykyaikanakin. Dataa syntyy jatkuvasti lisää, se tarkentuu teknisen kehityksen myötä ja siihen vaikuttavat ympäristön muutokset. Näin myös älykkäiden koneoppimisen mallien tulee pystyä oppimaan jatkuvasti muuttuvassa ympäristössä. Vanhentunutta tai jatkuvasti samaa virhettä toistavaa mallia ei voida kutsua älykkääksi. Juuri tämän, kyvyn oppia, voidaan nähdä olevan tekoälyn tärkeimpiä ominaisuuksia. (Alpaydin, 2021, s. 32)

2.4.1 Syväoppiminen



KUVA 2. Tekoälyn, koneoppimisen ja syväoppimisen suhde (Kelleher, 2020, s. 15)

Se miten AlphaGo onnistui voittamaan parhaimman ihmispelaajan Go:ssa, perustuu täysin koneoppimiseen, ja tarkemmin syväoppimiseen. Aluksi AlphaGo:lle opetettiin 100.000 ihmisten pelaamaa peliä, jonka avulla AlphaGo oppi imitoimaan pelaamista ihmisten tavoin. Tämän jälkeen AlphaGo pelasi itse itseään vastaan monta miljoonaa peliä ja oppi vahvistusoppimisen kautta omista virheistään koko ajan paremmaksi. (Kohs, 2017)

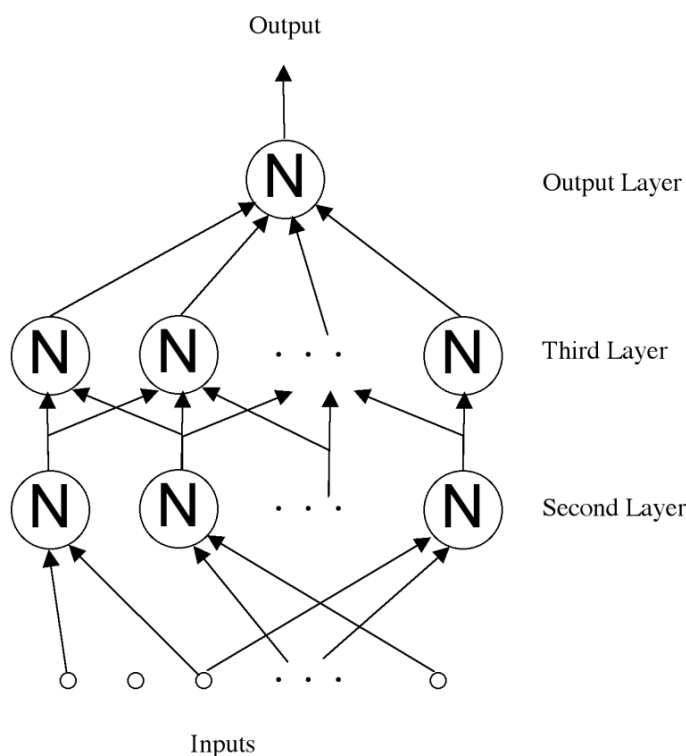
Vahvistusoppimisessa, toimija eli agentti, pyrkii selvittämään mikä toiminto sen hetkessä ympäristön tilassa, tuottaisi suurimman palkinnon. Haastavuutena on se, että palkintoon pääseminen vaatii usein useiden toimintojen sarjaa ja vasta palkinto tuottaa palautteen. Vahvistusoppiva järjestelmä arvottaa mahdolliset välitilat sen mukaan, kuinka hyvin ne johtavat palkintoon. Toiminnoksi valitaan sen hetkessä tilassa se, joka johtaa kasautuvasti suurimpaan palkintoon. (Alpaydin, 2021, ss. 155-160)

John D. Kelleher toteaa kirjassaan *Syväoppiminen – Kuinka tekoäly toimii* (2020, s. 8) että syväoppimisen todella ihmeellinen ominaisuus ei ole sen perustana olevan matematiikan monimutkaisuus,

vaan pikemmin, että se suoriutuu näin monimuotoisesta, jännittävästä ja vaikuttavasta tehtäväjoukosta niin yksinkertaisin laskutoimituksin. Syväoppiminen on koneoppimisen alalaji, ja täten tekoälyalan osa, kuten kuva 2 havainnollistaa. Syväoppiminen perustuu neuroverkkomallien luomiseen, jotka tekevät aineistoon perustuvia päätöksiä. Syväoppimista käytetään yleisemmin tilanteissa, joissa käytössä on paljon monimutkaista dataa.

2.4.2 Neuroverkot

Neuroverkko on laskennallinen malli. Neuroverkon esikuvana on ihmisen aivot, jotka koostuvat valtavasta määrästä neuroneja. Yksinkertaistettuna neuronit kytkeytyvät toisiinsa painoarvotettuja säikeitä pitkin ja neuronit joko välittävät toisista neuroneista tulleita pulsseja eteenpäin tai eivät välitä. Neuronit ovat näin ollen yksinkertaisia tietoa käsitteleviä yksiköitä, mutta yhdessä, neuroverkkona, pystyvät mallintamaan monimutkaisia suhteita. Jokainen neuroni ratkaisee pienen osan kokonaisuongelmasta ja näistä osaratkaisusta kootaan lopullinen ratkaisu. (Kelleher, 2020, ss. 64-68, 76)



KUVA 3. Neuroverkon rakenne (Stanford University: Nilsson, Nils J., 2009, s. 94)

Neuroverkko kuvataan yleensä kerroksina, jotka voivat olla syötekerroksia, piilokerroksia tai ulostulokerroksia. Syötekerrokset vastaanottavat datan, piilokerrokset käsittelevät dataa ja ulostulokerrokset kertovat lopputuloksen. Verkon syvyydenmittana käytetään piilokerrosten ja ulostulokerrosten summaa, ja verkkoa voidaan kutsua syväksi, kun sen syvyys on vähintään kolme. Kuvan 3 mallissa on yksi syötekerros, kaksi piilokerrosta ja yksi ulostulokerros, ja sen syvyys on kolme. Todellisuudessa syväoppivissa malleissa on paljon enemmän kerroksia ja syvyyttä, kuin kuvan mallissa. Neuroverkkojen erilaisuus muodostuukin käytetyistä kerroksista ja neuronien välisistä kytkennöistä. Suosituja neuroverkkotyyppisiä ovat yksisuuntainen, takaisin kytketty ja konvoluutioneuroverkko. (Kelleher, 2020, ss. 64-68, 225)

Neuroverkot perustuvat tietojenkäsittelytieteen tutkija Seppo Linnainmaan kehittämään takaisinviirtauksalgoritmiin (*backpropagation*), joka esiteltiin hänen pro gradussaan *Algoritmin kumulatiivinen*

pyörästysvirhe yksittäisten pyörästysvirheiden Taylor-kehitemänä vuonna 1970. Tällä derivointimenetelmällä pystytään mittaamaan monikerroksisen neuroverkon tuloksen herkkyyttä yksittäisten painojen muutoksille. Linnainmaan kehittämän algoritmin popularisoi myöhemmin toinen tietojenkäsittelytieteen tutkija ja Turing-palkittu Geoffrey Hinton ja siitä tuli neuroverkkojen ja itseoppivien tekoälyjen tärkein osa. Viimeisimmän vuosikymmenen Googlella tekoälytutkijana työskennellyt Hinton päätti lopettaa työnsä Googlella vuonna 2023, jotta voisi puhua suoraan tekoälykehityksen vaaroista (Levy, 2023). (Peltomäki, 2023; MinnaLearn & University of Helsinki, 2022)

2.5 Luonnollisen kielen käsittely

Luonnollisen kielen käsittely (*Natural Language Processing – NLP*) on tekoälyn osa-alue, joka perustuu neuroverkkojen hyödyntämiseen ja jolla tarkoitetaan koneen kykyä ymmärtää kirjoitettua tekstiä ja puhuttua kieltä. Luonnollisen kielen käsittelyn avulla kone saadaan ymmärtämään kieltä niin hyvin, että kone pystyy ymmärtämään sanojen merkityksiä ja sanoihin liittyviä konteksteja ja pystyy tämän jälkeen muodostamaan ihmismäisen järkevän vastauksen sille esitettyihin kysymyksiin. Laajimmalle levinnyt ja yleisesti käytetyin luonnollisen kielen käsittelyn kohde on hakukone. Google-haun alkuperäinen toimintatapa oli käydä läpi verkon sisältöä, indeksoida sisältöä ja laskea sivulle viittaavien linkkien määrää, ja näin arvostaa sisältö tärkeyden mukaan. Tuloksissa hakukone näytti hakusanojen mukaisen sisällön tärkeysjärjestyksessä. Tekoäly saapui Google-hakuun vuoden 2015 algoritmipäivityksen myötä, kun Google ilmoitti käyttävänsä koneoppimista hakukoneessaan ymmärtääkseen paremmin sille esitettyjä kyselyjä. Luonnollisen kielen käsittelyn tekniikkaa Google alkoi käyttämään neljä vuotta myöhemmin vuoden 2019 päivityksessä, joka tunnetaan nimellä BERT. BERT:n avulla hakukone ymmärtää paremmin kyselyjä, tulkitsee paremmin tekstiä ja osaa tunnistaa asioiden väliset suhteet ja kontekstin, jonka ansiosta sanoja analysoidaan yhteydessä hakulauseen kokonaisuuteen eikä yksittäin, kuten aikaisemmin. Aluksi BERT oli käytössä vain englannin kielellä, mutta koska luonnollisen kielen käsittelyn tekniikat soveltuvat hyvin myös kääntämiseen, toimii BERT nykyään yli 70 kielellä. (Kolari & Kallio, 2023, ss. 14, 36-37)

Luonnollisen kielen käsittelyssä neuroverkko muuntaa kielen sanat lukujen muodostamiksi vektoreiksi. Kyseisiä malleja on useita, mutta suosituimpia ovat Tomas Mikolovin Googlella kehittämät word2vec-mallit. Mallien perusajatus on, että samassa viitekehyksessä esiintyvien sanojen merkitykset ovat samankaltaiset, ja viitekehyksenä toimii sanaa ympäröivät sanat. Näin malli osaa tunnistaa sanat Helsinki ja Tukholma semanttisesti samankaltaisiksi, koska molemmat sanat esiintyvät mm. sanojen kaupunki, pääkaupunki ja pohjoismaat yhteydessä. Konekääntäjässä on puolestaan kaksi tällaista neuroverkkoa peräkkäin. Käännettävä lause syötetään ensimmäiseen neuroverkkoon sana sanalta kuitenkin käänteisessä järjestyksessä, koska tämän on todettu johtavan parempiin lopputuloksiin. Kun koko lause on käyty läpi, luo ensimmäinen neuroverkko syötelauseesta vektorin, joka siirretään ensimmäisenä toiseen neuroverkkoon. Vektorin jälkeen toiseen neuroverkkoon syötetään syötelause sana sanalta ja toinen neuroverkko, vektoria hyödyntäen, tuottaa sanoista käännöslauseen. (Kelleher, 2020, ss. 163-164)

Puheohjauksella toimivat virtuaaliset avustajat (kuten Siri, Alexa ja Cortana) käyttävät hyödykseen luonnollisen kielen käsittelyä. Nämä älykkäät avustajat toimivat samoin kuin luonnollisen kielen kä-

sittelymallit kirjoitetun kielen kanssa, mutta niihin on lisätty puheesta tekstiksi -teknologia (puheen-tunnistus) ja tekstistä puheeksi -teknologia. Ensiksi avustaja vastaanottaa äänisignaalin, jonka se muuntaa tekstiksi. Tämän syöte-tekstin perusteella avustaja osaa toteuttaa halutun toiminnon ja tuottaa tekstivastauksen toimintoon. Lopuksi avustaja muuntaa tekstivastauksen puheeksi, eli ”lukee vastauksen ääneen” käyttäjälle. Kaikki tämä tapahtuu niin nopeasti, että käyttäjä voi kuvitella keskustelewansa toisen ihmisen kanssa. Ainoastaan avustajan ”robottimainen” puhe paljastaa kyseessä olevan koneen, sillä tekoäly ei vielä tuota persoonallista keinotekoisia puhetta. Älykäs avustaja voisi oppia käyttäjän taustasta ja etnisyydestä ja mukailla näitä omassa puheessaan luodakseen itsestään samaistuttavan. (Kolari & Kallio, 2023, ss. 74-75; Alpaydin, 2021, s. 88)

2.5.1 Kielimallit

Kielimallit (*Language Models – LM* tai *Large Language Models – LLM*) mahdollistavat tekoälyn kyvyn vastata ja keskustella ihmisen tavoin. Kielimallien ytimenä toimii neuroverkko, jolle on annettu syötteenä laaja tekstiaineisto. Sen toiminta vastaa turboahdettua ennakoivaa tekstinsyöttöä. Kielimalli osaa opetusvaiheessa poimia tekstistä tärkeät avainsanat ja osat, joita se painottaa vastauksissa ja pyrkii tilastollisesti arvaamaan seuraavat sanat ja lauseet. Siksi pidemmät ja tarkemmin kuvaillut kysymykset tuottavat kielimalleilla parempia vastauksia. Mallin tuottama vastaus on lopputuloksena ymmärrettävää ja ihmismäistä, koska sille syötetty lähtöaineisto on ihmisten tuottamaa. Tämä saattaa hämätä käyttäjiä luulemaan, että kielimallilla olisi enemmänkin älyä, vaikka tekstin sisällöstä sillä ei ole mitään ymmärrystä. Siksi jotkin kielimallin vastaukset vaikuttavat oikeilta, mutta niissä on yhdistelty asioita väärin ja lopputuloksesta tulee hassu. Näissä virhetilanteissa tekoälyn sanotaan hallusinoivan. Petteri Järvinen on kiteyttänyt hyvin kielimallien uskottavuuden kirjassaan *Tekoäly ja minä* (2023): ”Kielimalli on kuin maailmankansalaiseksi muuttunut savolainen, jonka kanssa keskusteltaessa vastuu siirtyy kuulijalle”. (Järvinen, 2023, ss. 117-118)

Yksi merkittävä tekijä, joka vaikutti tämänhetkisen tekoälyhuuman alkuun, on Googlen tutkijoiden vuonna 2017 kehittämä Transformer-arkkitehtuuri, joka julkaistiin tutkimuksessa *Attention Is All You Need* (Vaswani, ym.). Kielimalleista, luonnollisen kielen käsittelystä tai näitä hyödyntävien kohteiden alkuperästä puhuttaessa, usein viitataan kyseiseen tutkimukseen ja Transformer-arkkitehtuuriin. Yksi vuoden 2023 Nordic Business Forum liike-elämän konferenssin teemoista oli tekoäly, ja kyseinen tutkimus heijastettiin tapahtumassa kankaalle kaksi kertaa kahden eri esittäjän toimesta, josta voi päätellä kuinka tärkeästä asiasta on kyse.

Aikaisemmat mallit ennen Transformer-mallia, perustuivat monimutkaisiin takaisinkytkettyihin tai konvoluutionaalisiin neuroverkkoihin, joissa kooderi ja dekodeeri yhdistetään huomiomekanismin avulla. Googlen tutkijat löysivät, että mallia yksinkertaistamalla, luopumalla täysin takaisinkytkennän ja konvoluution käytöstä verkkoarkkitehtuurissa ja keskittymällä pelkästään huomiomekanismeihin, saavutetaan tehokkaampia malleja, joiden koulutus on nopeampaa ja tuotos laadukkaampaa. Tämän mahdollistaa Transformer-mallin rinnakkainen datan käsittely, aikaisemman peräkkäisen käsittelyn sijaan. Lisäksi Transformer-arkkitehtuuri mahdollistaa mallien esiopettamisen suurella määrällä dataa ennen kuin malli hienosäädetään erityistä tehtävää varten, jolloin Transformer-arkkitehtuurilla

tuotetut mallit soveltuvat laajalti erilaisiin tehtäviin. Googlen BERT ja OpenAI:n GPT-kielimallit (*Generative Pre-trained Transformer*) perustuvat Transformer-arkkitehtuurin hyödyntämiseen. (Vaswani, ym., 2017; Kelleher, 2020, ss. 211-212)

2.6 Generatiivinen tekoäly

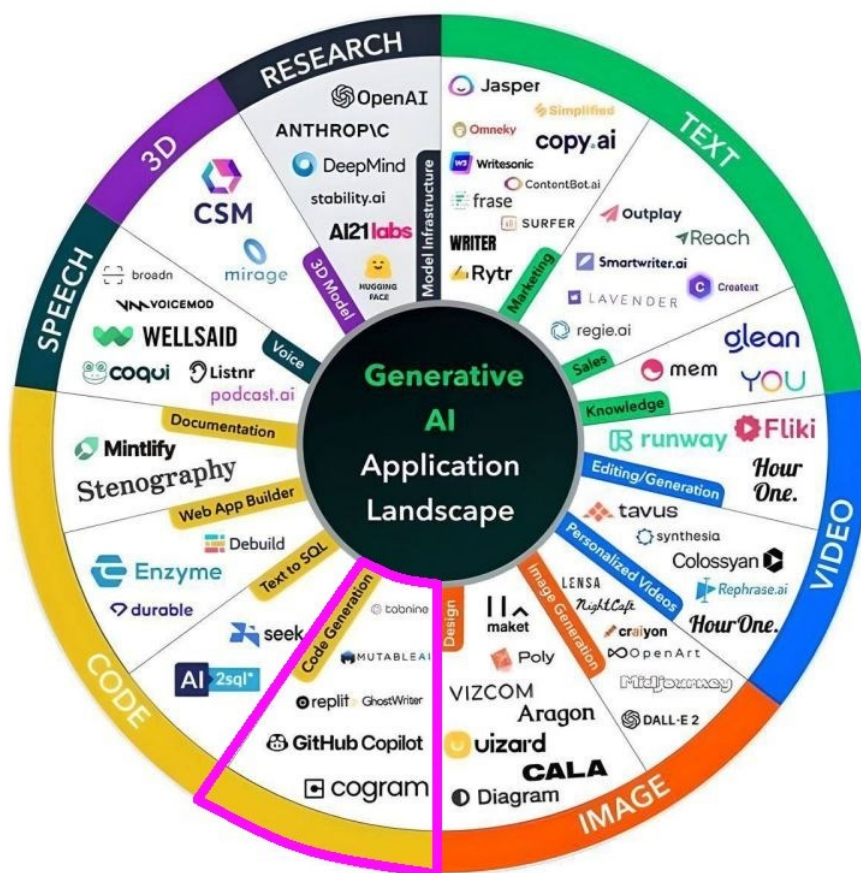
Generatiivisella tekoälyllä (*Generative Artificial Intelligence – GAI*) tarkoitetaan syväoppimismalleja, jotka voivat luoda korkealaatuista tekstiä, kuvia ja muuta sisältöä perustuen siihen dataan, jolla ne on opetettu (Martineau, 2023). Tämän hetken tunnetuin tekoäly-yritys on todennäköisesti OpenAI ja kaikista tunnetuin tekoälytuote heidän generatiivisen tekoälyn palvelunsa ChatGPT, joka on osasyynä tekoälyhuuman alkamiseen, kun yksityishenkilöt ympäri maailmaa pääsivät kokeilemaan tekoälyä ja aihe nousi maailmanlaajuisesti puheenaiheeksi vuonna 2022. OpenAI perustettiin vuonna 2015 tavoitteena edistää avoimen tekoälyn kehitystä, mutta rahoituksen puute ajoi säätiön yhtiöitymiseen ja OpenAI alkoi tarjota kaupallista tekoäly tekniikkaa muille yrityksille. Vuonna 2022 OpenAI julkaisi ChatGPT-palvelun, joka perustuu GPT-kielimalliin, jossa on 175 miljardin parametrin neuroverkko ja jonka kouluttamiseen on käytetty 45 teratavua tekstiä; Wikipedia, verkkokeskusteluja, kirjoja, blogeja, lehtiartikkeleita ja koodiesimerkkejä. ChatGPT on yksinkertaistetusti kaikki alat asian-tunteva chatbotti, joka osaa tuottaa ihmismäistä tekstiä ja keskustella monilla eri kielillä. Koska ChatGPT on esiovetetty tekoälymalli ja sen koulutusaineisto ulottuu vain vuoteen 2021 asti, se ei pysty keskustelemaan tuoreista tapahtumista eikä se hae tietoa internetistä (tämä ominaisuus on nyt myöhemmin lisätty syyskuussa 2023). ChatGPT rikkoi miljoonan rekisteröityneen käyttäjän rajan ennätysellisessä neljässä päivässä julkaisun jälkeen ja kahden kuukauden kuluttua palvelun käyttäjiä oli jo sata miljoonaa. ChatGPT:n suosio perustuu helppoon käyttöliittymään ja saatavuuden mahdollistamiseen kaikille, ominaisuuksiin, joita ei sitä edeltävissä GPT-malleissa ollut, eikä niinkään tekniseen tai tieteelliseen läpimurtoon. (Järvinen, 2023, ss. 120-122; Kolari & Kallio, 2023, ss. 43-44)

Toinen huomattava generatiivisen tekoälyn käyttökohde chattibottien ohella on kuvageneraattorit, joista tunnetuimpia ovat OpenAI:n DALL·E ja Midjourney. Nämä text-to-image-tekoälymallit luovat ja muokkaavat kuvia tekstimuotoisten kuvausten perusteella. Käyttäjä kirjoittaa palvelun syötteeksi tekstiä, jossa kuvailee luotavan kuvan sisältöä, ominaisuuksia ja tyyliä, ja palvelu tuottaa kuvia, joissa tekoäly on yhdistellyt syötteen sisällöt yhteen. Huomattavaa on, että tekoäly ei pelkästään osaa luoda kuvia kuvailuista kohteista, vaan osaa myös matkia taiteilijoita, taidesuuntauksia tai maalaustekniikoita kuvissa. Tekoälyllä tuotettujen kuvien käyttökelpoisuudesta kertoo se, että niillä on alettu korvaamaan ammattilaisten luomia kuvia sanoma- ja aikakauslehtien sekä kirjallisuuden kuvituksissa. OpenAI:n DALL·E:ssa on 12 miljardia parametria ja se on koulutettu teksti- ja kuvaparien avulla. Vaikka tekoälyllä tuotetut kuvat perustuvat opetusmateriaaliin, ei niistä voi tunnistaa alkuperäisiä kuvia, ja luodut kuvat tulkitaan uusiksi. Tekijänoikeuskysymykset ovatkin nousseet tekoälyn ja erityisesti generatiivisen tekoälyn yleistymisen myötä puheenaiheeksi viime vuosina. (Kolari & Kallio, 2023, ss. 53-55; Järvinen, 2023, s. 198)

Tekstin ja kuvien luomisen lisäksi generatiivinen tekoäly yletyy tällä hetkellä myös äänen, videoiden, musiikin ja – tämän opinnäytetyön kannalta tärkeimmän kohteen – koodin tuottamiseen. Ohjelmointi on paljolti loogista päättelyä ja ohjelmointikielet formaaleja, jonka takia ohjelmointi soveltuu hyvin tekoälyn käyttökohteeksi. ChatGPT ei pelkästään pysty kirjoittamaan toimivaa koodia halutulla

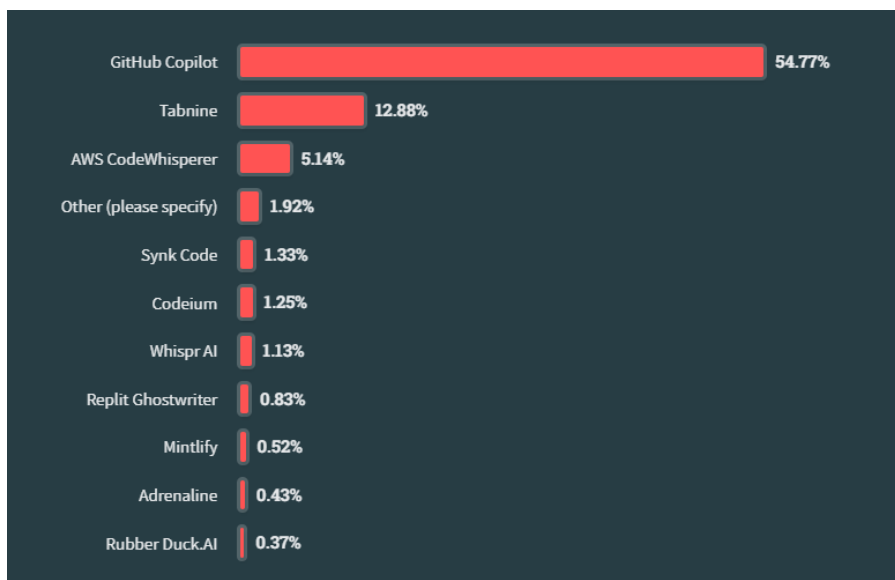
ohjelmointikielellä, vaan se myös kommentoi koodista lohkojen merkitykset, esittää sanallisen kuvauksen ohjelman toiminnallisuudesta ja suorittaa malliajon esimerkin omaisesti. Ohjelmointi on paljon muutakin kuin pelkästään koodin kirjoittamista; se on muun muassa suunnittelua, testausta ja tarpeiden tunnistamista. Siksi tekoälystä ei ole vielä ihmisen korvaajaksi ohjelmistotuotannon alalla, mutta loistavaksi apuriksi siitä on. (Järvinen, 2023, ss. 125-128; Kolari & Kallio, 2023)

3 GENERATIIVISET TEKÖÄLYTYÖKALUT OHJELMISTOKEHITYKSESSÄ



KUVA 4. Koodin generoinnin osuus koko generatiivisen tekoälyn sovellusympäristöstä (Marshall, 2023)

Tässä työssä keskitytään koodia generoiviin ja koodin kirjoittamisessa avustaviin ohjelmistokehittäjän tekoälytyökaluihin (kuva 4). Nämä työkalut on koulutettu massiivisilla koodikannoilla koneoppimisalgoritmeja hyödyntäen. Tekoälypohjaisia koodiavustajia on ollut aiemminkin, mutta ne ovat saavuttaneet aivan uuden tason tekoälyhuuman myötä, kun suurten kielimallien kehitys nytkähti ripeästi eteenpäin. Tekoälytyökalu ohjelmistokehittäjälle on kuin pariohjelmoija, joka on koulutettu avustamaan ohjelmoijaa työssään. Koodiavustaja ymmärtää koodin kontekstin ja analysoi sitä reaaliaikaisesti. Se osaa ennustaa mitä ohjelmoija olisi mahdollisesti seuraavaksi kirjoittamassa ja tarjoaa ehdotuksia ohjelmoijalle. Työkalu ymmärtää myös luonnollista kieltä ja sen kanssa pystyy keskustelemaan, esimerkiksi kysymään mitä kyseinen koodi tekee, tai sitä voi pyytää generoimaan ohjeistuksen mukaista koodia. Tekoälytyökalujen kehittäjät markkinoivat työkalujen vähentävän koodin kirjoittamiseen ja debuggaamiseen käytettyä aikaa, havaitsevan virheitä, automatisoivan tehtäviä, parantavan koodin laatua, edistävän parhaita käytäntöjä ja kiihdyttävän ohjelmoijan oppimista. (Tabnine, 2023)

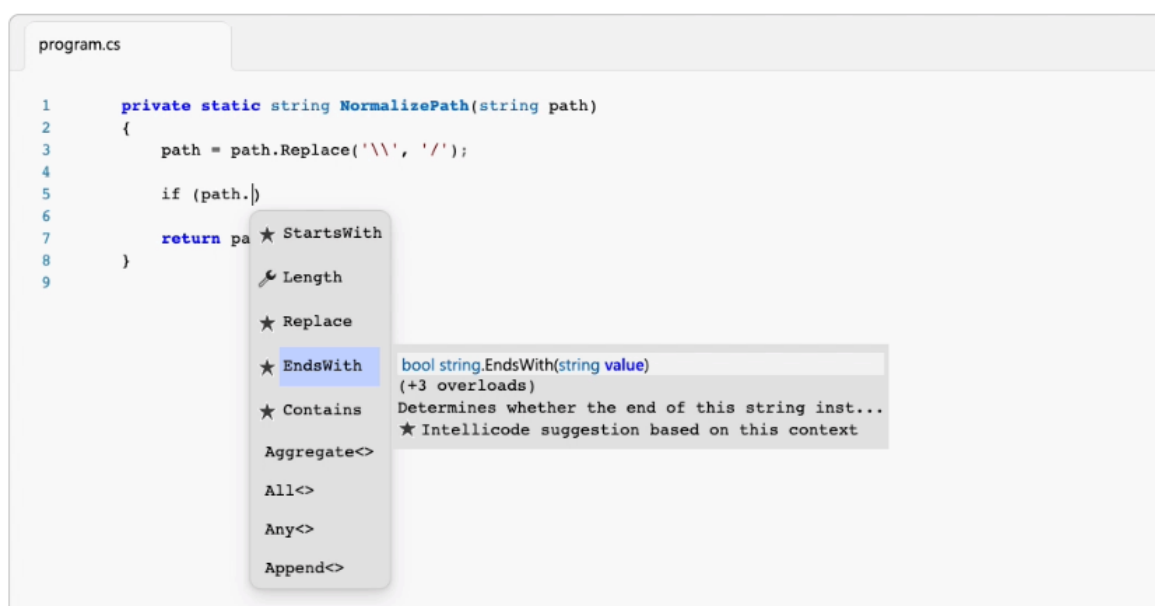


KUVA 5. Suosituimmat ohjelmistokehittäjien käyttämät tekoälytyökalut (Stack Overflow, 2023)

Eriaisia tekoälytyökaluja on ilmestynyt markkinoille runsaasti tämänhetkisen tekoälyhuuman myötä. Myös ohjelmistokehittäjille suunnattuja tekoälytyökaluja on markkinoilla useita, mutta työkalujen käyttö on keskittynyt vahvasti muutamaun tunnettuun työkaluun (kuva 5). Stack Overflow:n vuoden 2023 kehittäjäkyselyn perusteella yli puolet vastanneista yli 40.000 kehittäjästä on käyttänyt säännöllisesti Github Copilottia viimeisen vuoden aikana ja aikoo jatkaa käyttöä tulevan vuoden aikana. Github Copilot on kyselyn perusteella selkeästi suosituin tekoälytyökalu. Toiseksi suosituimpana tekoälytyökalu on Tabnine (13 %) ja kolmanneksi suosituimpana Amazon (*Amazon Web Services - AWS*) CodeWhisperer (5 %). Muiden tekoälytyökalujen käyttö on hyvin marginaalista. (Stack Overflow, 2023)

Seuraavaksi työssä esitellään tarkemmin IntelliCode, Github Copilot, Tabnine ja Amazon CodeWhisperer -tekoälytyökalut.

3.1 IntelliCode

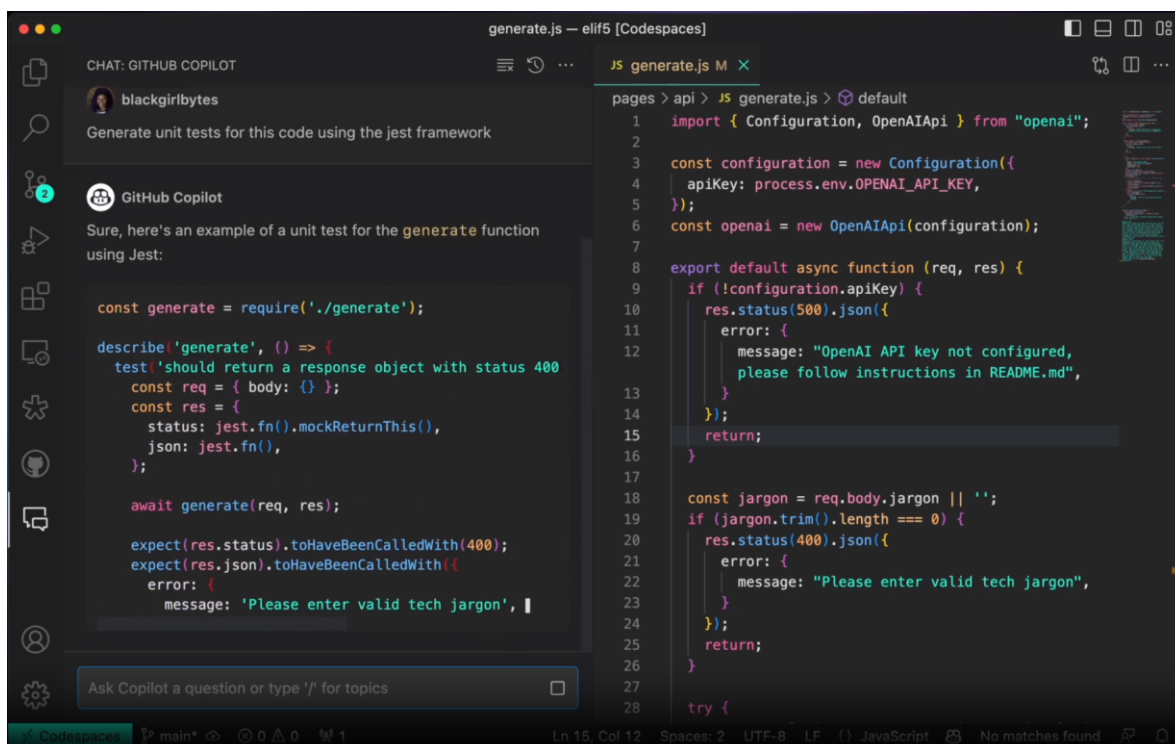


KUVA 6. IntelliCode-tekoälytyökalun koodintäydennysuositukset koodieditorissa (Microsoft, 2023)

Microsoftin kehitysympäristöihin on mahdollista liittää Microsoftin ilmainen IntelliCode-tekoälytyökalu. Uusimmat Visual Studio ja Visual Studio Code kehitysympäristöjen versiot sisältävät automaattisesti IntelliCode-työkalun mukanaan. Kyseisissä kehitysympäristöissä on aikaisemmin ollut käytössä IntelliSense-työkalu, joka on staattinen ja automaattinen koodintäydennystyökalu. IntelliSense muun muassa tarjoaa automaattista sanan täydennystä sitä mukaa kuin käyttäjä kirjoittaa, listaa tyyppin mukaisia metodeja ja tarjoaa aputietoja. IntelliCode puolestaan yhdistää IntelliSenseen tekoälyn, joka tekee siitä dynaamisen. IntelliCoden tekoäly ymmärtää koodin asiayhteyden, tarjoaa koko rivin pituisia täydennyksiä sekä oppii käyttäjän koodikannasta ja mukautuu käyttäjän koodaustyyliin. IntelliCode perustuu koodin käyttöön erikoistuneeseen GPT-C transformer -malliin, jolle on opetettu tuhansia julkisia yli 100 tähden Github koodivarastoja (*code repository*). (Microsoft, 2023; Microsoft, 2023; Ramel, 2022)

3.2 GitHub Copilot

Lokakuussa 2023 uutisoitiin Microsoftin toimitusjohtajan Satya Nadellan kertoneen Github Copilotilla olevan yli miljoona käyttäjää yli 37.000 yrityksessä maailmanlaajuisesti (Tiernan, 2023). Käyttäjämäärässä oli tapahtunut 40 % kasvu edellisestä kvartaalista (Tiernan, 2023). Stack Overflow:n vuoden 2023 kehittäjäkyselyn perusteella (kuva 5) voidaan todeta Github Copilotin olevan ohjelmistokehittäjien suosituin tekoälytyökalu. Microsoft osti Githubin itselleen 7,5 miljardilla dollarilla vuonna 2018 ja on nyt sijoittanut yli 10 miljardia dollaria OpenAI tekoäly-yritykseen. Näiden perusteella Microsoft on päättänyt panostaa vahvasti Github Copilotiin ja tekoälykehitykseen, ja tulee luultavasti pitämään tekoälytyökalunsa markkinoiden suunnannäyttäjää vielä tulevaisuudessakin.



The screenshot shows the GitHub Copilot Chat interface in a code editor. The chat window on the left contains the following text:

```
CHAT: GITHUB COPILOT
blackgirlbytes
Generate unit tests for this code using the jest framework

GitHub Copilot
Sure, here's an example of a unit test for the generate function using Jest:

const generate = require('./generate');

describe('generate', () => {
  test('should return a response object with status 400', () => {
    const req = { body: {} };
    const res = {
      status: jest.fn().mockReturnThis(),
      json: jest.fn(),
    };

    await generate(req, res);

    expect(res.status).toHaveBeenCalled();
    expect(res.json).toHaveBeenCalled();
    expect(res.json).toHaveBeenCalledWith({
      error: {
        message: 'Please enter valid tech jargon',
      },
    });
  });
});
```

The code editor on the right shows the following JavaScript code:

```
generate.js — elif5 [Codespaces]
pages > api > JS generate.js > default
1 import { Configuration, OpenAIApi } from "openai";
2
3 const configuration = new Configuration({
4   apiKey: process.env.OPENAI_API_KEY,
5 });
6 const openai = new OpenAIApi(configuration);
7
8 export default async function (req, res) {
9   if (!configuration.apiKey) {
10    res.status(500).json({
11      error: {
12        message: "OpenAI API key not configured,
13         please follow instructions in README.md",
14      },
15    });
16    return;
17  }
18
19   const jargon = req.body.jargon || '';
20   if (jargon.trim().length === 0) {
21     res.status(400).json({
22       error: {
23         message: "Please enter valid tech jargon",
24       },
25     });
26     return;
27   }
28   try {
```

KUVA 7. Github Copilotin Chat -ominaisuus käytössä koodauksessa (Rodriguez, 2023)

GitHub Copilot eroaa Microsoftin IntelliCodesta siinä, että se käyttää OpenAI:n Codex-tekoälymallia ja ymmärtää luonnollista kieltä. Codex on OpenAI:n ja Githubin yhteistyönä kehittämä tekoälymalli, joka perustuu OpenAI:n GPT-3 kielimalliin (OpenAI, 2021). Codex sisältää 12 miljardia parametria ja

sen kouluttamiseen on käytetty 159 GB koodia 54 miljoonasta Github koodivarastosta (Dakhel, ym., 2022, s. 1). Github Copilottia voidaan pitää paranneltuna ja maksullisena IntelliCoden versiona, niin sanottuna lippulaivatuotteena, jossa on mukana kaikki edistyneimmät ominaisuudet. Github Copilottia markkinoidaan tekoälyllisenä pariohjelmoijana, joka ymmärtää koodin kontekstin, tekee kooditäydennys ehdotuksia reaaliaikaisesti, pystyy generoimaan kokonaisia lohkoja koodia ja sen kanssa pystyy keskustelemaan koodista ja ohjelmointiin liittyvistä kysymyksistä (kuva 7). Github raportoi vuoden 2023 alussa, että nykyisin Github Copilotin avulla kirjoitetaan noin 46 % kaikesta koodista ja se nopeuttaa koodin kirjoittamista noin 55 % (Dohmke, 2023). (Github, 2023)

Toisin kuin IntelliCode, joka toimii täysin paikallisesti käyttäjän tietokoneella, Github Copilot tarvitsee internet-yhteyden tuottaakseen ehdotuksia. Vastaavissa kilpailijoiden työkaluissa on mahdollisuus ajaa palvelua paikallisesti tai yksityisellä palvelimella, mutta Github Copilot ei ainakaan vielä tue tällaista mahdollisuutta. Näin ollen tiedon on kuljettava oman verkon ulkopuolelle, joka altistaa tietoturvariskeille. Lisäksi Github Copilot pidättää oikeuden kehittää palveluaan työkalun käytöstä kerätyllä datalla. Tekoälyn yleistymisen myötä on alettu yleisesti keskustella generatiivisen tekoälyn tekijänoikeuksista: sekä tekoälyn koulutusaineiston tekijänoikeuksista että generatiivisen tekoälyn tuotosten tekijänoikeuksista. Näihin edellä mainittuihin aiheisiin liittyvät ongelmat ja epäselvyydet ovatkin varmasti suurimmat kynnyskysymykset tällaisten tekoälytyökalujen käyttöönotolle ohjelmistokehityksessä. (Github, 2023)

3.3 Tabnine



```

JS weather.js > [🔗] fetchWeatherForCity
1   import 'dotenv/config'
2   import fetch from "node-fetch";
3
4   const API_KEY = process.env.API_KEY;
5
6   const fetchWeatherForCity = async (city) => {
7   const url = 'https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${API_KEY}';
   const response = await fetch(url);
   const data = await response.json();
   return data;
}

```

KUVA 8. Tabninen koodintäydennys funktion deklaraation perusteella (Tabnine, 2023)

Tabnine on tekoälypohjainen ohjelmistokehittäjän työkalu, joka hyödyntää kehittyneitä koneoppimisen malleja. Tabnine vastaa paljolti Github Copilottia, mutta ilman Github tai Microsoft yhteyttä. Kuten Github Copilot, myös Tabnine pystyy liittämään kaikkiin suosituimpiin koodieditoreihin ja Tabnine tukee kaikkia suosituimpia ohjelmointikieliä. Tabnine huomioi koodin asiayhteyden ja oppii sekä mukautuu käyttäjän koodaustyyliin tarjoillessaan koodintäydennystä. Tabnine tarjoaa rivin ja lohkon pituisia koodintäydennyksiä, ja se pystyy generoimaan kokonaisia osuuksia luonnollisen kielen määrittelyn perusteella. Kuten Github Copilotissa, myös Tabnineen on tulossa chat-ominaisuus (tällä hetkellä beetatestauksessa), jonka avulla pystyy keskustelemaan koodista tekoälyn kanssa. (Tabnine, 2023)

Tabninessa on kuitenkin muutama huomattava eroavaisuus verrattaessa Github Copilotiin. Tabnille pystyy opettamaan yksityiset lähdekoodit, jolloin Tabninen täydennyksissä on valmiiksi huomioituna yrityksen/projektin koodauskäytännöt ja -tyylit. Lisäksi Tabninen tekoälyn koulutukseen on käytetty ainoastaan luvan myöntäneitä avoimen lähdekoodin koodivarastoja. Tabninen pystyy asettamaan ajoon yksityiselle palvelimelle, jolloin työkalun käyttäminen ei vaadi koodin lähettämistä organisaation ulkopuolelle, joka takaa täydellisen yksityisyyden ja turvallisuuden. Tämä mahdollistaa myös työkalun käyttämisen ilman internet-yhteyttä. Tabnine erottuu Github Copilotista ottamalla vahvemmin huomioon käyttäjän yksityisyyden ja opetusaineiston tekijänoikeudet, asiat, jotka ovat Github Copilotissa herättäneet negatiivista keskustelua työkalun ympärillä. (Tabnine, 2023)

3.4 Amazon CodeWhisperer

Myös Amazon on lähtenyt kilpailuun mukaan ja tarjoaa ohjelmistokehittäjille omaa tekoälytyökaluaan nimeltä Amazon CodeWhisperer. Amazon CodeWhisperer on julkaistu huhtikuussa 2023 ja Amazon on lähtenyt kilpailemaan hinnalla tarjoamalla yksilöllisensä ilmaiseksi. Alun perin työkalua tarjottiin vain Amazon Web Services (AWS) käyttäjille, mutta on nykyään vapaasti muidenkin käytettävissä. Kuitenkin suurimman hyödyn työkalusta saavat AWS-ekosysteemin parissa toimivat. Kilpailijoita harvemmat koodieditorit tukevat Amazon CodeWhispereriä, mutta työkalu tukee samalla tavalla kaikkia suosituimpia ohjelmointikieliä kuin kilpailijatkin. Amazon CodeWhisperer tarjoaa automaattista koodintäydennystä ja pystyy generoimaan lohkoittain koodia kommenttiin kirjoitetun määritellyn perusteella, mutta siitä puuttuu chat-ominaisuus. (Amazon Web Services, 2023)

The screenshot shows a code editor with the following code:

```

12 # function to iterate over an S3 bucket and send the files to a lambda
13 export const getFiles = async (bucketName, S3Client) => {
14   const params = {
15     Bucket: bucketName,
16   };
17
18   const files = [];
19   let isTruncated = true;
20   let continuationToken;
21
22   while (isTruncated) {
23     const response = await s3Client.listObjectsV2(params);
24     isTruncated = response.IsTruncated;
25     continuationToken = response.ContinuationToken;
26     response.Contents.forEach((file) => files.push(file.Key));
27   }
28   return files;

```

A tooltip above the code says: "Reference code under MIT License."

A "CodeWhisperer reference log" popup is visible, containing the following text:

CodeWhisperer reference log
[3/24/2023, 3:17:34 PM] Accepted recommendation with code `response.Contents.forEach((file) => files.push(file.Key));` provided with reference under MIT from repository `function-templates`. Added to `/demos/reference-tracker.js` (line at 14).

KUVA 9. Amazon CodeWhisperer ilmaisee generoidun koodin yhteneväisyydet avoimen lähdekoodin kanssa ja kertoo sen mahdollisesta lisenssistä (Amazon Web Services, 2023).

Amazon CodeWhispererissä on huomioitu erityisesti turvallisuus. Amazon CodeWhisperer suodattaa pois ehdotukset, joita voidaan pitää puolueellisina tai epäreiluinä, ilmoittaa mikäli ehdotus vastaa avoimen lähdekoodin ratkaisua ja tarjoaa linkin lähdekoodiin ja kertoo sen lisenssin, sekä sisältää

turvallisuus skannauksen, joka havaitsee mahdolliset haavoittuvuudet ja suosittaa parhaimpia turvallisuuskäytäntöjä. Amazon CodeWhispereriä pystyy myös mukauttamaan opettamalla sille omia koodivarastoja, ohjelmointirajapintoja ja kirjastoja, joiden perusteella työkalu muotoilee ehdotukset vastaamaan opetusmateriaalia. (Amazon Web Services, 2023)

4 TIETEELLINEN TUTKIMUS

Tieteellinen tutkimus tekoälytyökalujen hyödyistä ohjelmistokehityksessä keskittyy tällä hetkellä vahvasti Github Copilotin ympärille, eikä muista työkaluista juuri löydy julkaistuja tutkimuksia. Tablimesta löytyy mainintoja tutkimusten yhteyksissä, mutta työkalun käytön vaikutuksista ei löydy tutkittua tietoa. Amazon CodeWhisperer on puolestaan niin tuore julkaisu markkinoilla, että tutkimuksissa, joissa se mainitaan, sen todetaan olevan vielä kehityksen alla ja julkisen julkaisun tapahtuvan tulevaisuudessa. Kuten ohjelmistokehittäjien keskuudessa, myös tutkijoiden keskuudessa Github Copilot tuntuu olevan selkeästi suosituin ohjelmistokehityksen tekoälytyökalu ChatGPT:n ohella. Nykyisen tekoälyhuuman tuoma uusi aalto on edistänyt myös ohjelmistokehityksessä käytettyjen työkalujen kehitystä tuomalla tekoälyn osaksi alan työkaluja. Koska näiden tekoälytyökalujen laajemmasta käyttöönotosta ei ole kulunut vielä kovin kauan, ei myöskään tieteellinen tutkimus ole vielä kerennyt tutkimaan aihealuetta tarkemmin. Tutkimuksia on varmasti tällä hetkellä menossa ja tekeillä, mutta koska tutkimusprosessi ja julkaiseminen vievät oman aikansa, vertailtaviksi saatavia tutkimuksia aiheesta on julkaistu tähän hetkeen vähän.

Seuraavaksi työssä tutustutaan tieteelliseen tutkimukseen Github Copilot -tekoälytyökalun käytöstä ja sen hyödyistä ohjelmistokehityksen alalla. Valitut tutkimukset edustavat laajalti viittauksina käytettyjä tutkimuksia tekoälytyökalujen toimivuuden ja hyödyllisyyden arvioinnissa erinäisissä lähteissä. Tässä täytyy huomioida, että työkalun tutkimusajankohdat eroavat tutkimuksissa hieman toisistaan ja tutkimustulokset edustavat menneisyyttä. Kuten aikaisemmin on jo todettu, tekoälykehitys on tällä hetkellä hurjaa ja siksi tutkimusten tulokset ovat vanhoja jo julkaistaessa.

4.1 Stack Overflow Labs 2023 Developer Survey

Stack Overflow on suosittu ohjelmistokehittäjille suunnattu verkkosivusto, joka on tunnettu paikana, josta voi etsiä ratkaisuja ohjelmointiongelmiiin. Tämän vuoden (2023) Stack Overflow:n vuosittaiseen kehittäjäkyselyyn oli yli 90.000 vastaajaa ja kyselyyn oli lisätty kysymyksiä koskien tekoälyä. 44 % vastaajista kertoi käyttävänsä jo tekoälytyökaluja kehitystyössään ja 26 % on suunnitellut tekoälytyökalujen käyttöönottoa. Tekoälytyökaluja käyttävistä vastaajista 83 % kertoi käyttävänsä ChatGPT:tä ja 56 % Github Copilottia, jotka olivat selvästi suosituimmat käytössä olevat tekoälytyökalut. Tutkijat epäilevät, että tekoälytyökalujen kapea-alainen käyttö johtuu siitä, että kehittäjät eivät vielä luota tekoälyteknologiaan riittävästi ja työkalujen tulee todistaa toimivuutensa, jotta ne saavuttavat laajemman käyttäjäkunnan. (Stack Overflow, 2023; Stack Overflow Labs, 2023)

Kehittäjien suhtautuminen tekoälyn käyttämiseen osana kehitystyötä on tutkimuksen mukaan erityisen korkea. Vastaajista 77 % vastaajista on suotuisia tekoälyn käyttämiseen osana kehitystyötä. Kysyttäessä tekoälytyökalujen hyödyistä, tärkeimmiksi hyödyiksi todettiin tuottavuuden kasvu (33 %), oppimisen nopeutuminen (25 %) ja tehokkuuden parantuminen (25 %). Suurin osa vastaajista käyttää tekoälytyökaluja koodin kirjoittamiseen (83 %) ja debuggaukseen (49 %), ja suurin osa uskookin näillä osa-alueilla tapahtuvan tekoälytyökaluissa kehitystä tulevan vuoden aikana. Uskon vahvasti, että ensi vuoden tutkimuksen tulokset osoittavat yhä kohonneita tekoälytyökalujen käyttöasteita ja koettua hyötyä, tekoälyhuuman intensiteetin ansiosta. (Stack Overflow, 2023; Stack Overflow Labs, 2023)

4.2 The Impact of AI on Developer Productivity: Evidence from GitHub Copilot - Sida Peng, Eirini Kalliamvakou, Peter Cihon & Mert Demirer

Peng (Microsoft Research), Kalliamvakou (GitHub), Cihon (Github) ja Demirer tutkivat (2023), kuinka tekoälytyökalut vaikuttavat tuottavuuteen ammattimaisessa ohjelmistokehityksessä. Tutkijoiden mukaan heidän tutkimuksensa on ensimmäinen laatuaan, joka tutkii tekoälytyökalujen tuottavuus vaikutusta ohjelmistokehityksessä, ja tutkimus on julkaistu helmikuussa 2023. Tutkimus toteutettiin kesällä 2022, juuri ennen Github Copilotin julkaisua, kontrolloituna kokeena, jossa 95 ammattilaisohjelmoijaa jaettiin satunnaisesti koe- ja vertailuryhmään. Osallistujien ohjelmointikokemuksen keskiarvo oli 6 vuotta. Koeryhmää ohjeistettiin katsomaan minuutin pituinen perehdytysvideo Github Copilottiin ja heille lähetettiin asennusohjeet Github Copilotin käyttöönottamiseksi. Osallistujien tehtävänä oli toteuttaa HTTP-palvelin JavaScript-ohjelmointikieltä käyttäen. Osallistuvat saivat käyttää ohjelmointityössään kaikkia mahdollisia apuvälineitä, joita he tavallisestikin työssään käyttävät, paitsi koeryhmällä oli lisäksi pääsy Github Copilot -tekoälytyökaluun. (Peng;Kalliamvakou;Cihon;& Demirer, 2023)

Tutkimuksessa seurattiin kahta mittaria: tehtävän läpäisyä ja tehtävään käytettyä aikaa. Osallistujien henkilökohtainen koodivarasto sisälsi tehtävänannon, koodipohjan ja 12 testiä tehtävän hyväksytyn suorituksen tarkistamiseksi. Tehtävän onnistumisen arviointi perustui läpäistyjen testien määrään, ja tehtävän kesto mitattiin siitä hetkestä, kun koodivarasto vastaanotettiin, siihen asti, kun kaikki testit oli läpäisty onnistuneesti. Tehtävän suorituksen jälkeen molemmille ryhmille lähetettiin loppukysely, jossa kysyttiin Github Copilotin hyödyllisyydestä. Vertailuryhmälle näytettiin tässä vaiheessa minuutin pituinen perehdytysvideo ja pyydettiin arvioimaan, kuinka paljon nopeammin he uskoisivat suorittaneensa tehtävän käyttämällä avukseen Github Copilot -tekoälytyökalua.

(Peng;Kalliamvakou;Cihon;& Demirer, 2023)

Tutkimuksen tuloksena koeryhmän keskimääräinen tehtävän suoritus aika oli n. 71 minuuttia, kun taas vertailuryhmällä keskimääräinen suoritus aika oli n. 161 minuuttia. Github Copilot -tekoälytyökalua käyttänyt koeryhmä suoriutui n. 56 % nopeammin tehtävästä kuin vertailuryhmä, joka suoritti tehtävän ilman tekoälytyökalua, jota voidaan pitää huomattavana tuottavuuden parantumisena. Loppukyselyn perusteella molemmat ryhmät, sekä koeryhmä että vertailuryhmä, arvioivat Github Copilot -tekoälytyökalun käytön parantavan tuottavuutta keskimäärin noin 35 %, joka on alle tutkimuksessa havaitun tuloksen. Loppukyselyssä kysyttiin myös hintaa, jonka osallistujat olisivat valmiita maksamaan GitHub Copilot -tekoälytyökalun käytöstä. Koeryhmä, joka käytti työkalua tehtävässä hyödykseen, oli valmis maksamaan työkalusta keskimäärin n. 27 dollaria, kun taas vertailuryhmä, joka näki vain minuutin perehdytysvideon työkalusta, oli valmis maksamaan keskimäärin n. 17 dollaria. Tämän perusteella voidaan epäsuorasti todeta, että koeryhmä hyötyi Github Copilotin käytöstä tehtävän suorituksessa, koska heidän valmiutensa maksaa työkalun käytöstä on huomattavasti korkeampi kuin vertailuryhmällä. Tutkimuksen päätelmänä tutkijat esittävät, että Github Copilotin käytöllä on tilastollisesti merkitsevä ja käytännöllisesti huomattava vaikutus ohjelmistokehittäjien tuottavuuteen. (Peng;Kalliamvakou;Cihon;& Demirer, 2023)

4.3 Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models – Priyan Vaithilingam, Tianyi Zhang & Elena Glassman

Vaithilingamin, Zhangin ja Glassmanin tutkimuksessa (2022) pyrittiin selvittämään, kuinka ohjelmistokehittäjät käyttävät Github Copilottia vertailemalla 24 osallistujan suoriutumista ohjelmointitehtävistä Python-ohjelmointikielillä. Tutkijoiden mukaan koodin generointi vaatii enemmän virheettömyyttä, kuin muut tekoälyn käyttökohteet, mutta sisältää myös epävarmuutta ja epätäydellisyyttä. Aikaisempien tutkimuksien osalta oli selvinnyt, että käyttäjien luottamus tekoälyä kohtaan laskee automaation tehdessä virheitä, mikäli selityksiä ei ollut tarjolla. Toisaalta selitykset voivat johtaa perusteettomaan yliuottamukseen, joka kertoo selityksien tarjoamisen tärkeydestä ja hankaluudesta. Aikaisemmat tutkimukset, joissa koodin generointi on toteutettu käyttäen muita kuin kielimalleja, eivät ole havainneet merkitsevää hyötyä koodia generoivien tekoälytyökalujen käytössä. Kielimallit ovat huomattavasti lisänneet generoidun koodin virheettömyyttä ja laatua, joka toimi lisämotivaationa tutkimukselle. (Vaithilingam;Zhang;& Glassman, 2022)

Tutkimusta varten suunniteltiin kolme eri vaikeustasoista tehtävää, joista jokainen osallistuja suoritti sattumanvaraisesti kaksi tehtävää. Jokainen osallistuja suoritti sattumanvaraisesti yhden tehtävän käyttäen Microsoftin Visual Studio -koodieditoreissa oletuksena olevaa koodintäydennystyökalua nimeltä IntelliSense sekä toisen tehtävän käyttäen Github Copilot -tekoälytyökalua. Ennen tehtävää osallistujat saivat nopean opastuksen tehtävässä käytettävästä työkalusta, ja he saivat suorituksen aikana käyttää internetiä hyväkseen. Suoritus arvioitiin hylätyksi, jos suorituksen hyväksytysti suorittamiseen kului yli 20 minuuttia. Jokaisen suorituksen jälkeen osallistujat vastasivat kyselyyn, koskien käytettyä työkalua. (Vaithilingam;Zhang;& Glassman, 2022)

	Task 1 - Easy		Task 2 - Medium		Task 3 - Hard	
	Intellisense	Copilot	Intellisense	Copilot	Intellisense	Copilot
	9:35	1:46	7:48	12:53	13:41	11:08
	3:50	3:57	15:52	16:45	13:43	11:05
	4:49	4:55	16:28	7:26	22:42	4:04
	9:04	6:18	14:16	15:05	13:06	DNF
	5:18	1:18	7:35	13:24	23:13	19:54
	15:54	7:52	12:39	DNF	4:48	DNF
	5:27	3:12	10:47	6:02	DNF	DNF
	2:09	20:12	8:30	DNF	DNF	9:19
Average Time	7:01	6:11	11:44	11:56	13:36	11:06
Overall average time for all tasks combined					10:23	9:18

KUVA 10. Tutkimuksen suoritusten suoritusajat (Vaithilingam;Zhang;& Glassman, 2022)

Tutkimuksen tuloksena Github Copilotin käytöllä ei havaittu saavutettavan enemmän hyväksytyjä suorituksia kuin IntelliSenseä käytettäessä, vaan päinvastoin enemmän suorituksia hylättiin Github Copilottia käytettäessä. Tämä ero hylätyissä suorituksissa ei tosin ole tilastollisesti merkittävä. Vaikka Github Copilottia käytettäessä suoritusten suoritusajojen keskiarvo oli nopeampi kuin IntelliSenseä käytettäessä, ei tämäkään ole tilastollisesti merkittävä tulos. Tutkimuksen yksittäiset ja keskiarvoilliset suoritusajat sekä hylätyt suoritukset (DNF) löytyvät kuvasta 10. Selkein ero hylättyjen suoritusten syissä oli, että Github Copilotin generoima koodi toimi virheellisesti ja pitkittänyt debuggaus johti aikarajan ylittymiseen. Kyselytutkimus kuitenkin paljastaa, että suurin osa osallistujista koki Github

Copilotin hyödyllisemmäksi työkaluksi ja suosi sen käyttöä verrattaessa IntelliSenseen. Github Copilotta pidettiin hyödyllisenä, koska se tarjoaa hyvän pohjan, josta lähteä liikenteeseen, vaikka generoitu koodi ei olisi aivan oikeanlaista. Ongelmana Github Copilotin käytössä pidettiin sitä, että generoitua koodia oli hankala ymmärtää ja että se tarjoaa vain yhden ratkaisun, jota ei pysty vertailemaan muihin vaihtoehtoihin, toimintaa, jollaista pystyy internetistä etsimällä tekemään. Näihin ongelmiin on kuitenkin haettu ratkaisua Github Copilotin uudemmissa versioissa, joissa koodin generointi sisältää kommentteja koodin ymmärtämiseksi ja pystyy selailemaan useampia vaihtoehtoja generoitavasta koodista. Uusimpien ominaisuuksien joukossa Github Copilot tarjoaa nykyisin chat-ominaisuutta, joka pystyy selittämään auki generoivansa koodin ja siltä pystyy kysymään lisäkysymyksiä epäselviin kohtiin. (Vaithilingam;Zhang;& Glassman, 2022)

4.4 An Empirical Evaluation of GitHub Copilot's Code Suggestions - Nhan Nguyen & Sarah Nadi

Nguyen ja Nadin tutkimus (2022) käsittelee empiiristä tutkimusta, jossa tutkijat testasivat kuinka hyvin Github Copilot suoriutuu 33 koodaustehtävästä neljällä eri ohjelmointikielellä: Python, Java, JavaScript ja C. Tutkijat arvioivat Github Copilotin ensimmäisenä tarjoaman ratkaisun oikeellisuutta ja monimutkaisuutta jokaiseen tehtävään. Tutkimuksessa käytetään hyödyksi ohjelmoijien keskuudessa suosittua LeetCode-sivustoa, joka tarjoaa koodaustehtäviä työhaastattelujen teknisiin haastatteluihin valmistautumiseen. Tutkijat valitsivat LeetCodesta satunnaisesti 33 eritasoista tehtävää Github Copilotille suoritettavaksi. Vastausten monimutkaisuutta arvioidaan kognitiivisen ja syklomaattisen (koodin sisältämien ehtojen lukumäärä) monimutkaisuuden mittareilla, joiden arvot lasketaan vastauksista käyttäen SonarQube-laaduntarkastustyökalua. (Nguyen & Nadi, 2022)

Valittujen tehtävien tehtävänannoista muodostettiin riittävät koodipohjat, jotta Github Copilot pystyy havaitsemaan kontekstin ja generoimaan oikeanlaiset muuttujat, funktiot ja paluuarvot, jotta tehtävien testit on mahdollista läpäistä. Tämän jälkeen Github Copilotilla generoitiin vastaukset jokaiseen tehtävään jokaisella ohjelmointikielellä koodipohjia käyttäen. Lopuksi generoidut vastaukset syötettiin LeetCoden koodausympäristöön testien läpi ajamiseksi. Kuvasta 11 löytyvät suoritettujen 132 tehtävän tulokset. (Nguyen & Nadi, 2022)

	Question	# Tests	Number (%) test cases passed			
			Python	Java	JavaScript	C
Easy	Q1	57	57 (100%)	57 (100%)	57 (100%)	57 (100%)
	Q2	14	14 (100%)	14 (100%)	14 (100%)	14 (100%)
	Q3	34	34 (100%)	34 (100%)	34 (100%)	34 (100%)
	Q4	15	15 (100%)	15 (100%)	15 (100%)	15 (100%)
Medium	Q5	81	50 (62%)	12 (15%)	6 (7%)	50 (62%)
	Q6	596	596 (100%)	596 (100%)	0 (0%)	0 (0%)
	Q7	85	82 (96%)	85 (100%)	6 (7%)	0 (0%)
	Q8	58	57 (98%)	47 (81%)	58 (100%)	58 (100%)
	Q9	116	114 (98%)	116 (100%)	0 (0%)	116 (100%)
	Q10	58	58 (100%)	58 (100%)	14 (24%)	10 (17%)
	Q11	54	54 (100%)	54 (100%)	0 (0%)	54 (100%)
	Q12	49	42 (85%)	49 (100%)	49 (100%)	24 (48%)
	Q13	47	47 (100%)	46 (98%)	11 (23%)	0 (0%)
	Q14	50	17 (34%)	50 (100%)	50 (100%)	50 (100%)
	Q15	202	202 (100%)	202 (100%)	202 (100%)	202 (100%)
	Q16	15	15 (100%)	15 (100%)	0 (0%)	14 (93%)
	Q17	188	188 (100%)	188 (100%)	36 (19%)	0 (0%)
	Q18	9	2 (22%)	0 (0%)	0 (0%)	9 (100%)
	Q19	57	0 (0%)	34 (59%)	0 (0%)	0 (0%)
	Q20	210	110 (52%)	19 (9%)	110 (52%)	0 (0%)
	Q21	15	10 (66%)	10 (66%)	0 (0%)	10 (66%)
Hard	Q22	84	83 (98%)	0 (0%)	84 (100%)	0 (0%)
	Q23	17	17 (100%)	17 (100%)	0 (0%)	17 (100%)
	Q24	64	64 (100%)	64 (100%)	3 (5%)	60 (93%)
	Q25	70	70 (100%)	0 (0%)	0 (0%)	0 (0%)
	Q26	30	0 (0%)	30 (100%)	0 (0%)	30 (100%)
	Q27	38	0 (0%)	38 (100%)	35 (92%)	38 (100%)
	Q28	39	1 (2%)	14 (35%)	26 (66%)	2 (5%)
	Q29	138	0 (0%)	138 (100%)	0 (0%)	0 (0%)
	Q30	51	24 (47%)	26 (50%)	0 (0%)	28 (54%)
	Q31	49	0 (0%)	39 (79%)	7 (14%)	0 (0%)
	Q32	44	0 (0%)	27 (61%)	0 (0%)	0 (0%)
	Q33	101	10 (9%)	10 (9%)	7 (6%)	5 (4%)
Accepted			14 (42%)	19 (57%)	9 (27%)	13 (39%)
Wrong answer			12 (36%)	12 (36%)	12 (36%)	8 (24%)
Time limit exceeded			2 (6%)	1 (3%)	1 (3%)	1 (3%)
Compile errors			0 (0%)	0 (0%)	0 (0%)	8 (24%)
Runtime errors			5 (15%)	1 (3%)	11 (33%)	3 (9%)
Partially correct			13 (39%)	11 (33%)	11 (33%)	9 (27%)

KUVA 11. Tutkimuksen suorituksen lopputulos ja onnistumisaste (Nguyen & Nadi, 2022)

Tulosten perusteella Github Copilotin generoima ensimmäinen vastausehdotus tehtävänantoon oli 57 % todennäköisyydellä täysin toimiva vastaus Javalla, 42 % Pythonilla, 39 % C:llä ja 27 % JavaScriptillä. Kun huomioidaan täysin oikeat ja osittain oikeat vastaukset, voidaan todeta Github Copilotin pystyvän avustamaan ohjelmoijaa 61–91 % todennäköisyydellä ohjelmointitehtävissä, riippuen ohjelmointikielystä. Github Copilotin vastausten monimutkaisuuden arvioinnissa ei huomattu tilastollisesti merkittävää eroa eri ohjelmointikielten välillä, ja lopputuloksena molemmilla monimutkaisuuden mittareilla, kognitiivisella ja syklomaattisella, mediaaniarvot jäivät huomattavan mataliksi. Näiden tulosten perusteella tutkijat toteavat Github Copilotin generoivan yleisesti ymmärrettävää koodia, jota ei voida pitää monimutkaisena. (Nguyen & Nadi, 2022)

4.5 GitHub Copilot AI pair programmer: Asset or Liability? - Arghavan Moradi Dakhel, Vahid Majdinasab, Amin Nikanjam, Foutse Khomh, Michel C. Desmarais & Zhen Ming Jiang

Dakhelin, Majdinasabin, Nikanjamin, Khomhin, Desmaraisin ja Jiangin tutkimus (2022) keskittyy selvittämään, kuinka laadukasta koodia Github Copilot generoi ja kuinka tehokkaasti kehittäjät pystyvät työkalua hyödyntämään. Aihetta tutkitaan 1) selvittämällä kuinka toimivia ja tehokkaita ratkaisuja Github Copilot tuottaa perustavanlaatuisiin algoritmisiin kysymyksiin, ja 2) vertailemalla Github Copilotin ehdottamia ratkaisuja ihmisohjelmoijien tuottamiin ratkaisuihin. Ensimmäiseen tutkimuskysymykseen tutkijat valitsivat Github Copilotille ratkottavaksi annettavat tehtävät tietotekniikan opiskelijoiden opetuskirjasta ja valitut tehtävät kattavat lajittelualgoritmit, tietorakenteet, graafialgoritmit sekä kehittyneet suunnittelu- ja analyysitekniikat. Toisessa tutkimuskysymyksessä Github Copilotin generoimia vastauksia verrataan Python-ohjelmointikurssin opiskelijoiden tuottamiin vastauksiin, joka kattaa 4000 vastauksen vertailujoukon. (Dakhel, ym., 2022)

Ensimmäisen tutkimuskysymyksen löydöksenä Github Copilotin todetaan ymmärtävän algoritmit näiden nimien perusteella ja pystyvän generoimaan toimivaa ja optimoitua koodia, kunhan tehtävänanto on lyhyt ja ytimekäs. Toisinaan Github Copilot kuitenkin generoi turhan monimutkaisia ratkaisuja yksinkertaisiin tehtävänantoihin. Toisessa tutkimuskysymyksessä havaittiin, että Github Copilotin vastaukset kilpailevat opiskelijoiden vastausten kanssa samassa sarjassa. Opiskelijoiden vastaukset ovat toimivampia, mutta myös vaihtelevampia. Github Copilotin generoimat vastaukset puolestaan ovat yksinkertaisempia ja helpompia korjata. Toisaalta, Github Copilotilla on vaikeuksia ymmärtää tiettyjä kohtia tehtävänannoista, joita opiskelijoilla ei kuitenkaan ole. (Dakhel, ym., 2022)

Tutkijoiden johtopäätös tutkimuksen tulosten perusteella on, että Github Copilot on hyödyksi asiantuntevissa käsissä, koska se pystyy tuottamaan yhtä laadukasta koodia kuin ihmisohjelmoijakin, mutta muuttuu haitaksi aloittelijoiden käytössä, koska he eivät välttämättä osaa suodattaa pois Github Copilotin virheellisiä ehdotuksia. Toisaalta Github Copilot avustaa ohjelmoinnin oppimisessa ja kehittämisessä juuri aloittelijoita. Tutkijat uskovat, että Github Copilotin käyttäjämäärien kasvu julkaisun myötä tulee kehittämään ja korjaamaan sen tämän hetkisiä vajavaisuuksia. (Dakhel, ym., 2022)

4.6 Tulosten yhteenveto

Esitelyjen tutkimusten tutkimustulokset ovat moninaisia ja osittain ristiriitaisia. Tuloksissa tulee kuitenkin huomioida, että tutkimukset on toteutettu ennen Github Copilotin julkaisua (10/2021) tai vuoden sisällä sen julkaisusta. Tällä hetkellä julkaisusta on kulunut kaksi vuotta, ja työkalua on kehitetty paljon eteenpäin, joten tutkimusten tulokset eivät ole relevantteja tässä hetkessä vaan peilaavat menneisyyteen.

Nguyen ja Nadin (2022) tutkimuksen mukaan Github Copilot onnistui noin 40 % todennäköisyydellä generoimaan ensimmäisellä yrittämällä toimivan vastausehdotuksen sille annettuun tehtävään. Kun tuloksissa huomioidaan lisäksi osittain oikeat vastausehdotukset, Github Copilotin voidaan nähdä pystyvän avustamaan ohjelmoijaa noin 75 % todennäköisyydellä tehtävissä. Toisaalta Vaithilingam, Zhang ja Glassman (2022) eivät havainneet Github Copilotin käytöllä saavutettavan enemmän hy-

väksytyjä suorituksia oman tutkimuksensa tehtävissä, vaan päinvastoin työkalun havaittiin aiheuttavan enemmän hylättyjä suorituksia. Dakhel, ym. (2022) rinnastavat Github Copilotin vastausehdotukset tietotekniikan opiskelijoiden tasoiksi, mutta toteavat silti opiskelijoiden vastausten olevan toimivampia ja samalla vaihtelevampia. Tutkimusasetelmat tutkimuksissa eroavat toisistaan, mutta tulokset viittaisivat silti vastakohtaisiin havaintoihin työkalun hyödyllisyydestä.

Vertailtaessa työkalun tuottamia vastausehdotuksia ihmisohjelmoijien tuottamiin vastauksiin täytyy lisäksi huomioida vastauksen tuottamiseen käytetty aika. Pengin, Kalliamvakoun, Cihonin ja Demirerin (2023) tutkimuksen perusteella Github Copilottia käyttänyt koeryhmä suoriutui noin 56 % prosenttia nopeammin tehtävistä kuin vertailuryhmä, joka ei käyttänyt Github Copilottia. Tulosta voidaan pitää huomattavana tuottavuuden parannuksena, ja tulos on tilastollisesti merkitsevä. Myös Vaithilingam, ym. (2022) havaitsivat omassa tutkimuksessaan Github Copilotin käytön nopeuttavan suoritusten keskiarvoa, mutta heidän tuloksensa ei ole tilastollisesti merkitsevä. Työkalun käytön vaikutus tuottavuuteen muodostuu pitkälti siitä, kuinka toimivan ratkaisun työkalu pystyy generoimaan, niin että ohjelmoijalta kuluu mahdollisimman vähän aikaa sen muokkaamisessa toimivaksi ratkaisuksi. Vaihtoehtoisena toimintatapana on kirjoittaa koodi kokonaan itse, jolloin sen ymmärtämiseen ja korjaamiseen ei tarvitse käyttää aikaa. Jos työkalun käytöllä ei pystytä nopeuttamaan tätä prosessia, niin ei sillä voida nähdä olevan positiivista vaikutusta tuottavuuteen. Ohjelmistokehitys pitää toki sisällään muitakin osa-alueita kuin pelkästään koodin kirjoittamista, mutta näiden mittaaminen on osoittanut hankalaksi, eikä akateeminen tutkimus ole vielä julkaissut tuloksia generatiivisten tekoälytyökalujen vaikutuksista muiden ohjelmistokehityksen osa-alueiden tuottavuuteen.

Mitä tulee Github Copilotin generoiman koodin monimutkaisuuteen, niin sekä Nguyen ja Nadin (2022) että Dakhelin, ym. (2022) tutkimusten tulosten perusteella työkalun generoimaa koodia voidaan pitää yksinkertaisena ja ymmärrettävänä. Dakhel ym. (2022) havaitsivat kuitenkin, että toisinaan Github Copilot generoi turhan monimutkaisia vastauksia yksinkertaisiin tehtäviin. Mitattavien tulosten lisäksi työkalun hyödyllisyyttä voidaan arvioida käyttäjien kokemusten perusteella. Vaithilingam, ym. (2022) kyselytutkimuksessa selvisi, että käyttäjät kokivat Github Copilotin käytön hyödylliseksi nähtyään minuutin mittaisen esittelyvideon ja käytettyään työkalua kahta tehtävää ratkaistessaan. Github Copilottia pidettiin hyödyllisenä, vaikka generoitu ratkaisu ei olisi ollut aivan oikeanlainen, sillä se tarjoaa hyvän pohjan, josta lähteä rakentamaan toimivaa ratkaisua. Koeryhmän, joka käytti työkalua hyväkseen, ja vertailuryhmän, joka ei käyttänyt työkalua, antama keskimääräinen arvio Github Copilotin tuottavuuden parantamisesta oli noin +35 %. Tämän lisäksi koeryhmä oli valmis maksamaan Github Copilot -työkalun käytöstä enemmän kuin vertailuryhmä, joka kertoo siitä, että käyttäjien kokema hyöty on ollut vielä arvioitua hyötyä korkeampi.

5 KÄYTÄNNÖN KOKEILU

5.1 Kokeilun lähtökohdat ja määrittely

Opinnäytetyön toimeksiantajalla, Ponsella, haluttiin lähteä kokeilemaan tekoälytyökaluja ohjelmistokehityksen parissa. Toimeksiantajalla ei ole aikaisemmin ollut vastaavia työkaluja käytössä. Päätöksen taustalla on tekoälyhuuman myötä tullut tietoisuus tekoälyn mahdollisuuksista ja mahdollisuus tehostaa ohjelmistokehitystyötä. Mikäli tekoälytyökaluja hyödyntämällä voitaisiin saavuttaa edes vähäistä hyötyä työn tehokkuudessa ja tuottavuudessa laajalla käyttäjäryhmällä, olisi investointi kannattavaa, sillä työkalun käyttökulut ovat vähäiset. Työkalun käytön tietoturvakysymykset, taustaselvitykset ja työkalun käyttöönottoprosessi sitovat enemmän resursseja ja ovat huomattavampi kuuerä, kuin itse työkalun juoksevat kulut.

Ponsella päätettiin lähteä kokeilemaan Github Copilot -tekoälytyökalua, koska se on markkinoiden tunnetuin tuote ja sen käytöstä on saatavilla eniten tietoa. Päätökseen vaikutti varmasti myös se, että Github Copilotilla on asiakkainaan monia suuria yrityksiä, jolloin tuotteen voi olettaa olevan toimiva. Lisäksi Githubin taustalla oleva Microsoftin omistusyhteys tuo työkalulle uskottavuutta ja luotettavuutta yritysmaailmassa. Microsoftin tuotteisiin on yleisesti hyvin saatavilla konsultointiapua, sekä tuotteiden ja palveluiden hinnoittelussa etu kasvaa volyymien myötä.

Github Copilot otettiin kokeiluun testiryhmälle, joka työskenteli kokeilujakson ajan saman projektin parissa. Testiryhmän käyttäjät, noin kymmenen henkeä, kattavat ohjelmistokehityksen frontend-, backend-, DevOps-kehityksen ja testauksen osa-alueet ja vaiheet, joten testiryhmän kautta ohjelmistokehityksen erinäiset tehtävät tulevat kokeiltua työkalua käyttäen. Laaja-alainen ohjelmistokehityksen eri osa-alueissa ja vaiheissa tapahtuva työkalun kokeilu tuo parempaa näkemystä työkalun hyödyistä ja haitoista kokonaisuutena. Testiryhmällä oli jo entuudestaan tietoa Github Copilot -tekoälytyökalusta ja osalla oli jo kokemusta työkalun käytöstä, ennen kokeilujakson alkamista. Heitä, joilla ei ollut entuudestaan kokemusta Github Copilotin käytöstä, opastettiin katsomaan etukäteen valitut opastusvideot, joissa esiteltiin Github Copilotin käyttöä ja sen ominaisuuksia, jotta työkalun käytön aloittaminen olisi mahdollisimman nopeaa ja kokemusta työkalun käytöstä kertyisi mahdollisimman paljon kokeilujaksolta. Kokeilujakson pituudeksi muodostui noin yhden kuukauden mittainen jakso, jonka jälkeen testikäyttäjille lähetettiin kysely Github Copilotin käyttökokemuksista ja työkalun koetuista hyödyistä sekä haitoista.

5.2 Kokeilujakson odotukset

Aikaisempien tieteellisten tutkimusten tulokset ovat olleet enemmän positiivisia kuin negatiivisia, eikä tutkimuksissa ole noussut esille suurempia haittoja työkalun käytöstä. Tämän takia myös odotukset kokeilujakson tuloksille ovat positiiviset. Varhaisimmat tässä työssä esitellyistä tutkimuksista ovat toteutettu aikaan, jolloin Github Copilottia ei ollut vielä julkaistu julkiseen käyttöön, pari vuotta sitten, ja tietäen kuinka paljon tekoälykehitys on edistynyt näinä vuosina, oletuksena on, että myös Github Copilot on kehittynyt paljon tässä ajassa. Mediassa on myös jaettu paljon tietoa siitä, kuinka ihmeellistä tekoäly on ja mitä kaikkea Github Copilot pystyy tekemään, joten siltäkin osin odotukset ovat, että työkalun käytöllä olisi konkreettista vaikutusta kehittäjän työhön. Jos työkalun käytöllä

pystytään helpottamaan tai nopeuttamaan koodin kirjoittamista tai parantamaan ohjelmistokehittäjän tuottavuutta, koen työkalun täyttävän odotuksensa. Jos tulokset kuitenkin ovat huomattavasti merkittävämmät, olen innoissani kaikista niistä mahdollisuuksista, joita tekoälyn kehitys tuo mukanaan ohjelmistokehityksen alalle ja siitä laajamittaisesta hyödystä, jonka se mahdollistaa esimerkiksi toimeksiantajan toimintaan. Toisaalta on mielenkiintoista päästä vertailemaan, kuinka käytännön kokeilun kautta havaitut kokemukset vertautuvat markkinoituihin parannuksiin ja hyötyihin sekä tieteellisissä tutkimuksissa esiteltyihin tuloksiin.

5.3 Kyselytutkimuksen suunnittelu

Käytännön tutkimuksen toteutustavaksi valittiin kyselytutkimus, koska ohjelmistokehittäjän suoriutumisen arvioimiseksi on hankala määritellä sopivia mittareita, joten kokemusperäinen arviointi on mielestäni toiseksi paras mahdollinen tapa toteuttaa tämän mittakaavan tutkimus. Kokemusten kerääminen on hyvin intuitiivinen ja luotettava tutkimustapa kerätä vastauksia ja jos vastauksia on kerätty riittävän laajalta joukolta, siitä pystyy selvästi kiteyttämään keskeisimmät tulokset. Kyselytutkimuksen suurin heikkous piilee siinä, että voidaanko tutkimuksen tuloksiin luottaa, eli pystytäänkö luottamaan, että vastaajat ovat vastanneet kyselyyn rehellisesti. Tämän en kuitenkaan usko olevan tämän työn ongelmana, sillä kokeilu ja kysely toteutetaan hyvin matalalla tasolla ja sen tulokset voivat vaikuttaa vastaajien omaan työskentelyyn tulevaisuudessa. Niinpä uskon koekäyttäjille muodostuvan selkeän näkemyksen koejakson aikana siitä, että haluavatko he jatkaa tulevaisuudessa työskentelyään työkalun kanssa vai ilman. Lisäksi kun vaatimukset ovat hyvin matalat, riittävät kyselytutkimuksen kokemusperäiset tulokset selvittämään onko työkalun käytöllä saavutettavissa hyötyjä, jotka toimisivat perusteina työkalun laajemmalle käyttöönotolle.

Kyselyyn vastaaminen on vapaaehtoista ja kyselyyn vastataan nimettömänä. Kyselyn alussa vastaajilta kysytään muutamia taustatietoja tutkimuksen johtopäätöksien tekemistä varten. Monivalintakysymysten avulla pyritään selvittämään, kokiko vastaaja Github Copilot -työkalun hyödyllisenä, nopeuttavan koodin kirjoittamista, parantavan tuottavuutta sekä generoivan laadukasta koodia. Kyselyn lopussa on kolme avointa kysymystä, joiden avulla selvitetään, kuinka työkalun käyttö muutti työskentelyä, mitä pidettiin työkalun tärkeimpänä hyötynä ja missä olisi työkalussa eniten parannettavaa. Kysymyksiä laatiessa on huomioitu tässä työssä aikaisemmin esiteltyjen tutkimusten löydökset, jotta kyselytutkimuksen vastausten johtopäätöksiä voidaan verrata näiden aikaisempien tieteellisten tutkimusten johtopäätöksiin.

Kyselytutkimus (liite 1) kattaa yhteensä 15 kysymystä: neljä taustatietokysymystä, kahdeksan monivalintakysymystä ja kolme avointa kysymystä. Kysely on toteutettu englanniksi, jotta jokaisella testiryhmän jäsenellä olisi sama kysely vastattavanaan, jolla vältytään kysymysten tulkintaeroista ja tulkintaerojen vaikutuksilta tuloksiin. Kyselyyn vastaamiseen kuluu arvioilta enimmillään 10–20 minuuttia ja se on suunniteltu helposti vastattavaksi, jotta vastaukset olisivat intuitiivisia ja ettei kyselyn täyttäminen tuntuisi liian työläältä ja jäisi sen takia tekemättä. Noin yhden kuukauden mittaisen kokeilujakson jälkeen testiryhmän jäsenille lähetettiin kysely vapaaehtoisesti vastattavaksi. Kun vastausaika oli umpeutunut, kaikki vastaukset kerättiin yhteen ja aloitettiin vastausten käsittely tulosten selvittämiseksi.

5.4 Tulokset ja johtopäätökset

Participant	A	B	C	D	E	F	G
Programming experience in years	3	13	2	3	9	4	5
Copilot experience in months	3	10	2	6	20	13	1
Used Copilot's code completion	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Used Copilot's chat	Yes	No	Yes	Yes	No	Yes	Yes
Was Copilot useful	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Did Copilot speed up coding	Yes	Yes	Yes	Yes	Yes	Yes	Yes
How much did it speed up	60–80 %	20–40 %	40–60 %	20–40 %	20–40 %	20–40 %	20–40 %
Did Copilot improve productivity	Yes	Yes	Yes	No	Yes	Yes	Yes
How much did it improve	80–100 %	20–40 %	40–60 %	-	20–40 %	20–40 %	0–20 %
How functional was the code	Functional	Buggy	Functional	Functional	Functional	Functional	Functional
How understandable was the code	Very easy	Easy	Very easy	Easy	Easy	Easy	Easy
How complex was the code	Very simple	Don't know	Simple	Don't know	Complex	Simple	Simple

KUVA 12. Kyselyn monivalintakysymysten tulokset

Kyselyyn tuli seitsemän vastausta. Vastaukset anonymisoitiin ja jaettiin monivalintakysymyksiin ja avoimiin kysymyksiin, ennen vastausten analysointia. Kuvassa 12 on esitetty koonti monivalintakysymysten vastauksista. Vastaajien joukko on moninainen, sillä vastaajien ohjelmointikokemus vaihtelee muutamasta vuodesta yli kymmenen vuoden kokemukseen ja Github Copilotin käyttökokemus vaihtelee kuukaudesta melkein kahden vuoden käyttökokemukseen asti. Vastaajista kaikki olivat käyttäneet Github Copilotin automaattista koodintäydennystä (joka on työkalun perusominaisuus) ja melkein kaikki olivat käyttäneet työkalun chat-ominaisuutta. Vastaajajoukon kaksi kokeneinta ohjelmoijaa eivät olleet käyttäneet Github Copilotin Chat -ominaisuutta, joka on yllättävä havainto. Tämän voisi selittää se, että he eivät ole kokeneet hyödylliseksi tällaista ominaisuutta omassa työskentelyssään tai sitten heillä voi olla käytössä esim. ChatGPT tämän ominaisuuden sijasta. Tästä nouseekin heti esille parannusehdotus, että kyselyssä olisi voinut kysyä käyttävätkö vastaajat joitain muita tekoälytyökaluja hyväkseen työssään ja jos käyttävät, niin mitä työkaluja he käyttävät.

Kaikki kyselyyn vastanneet olivat yhtä mieltä, että Github Copilot -työkalu oli hyödyllinen heidän työssään. Kaikki vastaajat olivat myös sitä mieltä, että Github Copilot nopeutti koodin kirjoittamista ja kaikki muut paitsi yksi vastaaja olivat sitä mieltä, että työkalu paransi myös tuottavuutta. Kysyttäessä kuinka paljon vastaajat arvioivat Github Copilotin nopeuttavan koodin kirjoittamista, vastausten vaihteluväli on 20–80 % ja vastausten keskiarvo on arvovälissä 20–40 %. Tuottavuuden paranemista arvioitaessa on enemmän vaihtelua kuin koodin kirjoitusnopeuden paranemista arvioitaessa. Yksi vastaaja ei kokenut työkalun käytön parantavan tuottavuutta ja muiden vastaajien vastausten vaihteluväli on 0–100 %, keskiarvon asettuessa 20–40 % arvoväliin. Kysyttäessä Github Copilotin generoiman koodin toimivuutta ja ymmärrettävyyttä, vastaajat ovat selkeästi sitä mieltä, että generoitu koodi on toimivaa ja ymmärrettävää, muutamaa eriävää vastausta lukuun ottamatta. Koodin monimutkaisuutta kysyttäessä, hajontaa on enemmän ja vastaajista kaksi ei ole osannut sanoa mielipidettään asiaan. Jos arvioidaan koodin monimutkaisuuden keskimääräistä vastausta, niin silloin vastaajien mielestä Github Copilotin generoimaa koodia voidaan pitää yksinkertaisena. Kokonaisuutena monivalintakysymysten vastaukset ovat hyvin yhtenevät, ja niistä saa selkeän kuvan siitä, millaisena vastaajat ovat kokeneet Github Copilotin käytön osana työtään.

Myös avointen kysymysten vastauksista on löydettävissä asioita, jotka toistuvat vastaajien vastauksissa. Kysymyksen, miten Github Copilotin käyttö muutti työtäsi aiempaan verrattuna, vastauksista nousee esille, että vastaajat ovat käyttäneet vähemmän aikaa etsien vastauksia ongelmiin internetistä, Github Copilot on auttanut paljon toistuvassa ja yksitoikkoisessa työssä ja työkalusta on ollut hyötyä tehtävien aloittamisessa luomalla hyvän pohjan, josta lähteä liikkeelle. Vastauksissa nousee esille myös, että työkalun käyttö vaatii opettelua, jotta siitä saa irti parhaimman hyödyn, jonka takia työkalun käyttö voi aluksi hidastaa tai heikentää tuottavuutta, mutta hyödyttää pidemmällä aikavälillä. Kysyttäessä Github Copilotin tärkeimmästä hyödystä, vastauksissa toistuvat aikaisemmassa kysymyksessä esille nousseet asiat. Koodipohjan luonti, toistuvien tehtävien suorittaminen, vähemmän aikaa käytetty selaimessa ongelmanratkaisuun, yksikkötestien luominen ja kontekstietoinen chat-ominaisuus ovat tärkeimmiksi koetut hyödyt Github Copilotin käytölle. Erään vastauksen lause kuvastaa mielestäni hyvin aikaisemmin lueteltuja ominaisuuksia ja niistä ohjelmistokehittäjällä tulevaa hyötyä: *"Säästää minulta aikaa tylsissä ja arkipäiväisissä tehtävissä, joten voin keskittyä vaikeisiin tehtäviin"*.

Github Copilotin suurimmat kehityskohteet kyselyn perusteella ovat laajempi kontekstietoisuus ja nopeammat generoinnit. Työkalun toivottaisiin ottavan kontekstissa huomioon koko projektin, eikä pelkästään aktiivista tiedostoa tai kursorin ympärille olevaa koodia. Lisäksi esille nostettiin työkalun chat-ominaisuuden käyttöliittymän parantaminen tekstieditorissa sekä yleisesti generoidun koodin laadun parantaminen. Yksi vastaajista kertoo päätyneensä käyttämään ChatGPT:tä Github Copilotin Chat -ominaisuuden sijasta, koska ChatGPT antaa kokonaisvaltaisemman vastauksen. Chat-ominaisuus on Github Copilotin tuoreimpia uudistuksia, joten esille nousseet kehityskohteet voivat olla alkukankeutta ja näihin saataisiin parannuksia tulevaisuudessa. Toisaalta chat-ominaisuuden tulisi olla vähintään ChatGPT:n tasoinen, jotta käyttäjät ottaisivat ominaisuuden käyttöönsä ja välttyisivät tiedonhaun aiheuttamasta siirtymisestä tekstieditorista ulkoiseen selaimeen, jossa on omat haittansa.

Seuraavaksi vertaillaan käytännön kyselytutkimuksessa saatuja tutkimustuloksia aikaisemmin työssä esiteltyjen tieteellisten tutkimusten tuloksiin. Käytännön tutkimuksessa ei selvitetty Github Copilotin vastausehdotusten prosentuaalista toimivuutta, kuten Nguyen ja Nadin tutkimuksessa (2022), jossa todetaan työkalun onnistuvan noin 40 % todennäköisyydellä generoimaan toimivan vastausehdotuksen ensimmäisellä yrittämällä. Myöskään Vaithilingamin, Zhangin ja Glassmanin tutkimuksen (2022) löydökselle, että Github Copilotin käyttö aiheutti enemmän hylättyjä suorituksia kuin ilman työkalua tehdyissä suorituksissa, ei ole vastaavan kaltaista kohtaa käytännön tutkimuksessa. Kuitenkin suurin osa käytännön kyselytutkimuksen vastaajista oli sitä mieltä, että Github Copilot paransi tuottavuutta ja että työkalun generoima koodi on toimivaa. Vaithilingam, ym. kyselytutkimuksessa (2022) on kuitenkin vastaava kohta kuin käytännön tutkimuksessa, jossa kysytään vastaajien kokemusta Github Copilot -työkalun hyödyllisyydestä, ja molemmissa tutkimuksissa vastaajat kokevat työkalun käytön hyödylliseksi.

Peng, Kalliamvakou, Cihon ja Demirer havaitsevat tutkimuksessaan (2023), että Github Copilottia käyttänyt koeryhmä suoriutui tehtävistä nopeammin kuin vertailuryhmä ja myös Vaithilingam, ym. toteavat omassa tutkimuksessaan (2022) Github Copilotin käytön nopeuttavan ohjelmointitehtävien

suoritusaikojen keskiarvoa. Tämä on linjassaan käytännön tutkimuksen kanssa, jossa kaikki vastanneet olivat sitä mieltä, että työkalun käyttö nopeuttaa ohjelmointia ja vastaajien vastausten keskiarvo on, että työkalu nopeuttaa ohjelmointia arviolta 20–40 %. Mitä tulee Github Copilotin generoiman koodin monimutkaisuuteen ja ymmärrettävyyteen, niin käytännön tutkimuksen tulokset vastaavat sekä Nguyen ja Nadin (2022), että Dakhelin, Majdinasabin, Nikanjamin, Khomhin, Desmaraisin ja Jiangin (2022) tutkimuksien tuloksia, että työkalun generoima koodi on yksinkertaista ja ymmärrettävää. Kokonaisuutena voidaan todeta, että käytännön kyselytutkimuksen tulokset ovat linjassaan tieteellisten tutkimusten tulosten kanssa ja tulokset osoittavat, että Github Copilot -tekoälytyökalun käytöstä on hyötyä ohjelmistokehityksessä.

6 POHDINTA

Opinnäytetyön tarkoituksena oli perehtyä tekoälyyn, sen historiaan ja teoriaan, tutkia millaisia tekoälytyökaluja on saatavilla ohjelmointiin ja selvittää minkälaista hyötyä työkalut tarjoavat ohjelmoijalle. Työn toimeksiantaja oli kiinnostunut toteuttamaan kokeilun tekoälytyökalun käytöstä ja saamaan selville saavuttaako tällaista työkalua käyttämällä hyötyä ohjelmistokehityksessä. Työn tekemiseen varattiin neljän kuukauden ajanjakso syyskuusta joulukuuhun vuonna 2023. Sisältö aikataulutettiin kolmeen osaan, joille varattiin lähes samansuuruiset ajanjaksot: tekoälyn historia ja teoria, tieteellinen tutkimus ja mahdolliset työkalut sekä toimeksiantajan käytännön tutkimus.

Lopputulos vastaa tavoitetta. Tavoite jalostui työn edetessä ja näkemys lopullisen työn sisällöstä kirjastui aiheeseen perehdyttäessä syvällisemmin. Toimeksiantajalle löydettiin sopiva työkalu kokeiltavaksi ja tieteellisten tutkimusten sekä käytännön tutkimuksen kautta saatiin tuloksia tutkimuskysymykseen vastaukseksi. Suunniteltu aikataulu toteutui. Tämän mahdollisti vahva työn aloitus, sillä loppua kohti mentäessä prosessissa ilmeni hidasteita, joista lisää myöhemmin tässä luvussa. Näin pitkään projektiin mahtuu jo mukaan henkistä motivaation vaihtelua, eikä opinnäytetyö ole poikkeus. Onneksi alamäistä ja kuopista selvittiin ja työtä saatiin edistettyä jatkuvasti. Oli ilahduttavaa huomata kuinka välillä tuli saavutettua jonkinlainen flow-tila ja työn tekeminen vei mennessään.

Opinnäytetyön aihe on ajankohtainen ja siihen syvempi perehtyminen on ollut mielenkiintoista ja uskon sen olleen myös hyödyllistä tulevaisuuden työelämän kannalta. Tekoäly on nyt tullut osaksi työelämää. Tutkimuskysymys, onko tekoälytyökaluista hyötyä ohjelmistokehityksessä, on myös mietityttänyt itseäni, eikä pelkästään työn toimeksiantajaa, ja on ollut mukava saavuttaa samalla vastaus itselleen. Tulen käyttämään mahdollisia tekoälytyökaluja käytössä tulevaisuuden ohjelmistokehityksessäni. Olen vakuuttunut, että tekoälytyökalut ovat ainakin kulujensa arvoiset.

Tekoälyn tämänhetkinen kehitys on nopeatempoista. Tätä työtä tehtäessä, neljän kuukauden aikana, jouduttiin muutamaa alkupään kohtaa päivittämään, koska faktat muuttuivat työn aikana. Syyskuun lopulla vuonna 2023, yksitoista kuukautta julkisessa käytössä oltauan, ChatGPT sai pääsyn Internetiin, joka yllätti minut täysin. Mielikuvini oli iskostunut laki, että GPT-mallien koulutusaineisto jäädyttää näiden tietämyksen tiettyyn ajankohtaan. Tekoälyn käyttökohteiden laajentuminen ja tekoälyn kehittyminen tulevat johtamaan siihen, että monen ammatin työnteko tulee muuttumaan, eikä vähiten ohjelmointi, jossa on jo havaittavissa uuden aikakauden alun tunnelmaa. Jos Githubin raportoima luku, että tekoäly kirjoittaa tällä hetkellä yli 40 % kaikesta uudesta koodista, pitää paikkansa, näin todella on. Tuleeko tekoäly kirjoittamaan jatkossa kaiken koodin, tuleeko tekoäly syrjäyttämään ohjelmoijat ja miten tekoälyn laajentuminen muille ammattialoille tulee vaikuttamaan näihin työtehtäviin, ovat kysymyksiä, joihin emme vielä tiedä vastauksia, mutta joita monet ovat jo pohtineet. Muutos ei välttämättä ole negatiivista, vaikka niin me ihmiset sen ensimmäiseksi mielimme. Low-code- ja no-code-ohjelmistokehityksen suosio on ollut jatkuvasti kasvussa, kun sovel-luskehittämisen tarve on lisääntynyt, kehitystä tulisi saada nopeutettua ja alalla on ollut osaajapulaa. Mielestäni tekoäly osuu saamaan tarpeeseen kuin low-code- ja no-code-ohjelmistokehitys, eli tekoälyä hyödyntämällä päästään helpottamaan ja nopeuttamaan ohjelmistokehitysprosesseja.

Mikä yllätti kaikkein eniten työn aikana, ja mitä ei ollut tullut ajatelleeksi työn suunnittelussa, oli se kuinka hankala prosessi Github Copilotin käyttöönotto on yritysasiakkaalle. Ensinnäkin ostaminen on hankalaa, työkalun käyttöönotto vaatii ylimääräisten palveluiden olemassaoloa ja Githubin ohjeistus käyttöönottoon on suurpiirteistä. Ilmeisesti Github Copilotin myyminen itsenäisenä kokonaisuutena ei ole Githubille järkevää, vaan he tavoittelevat laajempia asiakaskokonaisuuksia. Näitten syitten takia käytännön tutkimuksen aloitus viivästyi ja kokeiluajanjakso lyheni. Onneksi kuitenkin olin aloittanut opinnäytetyön tekemisen reippaasti, eikä viivästys vaikuttanut työn valmistumiseen. Myöskään lyhentynyt kokeilujakso ei mielestäni vaikuta työn luotettavuuteen, mutta pidempi kokeilujakso olisi tarjonnut varmasti parempaa kuvaa työkalun todellisesta käytöstä. Työn kannalta oli kuitenkin mukava huomata, että useammalla testiryhmän jäsenellä oli kuitenkin jo aikaisempaa kokemusta Github Copilotin käytöstä, joka paikkaa lyhennettyä kokeilujaksoa.

Kokonaisuutena opinnäytetyöprosessi sujui suunnitelman mukaisesti ilman suurempia vastoinkäymisiä. Tähän vaikutti varmasti aikaisempi kokemus lopputöiden tekemisestä, jolloin tiesi mitä henkisesti odottaa ja mitä konkreettisesti vaaditaan työn saattamiseksi lopulliseen muotoonsa. Jos lähtisin tekemään työtä uudelleen alusta asti, keskittyisin vahvemmin kokeilujaksoon, ja jos resurssit mahdollistaisivat, ottaisin kokeiluun useamman tekoälytyökalun vertailtavaksi, jotta voisi havaita miten työkalut eroavat toisistaan, ja onko niistä selkeästi löydettävissä paras vaihtoehto. Jatkotutkimuksena voisi toteuttaa selvityksen Github Copilotin tai jonkin toisen ohjelmistokehityksen tekoälytyökalun käyttöönottoprosessista yrityksissä. Tämä jäi itselleni päällimmäisenä mieleen, johon olisin halunnut itse saada enemmän selvyyttä aihealueesta. Toisena jatkotutkimuskohteena olisin halunnut tutustua paremmin Tabnine-tekoälytyökaluun, joka oman tutkimukseni mukaan vastaa hyvin paljon Github Copilottia. Tabnine on itsenäinen yritys, eikä ole sidoksissa suurempaan palveluntarjoajaan, joten olisi mielenkiintoista tietää kuinka työkalun käyttö, hyödyt ja käyttöönotto eroavat Github Copilotin vastaavista.

Uskon, että tekoäly on tullut osaksi ohjelmistokehitystä pysyvästi. Aiheeseen perehtyminen on ollut hyödyllistä tulevaisuuden osaamisen kannalta. Opinnäytetyön lopputuloksena suositan toimeksiantajaa ottamaan käyttöön tekoälytyökalun osaksi ohjelmistokehitystään. Vaikka investointi siihen on näkemykseni mukaan jo nyt kannattavaa, tulee se olemaan entistä enemmän tulevaisuudessa.

LÄHTEET

- Aaltonen, M. (2019). *Tekoäly - ihminen ja kone*. Helsinki: Alma Talent Oy.
- Alpaydin, E. (2021). *Koneoppiminen*. (K. Pietiläinen, Käänt.) Helsinki: Terra Cognita Oy.
- Amazon Web Services. (2023). *Amazon CodeWhisperer*. Haettu 9.11.2023 osoitteesta Amazon: <https://aws.amazon.com/codewhisperer/>
- Chow, A. R.;& Perrigo, B. (17.2.2023). The AI Arms Race Is Changing Everything. *Time*. Haettu 8.9.2023 osoitteesta <https://time.com/6255952/ai-impact-chatgpt-microsoft-google/>
- Council of Europe. (2023). *History of Artificial Intelligence*. Haettu 6.9.2023 osoitteesta Council of Europe: <https://www.coe.int/en/web/artificial-intelligence/history-of-ai>
- Dakhel, A. M.;Majdinasab, V.;Nikanjam, A.;Khomh, F.;Desmarais, M. C.;& Jiang, Z. M. (2022). *GitHub Copilot AI pair programmer: Asset or Liability?* Tutkimus. Haettu 19.10.2023 osoitteesta <https://arxiv.org/abs/2206.15331>
- Dohmke, T. (14.2.2023). *GitHub Copilot for Business is now available*. Haettu 8.11.2023 osoitteesta Github Blog: <https://github.blog/2023-02-14-github-copilot-for-business-is-now-available/>
- Gates, B. (21.3.2023). The Age of AI has begun. *Gates Notes*. Haettu 8.9.2023 osoitteesta <https://www.gatesnotes.com/The-Age-of-AI-Has-Begun>
- Github. (2023). *Your AI pair programmer*. Haettu 8.11.2023 osoitteesta Github: <https://github.com/features/copilot>
- Järvinen, P. (2023). *Tekoäly ja minä - ihmisenä tekoälyn aikakaudella*. Helsinki: Tammi.
- Kelleher, j. D. (2020). *Syväoppiminen*. (K. Pietiläinen, Käänt.) Helsinki: Terra Cognita Oy.
- Koch, C. (19.3.2016). How the Computer Beat the Go Master. *Scientific American*. Haettu 7.9.2023 osoitteesta <https://www.scientificamerican.com/article/how-the-computer-beat-the-go-master/>
- Kohs, G. (Ohjaaja). (2017). *AlphaGo - The Movie* [Elokuva]. Haettu 13.9.2023 osoitteesta <https://www.youtube.com/watch?v=WXuK6gekU1Y>
- Kolari, J.;& Kallio, A. (2023). *Tekoäly 123 : matkaopas tulevaisuuteen*. Jyväskylä: Docendo.
- Levy, S. (11.8.2023). The 'Godfather of AI' Has a Hopeful Plan for Keeping Future AI Friendly. *Wired*. Haettu 14.9.2023 osoitteesta <https://www.wired.com/story/plaintext-geoffrey-hinton-godfather-of-ai-future-ai/>
- Marshall, B. (9.5.2023). *3 Unsavory Consequences of Generative AI*. Haettu 13.11.2023 osoitteesta Medium: <https://medium.com/@brandeismarshall/3-unsavory-consequences-of-generative-ai-2b9f5c29f52b>
- Martineau, K. (20.4.2023). *What is generative AI?* Haettu 4.10.2023 osoitteesta IBM Research: <https://research.ibm.com/blog/what-is-generative-AI>
- Microsoft. (2023). *IntelliCode*. Haettu 2.11.2023 osoitteesta Microsoft Visual Studio: <https://visualstudio.microsoft.com/services/intellicode/>

- Microsoft. (30.9.2023). *IntelliCode*. Haettu 2.11.2023 osoitteesta Microsoft Learn: <https://learn.microsoft.com/en-us/visualstudio/ide/intellicode-visual-studio?view=vs-2022>
- MinnaLearn & University of Helsinki. (10.6.2022). Elements of AI -online course. Haettu 14.9.2023 osoitteesta <https://www.elementsofai.com/>
- Nguyen, N.;& Nadi, S. (2022). *An Empirical Evaluation of GitHub Copilot's Code Suggestions*. Tutkimus. Haettu 17.10.2023 osoitteesta <https://dl.acm.org/doi/10.1145/3524842.3528470>
- OpenAI. (10.8.2021). *OpenAI Codex*. Haettu 7.11.2023 osoitteesta OpenAI: <https://openai.com/blog/openai-codex>
- OpenAI: Altman, Sam; Brockman, Greg; Sutskever, Ilya. (22.5.2023). *Governance of superintelligence*. Haettu 18.9.2023 osoitteesta OpenAI.com: <https://openai.com/blog/governance-of-superintelligence>
- Peltomäki, T. (17.2.2023). Tekoälyn takapiru. *Helsingin Sanomat*. Haettu 14.9.2023 osoitteesta <https://www.hs.fi/visio/art-2000009390029.html>
- Peng, S.;Kalliamvakou, E.;Cihon, P.;& Demirer, M. (2023). *The Impact of AI on Developer Productivity: Evidence from GitHub Copilot*. Tutkimus. Haettu 11.10.2023 osoitteesta <https://arxiv.org/pdf/2302.06590.pdf>
- Pietikäinen, M.;& Silvén, O. (2019). *Tekoälyn haasteet : koneoppimisesta ja konenäöstä tunnetekoälyyn*. Oulu: Oulun yliopisto. Noudettu osoitteesta <http://jultika.oulu.fi/Record/isbn978-952-62-2482-4>
- Ramel, D. (31.10.2022). *Exploring the 'Almost Creepy' AI Engine in Visual Studio 2022*. Haettu 2.11.2023 osoitteesta Visual Studio Magazine: <https://visualstudiomagazine.com/articles/2022/10/31/vs-ai-engine.aspx>
- Rodriguez, M. (20.7.2023). *GitHub Copilot Chat beta now available for every organization*. Haettu 8.11.2023 osoitteesta Github Blog: <https://github.blog/2023-07-20-github-copilot-chat-beta-now-available-for-every-organization/>
- Roshelova, A. (27.3.2023). *Revolutionizing Development: Exploring AI-Powered Code Tools — Copilot and Tabnine*. Haettu 6.11.2023 osoitteesta Medium: <https://medium.com/@aroshelova.tech/revolutionizing-development-exploring-ai-powered-code-tools-copilot-and-tabnine-6e1a88f1a2d7>
- Stack Overflow. (2023). *Developer Survey 2023*. Haettu 2.11.2023 osoitteesta Stack Overflow Survey: <https://survey.stackoverflow.co/2023/>
- Stack Overflow. (14.6.2023). *Hype or not? AI's benefits for developers explored in the 2023 Developer Survey*. Haettu 11.10.2023 osoitteesta Stack Overflow: <https://stackoverflow.blog/2023/06/14/hype-or-not-developers-have-something-to-say-about-ai/?cb=1>
- Stack Overflow Labs. (3/2023). *Developer sentiment around AI/ML*. Haettu 11.10.2023 osoitteesta Stack Overflow: https://stackoverflow.co/labs/developer-sentiment-ai-ml/?utm_source=social&utm_medium=blog&utm_campaign=survey_ai
- Stanford University: Nilsson, Nils J. (2009). *The Quest for Artificial Intelligence - A History of Ideas And Achievements*. Cambridge University Press. Noudettu osoitteesta <https://ai.stanford.edu/~nilsson/QAI/qai.pdf>

- Tabnine. (2023). Haettu 2.11.2023 osoitteesta Tabnine: <https://www.tabnine.com/>
- Tabnine. (20.8.2023). *AI Code assistant: no-fluff guide to tech, features, and best practices*. Haettu 6.11.2023 osoitteesta Tabnine: <https://www.tabnine.com/blog/ai-code-assistant-guide/>
- The Metaphysics Research Lab, Department of Philosophy, Stanford University. (9.6.2006). *The Modern History of Computing*. Haettu 7.9.2023 osoitteesta Stanford Encyclopedia of Philosophy: <https://plato.stanford.edu/entries/computing-history/>
- The Metaphysics Research Lab, Department of Philosophy, Stanford University. (4.10.2021). *The Turing Test*. Haettu 18.4.2023 osoitteesta Stanford Encyclopedia of Philosophy: <https://plato.stanford.edu/entries/turing-test/>
- Tiernan, R. (25.10.2023). *Microsoft has over a million paying Github Copilot users: CEO Nadella*. Haettu 7.11.2023 osoitteesta ZDNET: <https://www.zdnet.com/article/microsoft-has-over-a-million-paying-github-copilot-users-ceo-nadella/>
- Turing, A. M. (1.10.1950). Computing Machinery and Intelligence. *Mind* (236), 433-460. Haettu 13.1.2024 osoitteesta <https://academic.oup.com/mind/article/LIX/236/433/986238>
- Vaithilingam, P.;Zhang, T.;& Glassman, E. (2022). *Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models*. Tutkimus. Haettu 12.10.2023 osoitteesta <https://par.nsf.gov/servlets/purl/10366304>
- Vaswani, A.;Shazeer, N.;Parmar, N.;Uszkoreit, J.;Jones, L.;Gomez, A. N.;. . . Polosukhin, I. (2017). *Attention Is All You Need*. Tutkimus. Haettu 2.10.2023 osoitteesta <https://arxiv.org/pdf/1706.03762.pdf>
- Wiio, O. A. (2008). Tietotekniikkaa viidellä vuosikymmenellä. *Tietokone* (14/2008). Haettu 13.1.2024 osoitteesta https://archive.ph/20130429225805/http://www.tietokone.fi/lehti/tietokone_14_2008/tietotekniikkaa_viidel_la_vuosikymmenella_553

LIITE 1: KYSELY GITHUB COPILOT -TYÖKALUN KÄYTTÖKOKEMUKSISTA

Github Copilot survey**Background questions:**

- Your programming experience in years? (round to closest)

Answer:

-Your experience with Github Copilot in months? (round to closest)

Answer:

-Did you use the Github Copilot's code completion feature?

- (0) No
- (1) Yes

Answer:

-Did you use the Github Copilot's Chat feature?

- (0) No
- (1) Yes

Answer:

Multiple-choice questions:

- Did you find using Github Copilot useful for your work?

- (0) No
- (1) Yes

Answer:

- Did using Github Copilot speed up writing code?

- (0) No
- (1) Yes

Answer:

- If you answered **(1) Yes** to the previous question, how much would you estimate that using Github Copilot would speed up writing code?

- (0) 0–20 %
- (1) 20–40 %
- (2) 40–60 %
- (3) 60–80 %
- (4) 80–100 %
- (5) 100+ %

Answer:

- Did using Github Copilot improve your productivity?

- (0) No
- (1) Yes

Answer:

-If you answered **(1) Yes** to the previous question, how much would you estimate that using Github Copilot would improve your productivity?

- (0) 0–20 %
- (1) 20–40 %
- (2) 40–60 %
- (3) 60–80 %
- (4) 80–100 %
- (5) 100+ %

Answer:

- How functional do you find the code generated by Github Copilot?

- (0) Very buggy
- (1) Buggy
- (2) I don't know
- (3) Functional
- (4) Very functional

Answer:

- How understandable do you find the code generated by Github Copilot?

- (0) Very difficult to understand
- (1) Difficult to understand
- (2) I don't know
- (3) Easy to understand
- (4) Very easy to understand

Answer:

- How complex do you find the code generated by Github Copilot?

- (0) Very complex
- (1) Complex
- (2) I don't know
- (3) Simple
- (4) Very simple

Answer:

Open questions:

- How did using Github Copilot change your work compared to before?

Answer:

-

- What do you think is the main/most important benefit of using Github Copilot?

Answer:

-

- What do you think needs the most improvement in Github Copilot?

Answer:

-