Ohjelmointitehtävien ratkaisu ChatGPT:llä ja sen vaikutukset ohjelmoinnin opetukseen

Pro gradu -tutkielma
Turun yliopisto
Tietotekniikan laitos
Tietojenkäsittelytiede
2024
Juuso Rytilahti

TURUN YLIOPISTO
Tietotekniikan laitos

Juuso Rytilahti: Ohjelmointitehtävien ratkaisu ChatGPT:llä ja sen vaikutukset ohjelmoinnin opetukseen

Pro gradu -tutkielma, 66 s., 2 liites.
Tietojenkäsittelytiede
Toukokuu 2024

Laajojen kielimallien yleistyminen ja niiden suorituskyvyn nousu muuttaa myös ohjelmoinnin opetusta. Tässä tutkielmassa perehdytään ChatGPT:n kyvykkyyteen suomenkielisissä ohjelmoinnin perusteiden tehtävissä ja siihen, miten se voi vaikuttaa ohjelmoinnin perusteiden opetukseen. Tämä tutkielma esittää ChatGPT:n kyvykkyyden ratkaista sanallisia ohjelmoinnin tehtäviä suomeksi Pythonilla. Tutkielmassa käsitellään myös sitä kuinka kuinka ChatGPT vaikuttaa ohjelmoinnin perusteiden opetukseen. Tutkimus tehtiin keväällä 2023.
Tutkimuksessa simuloitiin ChatGPT:tä käyttävää oppilasta. Siinä käsiteltiin kahta erilaista lähestymistapaa, kumpaakin testattiin GPT-3.5:llä ja GPT-4:llä. Ensimmäisessä lähestymistavassa simuloitiin oppilasta, joka omaa hieman perustietoja ohjelmoinnista. Toinen lähestymistapa simuloi täysin ohjelmoinnista tietämätöntä opiskelijaa. Tutkimuksessa saatu lopputulos vaihteli hieman valitun työkalun ja lähestymistavan mukaan, oppilaan saavuttaessa kurssin loppupisteistä 63.5%-86.2% riippuen valitusta lähestymistavasta. Työkalua hyödyntävä opiskelija olisi myös voinut läpäistä kaikki kolme kurssilla käytössä ollutta loppukoetta. On myös huomattava, että mikäli vain kurssin ohjelmointitehtävät otettiin huomioon, ChatGPT:llä avustettu oppilas olisi voinut vastata 107 (75.9%) tai jopa 139 (98.6%) kurssin 141 ohjelmointitehtävästä. Tutkielmassa tarkastellaan myös ChatGPT:n tekemiä yleisiä virheitä konkreettisten esimerkkien kautta.
Tämä tutkielma sisältää myös pohdintaa siitä, kuinka laajan kielimallin käyttöä huijaamiseen voitaisiin vaikeuttaa muokkaamalla ohjelmoinnin tehtäviä, ja kuinka haasteellista se on. Havainnollistavana esimerkkinä käytetään tehtävää, jonka ratkaisu ei vaadi suoraan ohjelmointia. Lisäksi tutkielmassa reflektoidaan laajojen kielimallien vaikutusta ohjelmoinnin opetukseen nyt ja tulevaisuudessa.

Asiasanat: ChatGPT, ohjelmointi, ohjelmoinnin opetus, laaja kielimalli

UNIVERSITY OF TURKU
Department of Computing

Juuso Rytilahti: Ohjelmointitehtävien ratkaisu ChatGPT:llä ja sen vaikutukset ohjelmoinnin opetukseen

Master of Science Thesis, 66 p., 2 app. p.
Computer Science
May 2024

The emergence of large language models and their raised capabilities also affects the teaching of programming. This thesis shows ChatGPT's performance and ability to solve verbal programming exercises written in Finnish with the programming language Python. More broadly, it is also explored how ChatGPT affects to the teaching of fundamentals of programming. The study was conducted during the Spring of 2023.

The research was conducted by simulating a student utilizing ChatGPT. We opted for two different approaches, which were tested with GPT-3.5 and GPT-4 models. The first approach simulated an amateur programmer with some prior programming knowledge. The second approach simulated a student without even a basic knowledge or understanding of programming. The performance varied, simulated student gaining between 63.5% and 86.2% of the course's total score. The tool-assisted students were also able to pass 3 different versions of the course's exam. Notably, if focused only on programming exercises, the simulated student could answer correctly 107 (75.9%) to 139 (98.6%) of the course's 141 programming exercises. The study also highlights the most common mistakes made by the models with concrete examples.

This thesis also includes a reflection on how one could try to make cheating with a large language model more difficult by modifying the assignments and how hard this is to achieve. An exercise, which solving does not require programming directly is used as an illustrative example. Additionally, the discussion section contains some general reflection on the impact of large language models on the teaching of programming now and in the future.

Keywords: ChatGPT, programming, programming education, large language model

Sisällys

Kuvat

Taulukot

1 Johdanto

Tekoälyn kehitys on ollut nopeaa parin viimeisen vuoden aikana. Laajat kielimallit ovat kehittyneet ratkaisemaan useita erilaisia tehtäviä eri sovellusaloilla. Laajojen kielimallien esiinmarssi tulee vaikuttamaan myös ohjelmointiin ja ohjelmoinnin opetukseen. ChatGPT:n suosio on kasvanut todella nopeasti ja sitä hyödynnetään laajasti eri toimialoilla.

Myös opiskelijat hyödyntävät laajoja kielimalleja. On siis tärkeää ymmärtää tämän teknologian tuomat mahdollisuudet ja rajoitukset. Avoin kysymys on myös kuinka laajojen kielimallien käyttö vaikuttaa opetukseen ja oppimiseen. Tämä tutkielma käsittelee tätä aihepiiriä pääasiallisesti keskittyen siihen, kuinka laajojen kielimallien kyvykkyys vaikuttaa opetukseen ja huijaamisen mahdollisuuksien lisääntymiseen ohjelmoinnin opetuksen kontekstissa.

Uusi teknologia tuo kuitenkin aina myös uusia uhkakuvia. Vaikka huijaamisen mahdollisuus kursseilla on aina ollut olemassa opiskelukaverilta kopioimisen ja luntaamisen muodossa, laajojen kielimallien yleinen saatavuus ja kyvykkyys saattavat laskea kynnystä huijaamiselle niiden käytön helppouden takia. Onkin tärkeää selvittää kuinka hyvin nykyiset laajat kielimallit (esim. ChatGPT) kykenevät suoriutumaan yksinkertaisista ohjelmoinnin perusteiden tehtävistä ja käydä läpi erilaisia mahdollisuuksia siitä, mihin suuntaan tehtäviä voitaisiin viedä laajojen kielimallien käytön vaikeuttamiseksi ja onko tälläinen lähestymistapa ylipäätään mahdollinen.

1.1 Tutkimuskysymykset

Tutkielma käsittelee ohjelmointitehtävien ratkaisua ChatGPT:llä ja laajojen kielimallien vaikusta ohjelmoinnin opetukseen. Tutkimuksessa simuloidaan ChatGPT:hen vahvasti tukeutuvaa oppilasta. Tutkimuskysykset on esitelty alla.

TK:1 Kykeneekö ChatGPT:tä hyödyntävä alkeelliset taidot omaava opiskelija läpäisemään ohjelmoinnin peruskurssin?

Ensimmäinen tutkimuskysymys keskittyy hieman ohjelmointia osaavan opiskelijan simulointiin. Tämän tutkimuskysymyksen tarkoituksena on selvittää kuinka helposti ohjelmoinnin perustaidot omaava opiskelija voi huijata kurssilla. Tutkimuksessa valittu näkökulma ChatGPT:tä hyödyntävän ohjelmoinnin alkeet osaavan opiskelijan simuloinnista pyrkii myös antamaan kontekstia laajojen kielimallien tuomiin opetukseen liittyviin haasteisiin ottamlla huomioon laajan kielimallin suorituskyvyn lisäksi myös inhimillisen elementin. Tämän tutkimuskysymyksen lopputulos auttaa myös mahdollisesti havaitsemaan laajojen kielimallien käyttöön ohjelmoinnin opetuksessa liittyviä mahdollisuuksia ja riskejä.

TK2: Voiko täysin tietämätön opiskelija läpäistä ohjelmoinnin peruskurssin osaamatta ollenkaan ohjelmoida?

Toinen tutkimuskysymys keskittyy täysin ohjelmoinnista tietämättömän opiskelijan simulointiin. Tämän tutkimuskysymyksen tarkoituksena on selvittää, kykeneekö täysin ohjelmointia osaamaton opiskelija läpäisemään ohjelmoinnin perusteet -kurssin. Käytännössä tämä tutkimuskysymys vastaa siihen, voiko opiskelija ohittaa kurssin suorittamiseen vaadittavat oppimistavoitteet täysin ja silti läpäistä kurssin. Opiskelijan kannalta motivaatio tälläiseen käytökseen voisi olla esimerkiksi halu saavuttaa opintopisteitä vähäisellä vaivalla. Opettajan tai kursseja tarjoavan tahon näkökulmasta tämä voi johtaa pitkällä aikavälillä sekä kurssin, että jopa kurssiin liittyvän

tutkinnon arvostuksen romahtamiseen.

TK3: Mitkä ovat ChatGPT:n tekemät yleisimmät virheet ohjelmoinnin perusteiden tehtävissä?

TK3 liittyy enemmän laajan kielimallin (tässä tapauksessa ChatGPT) tekemien virheiden analysointiin. Tarkoituksena on antaa kontekstia laajan kielimallin suorituskyvystä ja tunnistaa sen tekemät yleisimmät virheet. Kun yleisimmät laajan kielimallin tekemät virheet ovat tunnistettu, niistä kerättyä tietoa voitaisiin esimerkiksi käyttää tehtävänannon vaikeuttamiseen.

TK4: Kuinka helposti opiskelija voi huijata kursseilla ja onko mahdollista kiertää ChatGPT:n kykyä ratkaista tehtäviä esimerkiksi vaihtamalla tehtävätyyppiä tai kysymyksenasettelua?

TK4 keskittyy suoraan erilaisiin huijaamisen mahdollisuuksiin erityisesti ohjelmoinnin kursseilla. Samalla pyritään selvittämään, kuinka helposti laajojen kielimallien käyttöä voidaan vaikeuttaa, ja onko tämä tarkoituksenmukaista. Käytettyä kontekstia sanallisista ohjelmointitehtävistä pyritään myös laajentamaan havainnollistavalla esimerkillä, sillä ohjelmoinnin opetuksessa voi käyttää myös useita erilaisia tehtävätyyppejä sanallisen ohjelmistotehtävän lisäksi.

1.2 Tutkielman rakenne

Tutkielman luvussa 2 esitellään tutkielman ymmärtämiseen vaadittavia tekniikkaan liittyviä taustatietoja. Tämä luku sisältää perustiedot kone- ja syväoppimisesta, luonnollisen kielen käsittelystä ja laajoista kielimalleista. Laajojen kielimallien läpikäymisen jälkeen erityiseen tarkasteluun otetaan ChatGPT ja sitä edeltävät GPT-mallit.

Luvussa 3 käydään läpi läpi laajojen kielimallien käyttöä ohjelmoinnissa ja opetuksessa. Ennen laajoihin kielimalleihin siirtymistä luvussa myös esitellään ohjelmoinnin opetuksen järjestelmiä ja niissä käytettyjä menetelmiä. Näistä esitetään myös konkreettinen esimerkki (ViLLE). Tämän lisäksi luvussa sivutaan eettisen ohjeistuksen tärkeyttä, ja käydään läpi Gimpel et al. [1] luomaa opasta opettajille ja opiskelijoille tekoälyn eettisestä hyödyntämisestä opetuksen kontekstissa. Lopuksi käydään läpi laajalla kielimallilla tuotetun tekstin tunnistamiseen liittyviä haasteita.

Luvussa 4 kerrotaan tutkimusasetelma. Aluksi kerrotaan valittu tutkimusmenetelmä, ja sen jälkeen käydään lyhyesti läpi tutkimuksessa käytetty tietoaineisto. Lopuksi kuvataan tutkimuksessa käytetty metodologia. Tutkimuksen tulokset käydään läpi luvussa 5. Luvussa käydään aluksi läpi simuloitujen oppilaiden tulokset ja sen jälkeen analysoidaan ChatGPT:n antamissa vastauksissa olevia virheitä. Tutkimuksen johtopäätökset ja tutkimukseen liittyvää pohdintaa käydään läpi luvussa 6. Tämän lisäksi luvussa tarkastellaan huijaamisen vaikeuttamista erilaisin keinoin. Osana tehtyä pohdintaa esitetään myös havainnollistava esimerkki. Luvussa 7 tehdään lyhyt yhteenveto tutkimuksesta ja vastataan lyhyesti tässä tutkielmassa esiteltyihin tutkimuskysyksiin.

1.3 ChatGPT:n käyttö

ChatGPT:tä on käytetty tässä tutkielmassa useissa erilaisissa tehtävissä. Esimerkiksi kerätty tutkimusaineisto on käytännössä kooste ChatGPT:n vastauksia ohjelmoinnin perusteiden tehtäviin. Samoin ChatGPT:tä on käytetty joissakin muissa tässä tutkielmassa olevista esimerkeistä. Kaikki ChatGPT:n luomat vastaukset ovat pyöreäkulmaisessa laatikossa harmaalla taustalla. ChatGPT:tä ei ole käytetty tutkielman asiatekstin tuottamisessa tai muokkaamisessa.

Tämän lisäksi tässä tutkielmassa ChatGPT:tä on käytetty datan analysoinnissa

ja LaTeX-tyylien luomisessa. Datan analysointiin käytetyn ohjelmakoodin generoimisessa käytettiin apuna ChatGPT:tä. ChatGPT:lle annettiin kehotteessa ohjeet kuinka verrata excel-tiedostoon kerättyjä tietoja ja minkälaisia kuvaajia näistä pitäisi piirtää. ChatGPT:n luoma ohjelmakoodi tarkistettiin kirjoittajan toimesta ja siihen tehtiin tarvittaessa pieniä muutoksia. LaTeX-tyylien luomisessa ChatGPT:tä on käytetty tunnistamaan mahdollisia kirjastoja ja ja tyylitykseen liittyvien ongelmien korjaamisessa. Samoin osa määritellyistä tyyleistä ovat ChatGPT:n avulla generoituja.

2 Laajat kielimallit ja niiden teknologista taustaa

Tässä luvussa käsitellään laajoissa kielimalleissa hyödynnettävien teknologioiden taustoja. Jotta kieltä voidaan käsitellä koneellisesti, se pitää muokata koneelle ymmärrettevään muotoon. Esimerkiksi sanojen välisiä yhteyksiä on mahdollista mallintaa. Tätä osa-aluetta kutsutaan luonnollisen kielen käsittelyksi, ja sen perusteita käydään tarkemmin läpi luvussa 2.2. Laajat kielimallit ovat syväoppimisen osa-alue. Syväppiminen on puolestaan koneoppimisen osa-alue. Kone- ja syväoppimisen perusteita käydään tarkemmin läpi luvussa 2.1.

Teknologian taustan käymisen jälkeen käydään läpi laajoja kielimalleja ja niihin liittyvää termistöä Tämän jälkeen käsitellään eri GPT-malleja kronologisessa järjestyksessä.

2.1 Koneoppiminen ja syväoppimisen menetelmät

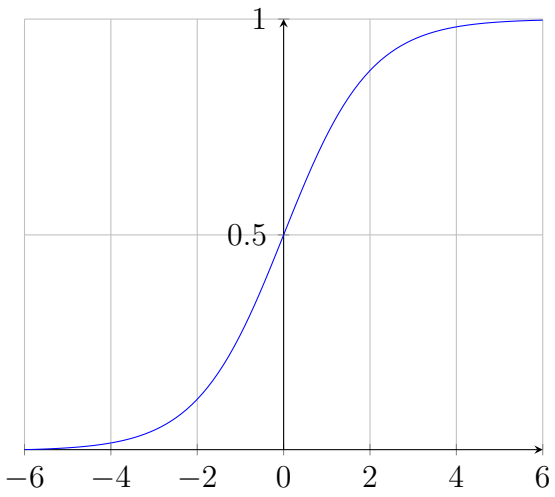
Koneoppimisella tarkoitetaan oppimisalgoritmeihin pohjautuvia malleja, joita koulutetaan tunnistamaan aineistosta tiettyjä ilmiöitä käyttämällä dataa. Koneoppimisen ideana on kouluttaa malli tunnistamaan koulutusdataa yhdistäviä piirteitä ja käyttää tätä mallia tunnistamaan samoja piirteitä uudesta datasta. Koneoppimista voidaan soveltaa useilla eri alueilla esimerkiksi kuvien luokittelussa, tekstinkäsittelyssä tai piirilevyjen suunnittelussa. [2]

Koneoppimisalgoritmeja on olemassa useita erilaisia. Algoritmien vahvuukset ja heikkoukset poikkeavat usein toisistaan. On tärkeää valita oikea malli, mutta myös mallin parametrien säätö on tärkeä osa-alue kun tavoitellaan koulutettavalle mallille parempaa suorituskykyä. Sisäisten parametrien muokkaus voi vaikuttaa huomattavasti mallin suorityskykyyn. [2]

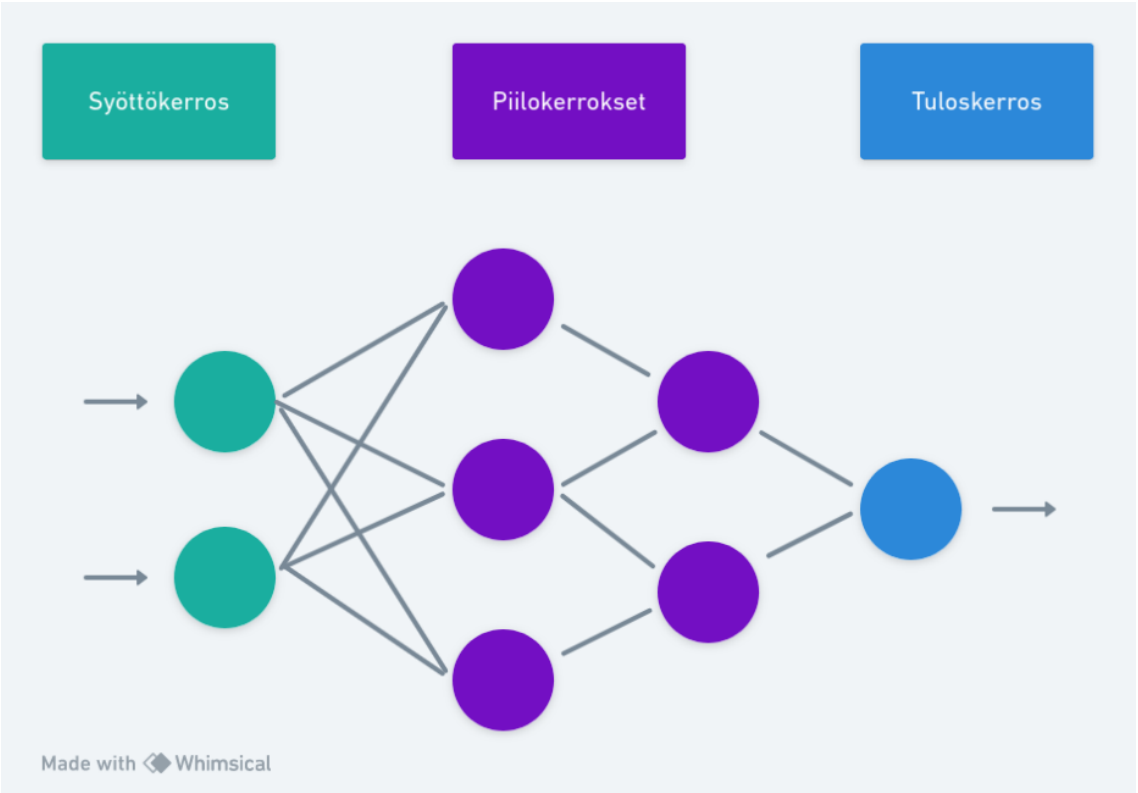
Syväoppiminen voidaan nähdä koneoppimisen alakategoriana. Syväoppimisella viitataan yleensä useita kerroksia sisältäviin keinotekoisiin malleihin. Nämä kerrokset koostuvat yksittäisistä moduuleista, jotka aktivoituvat perustuen aiemman kerroksen moduuleihin. Syväoppimista hyödyntävät mallit ottavat alkuperäisen datan vastaan syöttökerrroksessa ja muuttavat sen abstraktimpaan muotoon kerros kerrokselta. Mikäli datan määrä ja näiden kerroksien määrä on riittävän suuri, voidaan saavuttaa hyvä suorituskyky. Syväoppisen mallit ovat usein perinteisiä koneoppimismalleja tehokkaampia abstraktoimaan tietoa ja sen takia syväoppimismenetelmät voivat usein saavuttaa perinteisiä koneppimismalleja paremman suorituskyvyn. Syväoppimisen malleja voidaan hyödyntää useissa erilaisissa tehtävissä esimerkiksi kuvantunnistamisessa tai tekstin luokittelussa. [3]

Useimmat syväoppimisen mallit ovat neuroverkkoja. Neuroverkot koostuvat useista eri kerroksista ja näiden kerrosten sisällä on neuroneita. Neuroneiden väliset yhdeydet kutsutaan synapseiksi. Aktivaatiofunktiona toimii useimmiten sigmoid-funktio. Esimerkki Sigmoid-funktiosta voidaan nähdä kuvassa 2.1.

Yksinkertaisimmillaan neuroverkon voidaan ajatella koostuvan syöttökerroksesta, yhdestä piilokerroksesta ja tuloskerroksesta. Se, mikä neuroni aktivoituu määräytyy aiemman kerroksen tilan perusteella. [4] Abstrakti esimerkki hieman monimutkaisemman monikerroksisesta neuroverkon rakenteesta voidaan nähdä kuvassa 2.2.



Kuva 2.1: Sigmoid-funktion kuvaaja.



Kuva 2.2: Kuvaus monikerroksisen neuroverkon rakenteesta.

2.2 Luonnollisen kielen käsittely

Luonnollisen kielen käsittely on yksi data-analytiikan osa-alueista. Erityisesti internetin yleistymisen myötä tarjolla on ennennäkemätön määrä dataa. Tämän datan hyödyntäminen on ollut haastavaa, sillä suuren datamäärän käsittely vaatii paljon laskentatehoa. Kuitenkin tekniikan kehittyessä on tullut mahdolliseksi kouluttaa koneoppimiseen liittyviä malleja entistä suuremmilla data määrillä, ja laajat kielimallit ovatkin hyvä esimerkki siitä, mitä teknologian kehittyminen mahdollistaa koneoppisen kontekstissa.

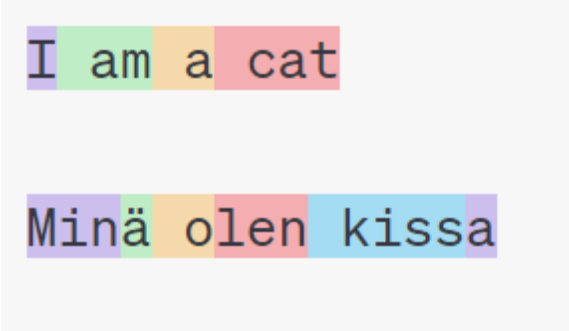
Tekstin mallintaminen koneoppimiseen sopivaan muotoon voidaan tehdä muuttamalla teksti numeroiksi. Tämä voidaan tehdä, esimerkiksi niin, että jokainen käytetyssä aineistoissa esitetty sanan esiintyvyys eri dokumenteissa samassa korpuksessa (korpus tarkoittaa kokoelmaa dokumentteja) on merkitty binäärisesti exceltiedostoon. [5] Esimerkki tästä nähdään taulukossa 2.1. Kun data on saatu siirrettyä taulukkoon koneen ymmärtämässä muodossa, niin tekstin analysointi onnistuu data-analytiikassa yleisesti käytösssä olevien keinojen avulla.

Eläin	**Kissa**	**Koira**	**lääkäri**
1	1	0	0
1	0	1	1
1	1	1	1

Taulukko 2.1: Esimerkki taulukko binäärisesti merkatuista sanoista dokumenteissa. Ensimmäisessä dokumentissa esiintyvät sanat *eläin* ja *kissa*, mutta eivät sanat *koira* tai *lääkäri*.

Eräs tärkeä luonnollisen kielen käsittelyn osa-alue on tokenisaatio. Tokenisaatiolla tarkoitetaan tekstin hajoittamista pienempiin osiin, *tokeneihin*. Tokenit voivat esi-

merkiksi olla yksittäisiä sanoja. Haasteellisia tehokkaan tokenisaation kannalta ovat esimerkiksi merkit, joita käytetään useissa eri yhteyksissä. Esimerkiksi pistettä (.) käytetään useissa erilaisissa käyttötarkoituksissa riippuen kontekstista. Sen käyttö voi esimerkiksi merkata sanan tai lauseen loppua, mutta samoin sitä voidaan myös käyttää numeroiden välissä desimaalin erottajana. Mikäli tilanne olisi tämä, olisi järkevintä laskea koko yksittäinen sana tokeniksi. Lisähuomautuksena välilyöntejä, rivinvaihtoja ja muita sanojen välissä olevia merkkejä ei pidetä tokeneina. Yleisesti käytetty yhteinen kollektiivinen nimitys näille merkeille on *tulostumaton merkki* (eng.white space). Tokeneiden muodostamiseen käytettyjä sääntöjä voidaan myös muokata sopimaan paremmin käytettyyn aineistoon. [6] Käytännön esimerkki tokenisaatiosta voidaan nähdä kuvassa 2.3.

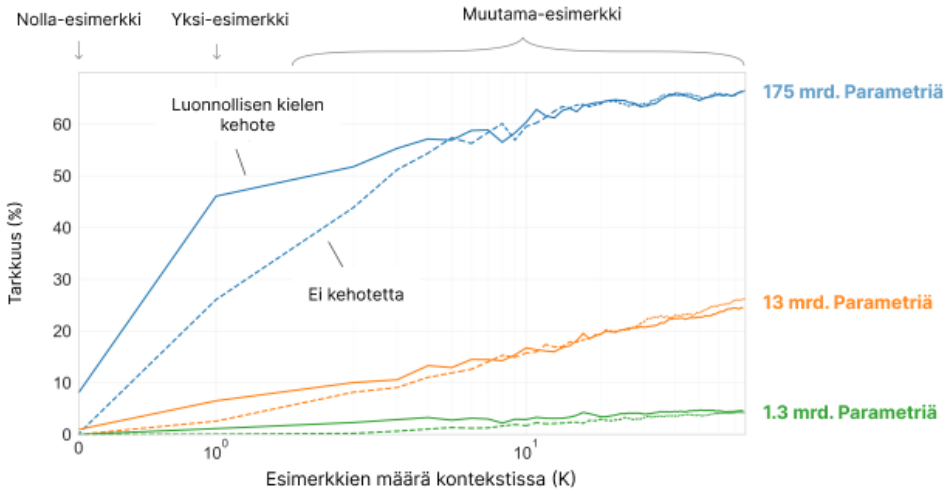


Kuva 2.3: Esimerkki tokenisaatiosta suomen ja englannin kielellä. Huomaa, että kuvassa esitelty tokenisaatio on optimoitu englannin kielelle (englanninkielisessä lauseessa tokenit ovat yksittäisiä sanoja, suomenkielisessä taas yksittäiset sanat koostuvat useammasta eri tokenista).

2.3 Laajat kielimallit

Mallit, jotka pyrkivät ennustamaan seuraavan sanan tai tokenin käyttämällä todennäköisyysjakaumaa, kutsutaan kielimalleiksi. Laajat kielimallit esikoulutetaan laajalla aineistolla, joista mallit oppivat sanojen merkityksiä sanojen keskinäisten riippuvuussuhteiden kautta. Useimmat nykyiset laajat kielimallit hyödyntävät transformer-arkkitehtuuria. [7]

Laajoissa kielimalleissa on vahvaa näyttöä siitä, että mitä enemmän parametrejä ja mitä suurempi on mallin kouluttamiseen käytetyn datan (tokeneiden) määrä, sitä parempi sen suoriutumiskyky on [8]. Parametrit ovat mallin sisäisiä, muokattavia lukuja joita voidaan sanoa myös painoiksi [3]. Tom B. Brown et. al huomauttavat GPT-3 mallin esittelevässä artikelissa [9], että useimmat kielimallit vaativat tehtävään liittyviä erikoistuneita tietojoukkoja (eng. data set). Laajat kielimallit pyrkivät ratkaisemaan tämän ongelman hyvin suurella mallin harjoitteluaineistoilla. Havainnollistava esimerkki tästä on kuvassa 2.4. Samoin esimerkiksi GPT-3 mallin suoriutuminen vaikuttaa myös parantuvan mitä suurempi malli on kyseessä.



Kuva 2.4: Brown et al. [9] tekemä vertailu erikokoisilla GPT-3 malleilla siitä, kuinka mallin parametrien määrä vaikuttaa laajan kielimallin kykyyn oppia tarjottavista esimerkeistä vastauksia.

Transformerit

Transformer-arkkitehtuurin esittelivät Vaswani et al. [10] vuonna 2017. Aiemmista malleista poiketen heidän kehittämänsä malli hylkäsi kokonaan takaisinkytketyn neuroverkon (eng. recurrent neural network) ja sen sijaan hyödynsi itsehuomiomekanismia (eng. self-attention mechnism). Tämä lähestymistapa mahdollisti te-

hokkaamman samanaikaisen ajon mallin koulutuksessa. Heidän mukaansa "mallin hyödyntävä arkkitehtuuri on laskennallisesti nopeampaa kun sekvenssin koko n on suurempi kuin esitysulottuvuus d".

Ensimmäinen transformer-arkkitehtuuria hyödyntävä malli päihitti aiemmin kehityt mallit tekstien koneellisessa kääntämisessä englannista saksaksi ja englannista ranskaksi. Vertailussa käytettiin WMT 2014 standardia tietoaineistoa. On myös huomattava, että suorituskykyisen mallin kouluttamiseen tarvittiin myös suhteellisen vähän resursseja (8kpl P100-näytönohjaimia 3.5 päivää kestävällä koulutuksella). [10]

Vaikkakin transformer-arkkitehtuuria hyödynnettiin myöhemmin ChatGPT:ssä, sitä on käytetty myös muiden laajojen kielimallien kehittämisessä. Devlin et al. [11] kehittivät kielimallin BERT (Bidirectional Encoder Representations from Transformers). BERT:tiä kouluttaessa mallilta peitettiin n. 15% satunnaisesti valittuja tokeneja jokaisesta sekvenssistä ja malli yritti ennustaa siltä peitetyt tokenit.

2.4 Laajojen kielimallien käyttöön ja kouluttamiseen liittyviä haasteita ja ongelmia

Yleisesti suurien kielimallien haasteen pidetään sitä, että sen tuottamaan tekstiin ei voi täysin luottaa. Laajan kielimallin tuottama teksti voi sisältää *hallusinaatioita* [12]. Hallusinaatioilla kuvaillaan tekoälyn tuottamaa tekstiä, joka voi olla sisältää esimerkiksi virheellistä tietoa. Tekoälyn tuottamat ohjeet voivat myös olla esimerkiksi vaarallisia tai sen tuottama teksti voi olla toksista. Hallusinoinnin tunnistamista vaikeuttaa se, että laajojen kielimallien tuottama teksti on usein myös mallin hallusinoidessa sujuvaa. Tekstin vakuuttava kirjoistustyyli siis saattaa hämätä tekstiä lukevaa ihmistä pitämään tuotettua tekstiä luotettavana. Tämän lisäksi eräs ongelma on se, että laajan kielimallin tuottama vastaus ei saata vastata sille kehotteen

antaneen ihmisen aietta. Tämän ongelman ratkaisuksi ei ole tarjolla triviaaleja ratkaisuja, mutta siihen on esitetty useita erilaisia mahdollisia lähtökohtia. Esimerkiksi Ouyang et al. [13] esittivät ratkaisuksi mallien hienosäädön ihmisiltä kerätyn palautteen perusteella. He myös kehittivät ihmisiltä kerättyä palautetta hyödyntävä mallin nimeltä InstructGPT. InstructGPT:tä käsitellään tarkemmin luvussa 2.8.

Laajojen kielimallien käyttöön liittyy myös hyvin paljon avoimia lainopillisia ja eettisiä kysymyksiä. Laajat kielimallit hyödyntävät hyvin laajaa määrää dataa. Useimmiten suurin osa koulutusdatasta on kerätty internetistä automaattisilla hakuroboteilla (eng. web crawler).

Laajojen kielimallien käytössä on paljon ratkaisemattomia kysymyksiä, joihin haetaan vastausta akateemisessa maailmassa ja sen ulkopuolella. Tällä hetkellä ympäri maailmaa useat toisistaan riippumattomat tahot ovat käynnistäneet oikeudenkäyntejä generatiivisia tekoälyä kehittäviä yrityksiä mm. Alphabetia (mm. Googlea hallinnoiva monialakonserni) ja organisaatioita mm. OpenAI:ta vastaan.[1]

On myös epäselvää, missä laajuudesssa esimerkiksi erilaiset kuvapankit, kuten Shutterstock[2] tai Getty Images[3], voivat hyödyntää hallussaan olevia kuvia tekoälymallien luomiseen ilman erillistä lupaa. Eri palvelut myös pyrkivät erilaisiin ratkaisuihin. Esimerkiksi Getty Images on kieltänyt kokonaan tekoälyllä luotujen kuvien myymisen alustallaan [4]. Toisenlaisen lähetymistavan esimerkiksi mainittakoon Shutterstock. Shutterstock on kieltänyt lisenssihuolien vuoksi kolmansia osapuoli myymästä sivustollaan tekoälyllä generoitua sisältöä, mutta myy kuitenkin itse tekoälyn avulla tuotettuja kuvia[5].

[1] https://www.reuters.com/legal/litigation/google-hit-with-class-action-lawsuit-over-ai-data-scraping-2023-07-11/

[2] https://www.shutterstock.com/fi/

[3] https://www.gettyimages.fi/

[4] https://www.theverge.com/2022/9/21/23364696/getty-images-ai-ban-generated-artwork-illustration-copyright?scrolla=5eb6d68b7fedc32c19ef33b4

[5] https://www.theverge.com/2022/10/25/23422359/shutterstock-ai-generated-art-openai-dall-e-partnership-contributors-fund-reimbursement

2.5 Laajojen kielimallien suoriutumiseen vaikuttavia tekijöitä

Laajojen kielimallien suoritumiseen vaikuttavat useat useat eri seikat. Yleisesti kielimallien suoriutumiseen vaikuttaa suuresti kielimallille annettu syöte. Jopa sanojen ja lauseiden järjestys vaikuttaa mallin tuottamaan lopputulokseen. Hyvän kehotteen tulisi olla selkeä, kieliopillisesti hyvä, sisältää kontekstia esitetylle pyynnölle tai kysymyksella, määritellä kehotteessa esitettävä ongelma mahdollisimman tarkasti, sisältää vastauksen toivotun pituuden ja siis yleisesti olla mahdollisen tarkka kuvaus siitä, minkälainen vastaus kielimallilta halutaan.[6] Samoin, kuten aiemmin mainittu luvussa 3.5, myös asetettu lämpötila ja muut mallin asetukset vaikuttavat mallin tuottamaan vastauksen laatuun. Brown et al. [9] määrittelevät joitakin termejä laajoihin kielimalleihin liittyen. Nämä määritelmät löytyvät alla olevasta listauksesta.

- **Hienosäätö** (eng. Fine-Tuning): Hienosäädössä valmiiksi koulutetun mallin painoja hienosäädetään kouluttamalla sitä normaalisti muutamalla tuhannella esimerkillä.
- **Nolla-esimerkki** (eng. Zero-Shot): Mallille annetaan pelkästään luonnollisella kielellä kirjoitettu kuvaus, jonka perusteella se tarjoaa vastauksen kehotteeseen.
- **Yksi-esimerkki** (eng. One-Shot): Mallille annetaan yksi konkreettinen esimerkki luonnollisen kuvauksen lisäksi.
- **Muutama-esimerkki** (eng. Few-Shot): Mallille annetaan muutama esimerkki luonnollisen kuvauksen lisäksi. Mallin painoja ei muuteta.

[6] `https://platform.openai.com/docs/guides/gpt-best-practices/strategy-write-clear-instructions`

Kuitenkin myöhemmin on löydetty myös muita tapoja parantaa kielimallin suoriutumista erilaisista tehtävistä. Näitä ovat esimerkiksi Wei et al. [14] esittelemä tekniikka *ajatusketju* (eng. *Chain-of-Thought* ja Yao et al. [15] kehittämä tekniikka *ajatuspuu* (eng. *Tree of Thoughts*). Näitä tekniikoita ja niiden rajoituksia käydään tarkemmin läpi kohdissa 2.5 ja 2.5.

Ajatusketju

Ajatusketjun idea on saada laaja kielimalli tuottamaan myös vastaukseen tarvittavat väliaskeleet. Tämä vaikuttaa vähentävän kielimallin tekemiä virheitä ts. parantavan mallin suorituskykyä. Väliaskeleiden tuottaminen myös mahdollistaa kielimallin logiikan seuraamisen. Tätä voidaan hyödyntää muutama-esimerkki -tekniikan antamassa kontekstissa. [14] Konkreettinen esimerkki ajatusketjusta nähtävissä alla kuvassa 2.5.

Käytännön esimerkki ajatusketjun hyödyntämisestä

Normaali kehote

Laajalle kielimallille annettu kehote

K: Rogerilla on 5 tennispalloa. Hän ostaa 2 pakkausta lisää tennispalloja. Jokaisessa pakkauksessa on 3 tennispalloa. Kuinka monta tennispalloa hänellä on nyt?

V:Vastaus on 11.

K: Kahvilassa on 23 omenaa. Jos kahvilan työntekijät käyttivät 20 omenaa tehdäkseen lounasta ja ostivat 6 omenaa lisää, kuinka monta omenaa heillä on?

Laajan kielimallin vastaus

V: Vastaus on on 27.

Ajatusketjua hyödyntävä kehote

Laajalle kielimallille annettu kehote

K: Rogerilla on 5 tennispalloa. Hän ostaa 2 pakkausta lisää tennispalloja. Jokaisessa pakkauksessa on 3 tennispalloa. Kuinka monta tennispalloa hänellä on nyt?

V:Roger aloitti 5 pallolla. 2 kolmen kappaleen pakkauksessa on yhteensä 6 palloa. 5 + 6=11. Vastaus on 11

K: Kahvilassa on 23 omenaa. Jos kahvilan työntekijät käyttivät 20 omenaa tehdäkseen lounasta ja ostivat 6 omenaa lisää, kuinka monta omenaa heillä on?

Laajan kielimallin vastaus

V: Kahvilassa oli alunperin 23 omenaa. He käyttivät 20 tehdäkseen lounaan. Joten heillä oli sen jälkeen 23 - 20 = 3. He ostivat sen jälkeen 6 omenaa lisää, joten heillä on 3 + 6 = 9. Vastaus on siis 9.

Kuva 2.5: Wein et al. [14] esittelemä käytännön esimerkki ajatusketjun toimivuudesta käännettynä suomeksi. Kehote hyödyntää muutama-esimerkki -tekniikkaa.

Ajatuspuu

Yao et al. [15] esittelemä ajatuspuu generalisoi ajatusketjussa esitellyn lähestymistavan. Tarkempi kuvaus ajatuspuun prosessin kulusta nähdään kuvassa 2.6. Ajatuspuu toimintamalli jakautuu neljään eri vaiheeseen. Toimintamallin tämänhetkistä tilannetta kuvataan tilaksi (eng. state). Tila sisältää alkuperäisen ratkaistavan ongelman, uuden ajatuksen ja siihen ajatukseen johtaneet ajatukset.

1. **Ongelman hajoittaminen erillisiksi ajatuksiksi**

   Ajatuspuun määritelmä lähtee useista erillisistä ajatuksista. Nämä ajatukset ovat laajan kielimallin tuottamia vastauksia. Yksi ajatus voi olla esimerkiksi yhtälö, kirjoitussuunnitelma tai sisältää vain muutaman sanan. Yao et al. [15] ovat käyttäneet omassa tutkimuksessaan havainnollistavina esimerkkeinä pientä 5 x 5 sanaristikkoa, *lopputulos 24*-peliä ja luovan kirjoittamiseen liittyvää alustavan suunnitelman luomista. He nostavat esille, että tällä menetelmällä ratkaistava ongelma voi olla mikä vain, kunhan kontekstin aiheuttama kokorajoitus otetaan huomioon.

2. **Ajatuksien luominen**

   Ajatuksien luomiseen he ehdottavat kahta erilaista lähestymistapaa. Lopullinen valinta näiden lähtökohtien välillä riippuu ratkaistavasta ongelmasta.

   (a) **Esimerkistä** Ajatuksien luonti esimerkin kautta sopii ongelmille, joiden ongelma-avaruus on laaja. Yao et al. [15] käyttivät tästä esimerkkinä luovaa kirjoittamista.

   (b) **Äänestys**. Äänestys-malli sopii ongelmille, joiden ongelma-avaruus on pieni tai kohtuullisen rajattu. Ajatukset (ongelman osittaiset ratkaisut) arvioidaan ja parhaaksi arvioitu ajatus annetaan seuraavaksi kehotteeksi.

3. **Tilan arvioija** Tilan arvioija toimii heurestiikkana etsimiseen käytetylle algoritmille, joka päättää mikä tai mitkä ajatukset tutkitaan seuraavaksi ja se myös

määrittää seuraavana tutkittavien ajatusten järjestyksen. Yao et al. [15] esittämässä lähestymistavassa laaja kielimalli perustelee, mikä esitetyistä uusista tiloista on paras vaihtoehto. He ehdottavat tilan arviointiin kahta mahdollista lähestymistapaa

(a) **Arviointi** Nykyinen tila annetaan arviointikehotteelle, joka arvioi mahdolliset tulevaisuuden tilat. Tämä arviointi voi olla esimerkiksi lukuarvo väliltä 1-10. Parhaan arvioinnin saanut tila valitaan nykyiseksi tilaksi, ja arviointi-kehotteelle annetaan arvioitaviksi seuraavat mahdolliset tilat. Arviointikehotteen tuottama arviointi ei ole täydellinen, mutta sen tuottama alustava arviointi on riittävä.

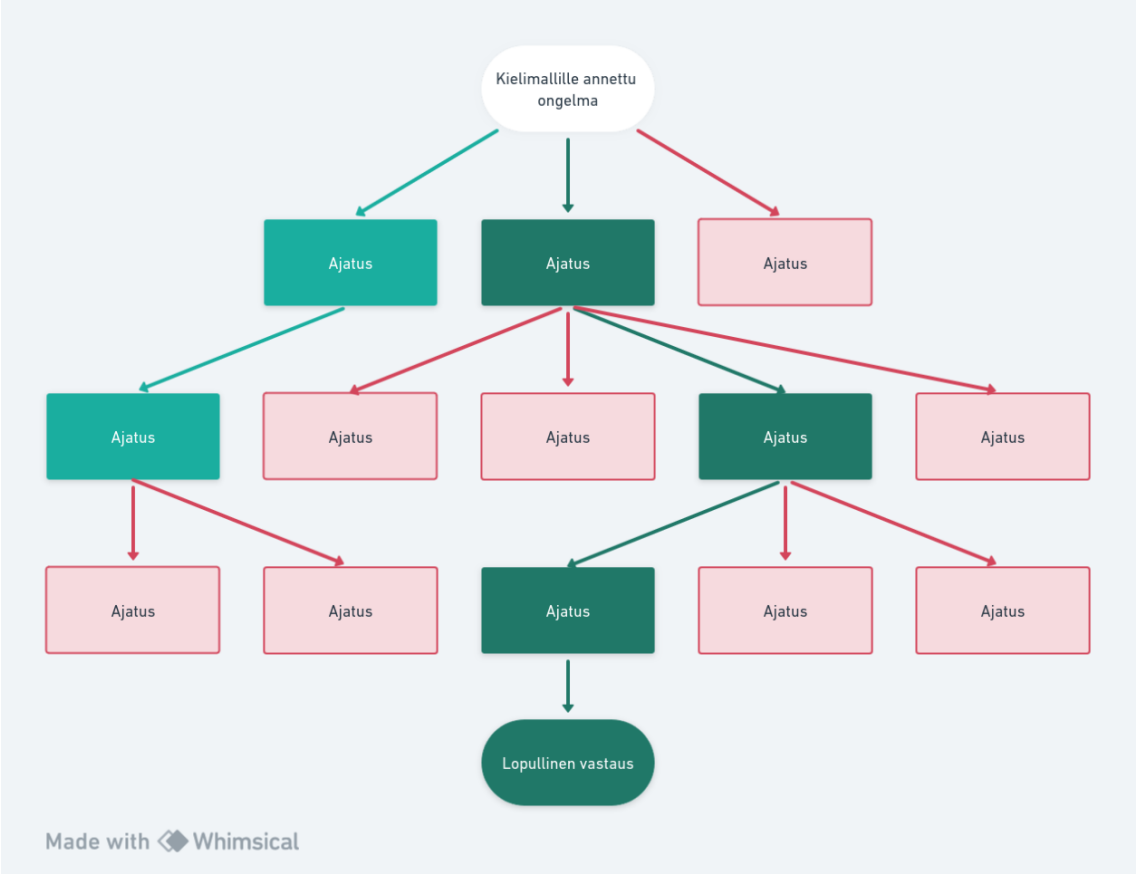
(b) **Äänestys** Äänestys käydään mahdollisten seuraavien tilojen välillä. Tässä mallissa malli äänestää parhaan ratkaisun puolesta.

4. **Hakualgoritmi** Viimeinen vaihe ajatuspuussa on käydä ajatuksia läpi hakualgoritmin avulla. Heidän tekemässään tutkimuksessa käsiteltiin vain kahta yksinkertaista algoritmia, jotka ovat esitelty alla.

(a) **Leveyshaku (BFS)** algoritmi pitää muistissaan ennalta sovitun määrän parhaimman tuloksen äänestyksessä saaneita lähestymistapoja. Yao et al. [15] pitivät esittämissään esimerkeissä puun syvyyden kolmessa tai alle ja parhaiden tilojen määrän korkeimmillaan viidessä.

(b) **Syvyyshaku (DFS**) Syvyyshaussa tutkitaan aluksi kaikista lupaavinta tilaa. Mikäli tilan arvioija päättää päättää, että haluttu lopputulos on saavutettu tai sitä on mahdotonta saavuttaa nykyistä tilaa jatkamalla, algoritmi palaa takaisin aiempaan tilaan.

Tämän lisäksi Hubert [16] on ehdottanut tämän lähestymistavan suoraviivaistamista yhteen kehotteeseen, esimerkki tästä on nähtävissä kuvassa 2.7.



Kuva 2.6: Yao et al. [15] esittelemän ajatuspuun abstrakti rakenne..

Hubertin esittämä variaatio ajatuspuusta

Kuvittele kolme erilaista asiantuntijaa vastaamassa tähän kysymykseen. Kaikki asiantuntijat kirjoittavat ylös 1 askeleen heidän ajatuksistaan ja jakavat sen ryhmänsä kanssa. Sen jälkeen kaikki asiantuntijat esittävät seuraavan askeleen.

Jos joku asiantuntijoista tajuaa olevansa väärässä, hän poistuu keskustelusta.

Kysymys on...

Kuva 2.7: Hubertin esittämä yhden kehotteen variaatio ajatuspuusta. Esimerkki on suora käännös Hubertin esittämästä esimerkistä [16].

2.6 GPT ja GPT-2

Radford et al. [17] kehittivät alkuperäisen GPT-mallin. Se hyödyntää transformer-arkkitehtuuria, ja sen kontekstin koko oli suurimmillaan 512 tokenia. Mallin kouluttamiseen käytettiin BooksCorpus -tietoaineistoa [18], joka koostuu yli 7000 kirjasta. Se kykeni suorittamaan useita erilaisia tehtäviä, esimerkiksi tekstin luokittelua, samankaltaisuuden arviointia, monivalintojen luomista. Sen suorituskykyä mitattiin myös antamalla sille ns. maalaisjärkeä sisältäviä ongelmia. He hienosäätivät mallin parametrit jokaiseen mainittuun tehtävään erikseen.

GPT-2 arkkitehtuuri oli samankaltainen GPT-1 kanssa joillakin pienillä muutoksilla. Yksi merkittävimmistä muutoksista oli mallin koko sen edeltäjään verrattuna. Isoimmassa GPT-2 mallissa on yli 1,5 miljardia parametria. Samalla sen käyttämä sanasto laajeni hieman yli 50 000 sanaan aiemman mallin käyttämästä 40 000 sanan sanastosta. Samalla myös kontekstin koko tuplaantui 1024 tokeniin. [19]

2.7 GPT-3

GPT-3 on autoregressiivinen transformer -kielimalli. Se jakaa samanlaisen arkkitehtuurin GPT-2 mallin kanssa. GPT-3 malllilla viitataan malleista laajimpaan, 175 miljardin parametrin malliin. Malli oli kymmenen kertaa suurempi kuin yksikään sitä edeltänyt laaja kielimalli. Tämä myös aiheutti sen, että mallin suorituskyky oli huomattavasti edeltäviä malleja laajempi useissa erilaisissa luonnollisen kielen tehtävisssää, esimerkiksi kääntämisessä, kysymyksiin vastamaisessa, ja yksinkertaisissa aritmeettisissa operaatioissa. [9]

GPT-3 mallin kouluttamiseen käytetyn tietoaineiston määrä on huomattava. GPT-3 harjoittamiseen käytetty tietoaineisto koostuu muokatusta versiosta Common Crawl -tietoaineistosta [20], laajennetun version Kaplanin [8] alunperin luomasta WebText-tietoaineistosta [19], kaksi kirjoista koottua tietoaineistoa ja englannin

kielellä kirjoitetun Wikipedian. Brown et al. [9] huomasivat, että Common Crawl -tietoaineiston siistiminen ja korkeampilaatuisen datan painottaminen johtivat parempaan lopputulokseen mallin suorituskyvyssä.

2.8 InstructGPT

InstructGPT on OpenAI:n kehittämä laaja kielimalli, joka perustuu GPT-3 malliin [13]. Se jaa saman harjoitteluaineiston ja rakenteen kuin GPT-3. Ainoa ero näiden mallien välillä on, että InstructGPT:n malli ovat hienosäädetty ihmisten palautteen avulla. Ouyang et al. [13] mukaan InstructGPT:n tavoite oli saada laaja kielimalli seuraamaan käyttäjän aikomusta ja ohjeita pelkän seuraavan sanan ennustamisen sijaan. Tämä lähetysmistapa laajan kielimallin tehokkaaseen hyödymtämiseen vaikuttaa myös toimivan, sillä InstructGPT noudatti ohjeita myös aineiston kanssa, johon se ei juurikaan saanut ohjeistusta ts. malli osoitti pientä kykyä generalisoida ohjeiden seuraamisen.

Mallilla on GPT-3 verrattuna useita parannuksia. Mallin esitelleessä tutkimuksessa ihmisille annettiin arvioitaviksi InstructGPT:n ja GPT-3:n tuottamia vastauksia. Arvioijat pitivät yleisesti ottaen InstructGPT:n 1.3 miljardin parametrin kokoisen mallin tuottamia vastauksia parempina kuin huomattavasti suuremman, 175 miljardilla parametrilla harjoitetun GPT-3-mallin tuottamia vastauksia. He myös testasivat mallia tietoaineistolla RealToxicityPrompts [21]. Kun mallille annettiin ohjeeksi olla kunnioitettava, se tuotti n. 25% vähemmän toksisia vastauksia kuin GPT-3. Kuitenkin mallin vastauksissa esiintyvät vinoumat olivat samalla tasolla kuin aiempien mallien. Malli siis toisti sen harjoittamiseen käytetyn tietoaineistossa esiintyviä vinoumia yhtä usein kuin GPT-3-malli. Huomionarvoista on myös hallusinaatioiden määrän väheneminen noin puolella verrattuna GPT-3 malliin. InstructGPT myös tuotti kaksi kertaa todennäköisemmin oikean vastauksen esitettyyn kysymykseen kuin GPT-3.

2.9 ChatGPT

ChatGPT tarjoaa käyttöliittymän GPT-mallien käyttämiselle. Kielimallin kehittänyt OpenAI ei ole julkaissut ChatGPT:stä vertaisarvioitua artikkelia. OpenAI kuitenkin kertoo ChatGPT:n toimintaperiaatteista ja sen muista perustiedoista jotain ChatGPT:n esittelevällä verkkosivulla [22]. ChatGPT:stä on kuitenkin myös julkaistu useita artikkeleita eri tutkijoiden toimesta, esimerkiksi Wu et al. [23] ja Deng et al. [24] ovat julkaisseet katsauksen ChatGPT:stä.

ChatGPT:n tekninen ratkaisu pohjautuu sitä edeltävään InstructGPT-kielimalliin. Se myös käyttää pohjanaan samaa dataa kuin InstructGPT, kuitenkin joillain muokkauksilla ja lisäyksillä. Sen kouluttamismalli on ihmisen palautteeseen perustuva vahvenneoppiminen (eng. Reinforcement Learning from Human Feedback). ChatGPT:stä on julkaistu kaksi eri mallia (GPT-3.5 ja GPT-4). Näiden lisäksi on julkaistu useita erilaisia versioita, jotka poikkeavat toisistaan esimerkiksi ominaisuuksien (esimerkiksi kyky suorittaa Python-koodia) kuin myös esimerkiksi kontekstin koon perusteella.

OpenAI listaa myös samalla ChatGPT:n esittelevällä sivulla useita sen potentiaaliin ja käyttötapoihin liittyviä rajoituksia. Näitä ovat hallusinaatiot, kehotteen sisällön merkitys mallin kykyyn tuottaa oikeita vastauksia, mallin taipumus käyttää hyvin verboosia kieltä, sen taipumusta toistaa joitakin lauseita ja että se saattaa tuottaa haitallisia ohjeita tai toistaa kouluttamiseensa käytetyn aineiston stereotypioita. He myös nostavat esille, että useat näistä haasteista eivät ole helposti ratkaistavia, koska ne riippuvat mallin tietämyksestä ja mallin kouluttamiseen käytetyn datan laadusta.

3 Laajojen kielimallien käyttö opetuksessa

Ennen ChatGPT:tä laajojen kielimallien käyttö opetuksessa ei ollut yleistä. Aluksi alaluvuissa 3.1 ja 3.2 käydään läpi muita ohjelmoinnin opetukseen liittyviä järjestelmiä ja menetelmiä. Alaluvussa 3.3 esitellään käytännön esimerkki oppimisalustasta esittelemällä myös tässä tutkimuksessa käytettyä ViLLE-oppimisalustaa. Alaluvussa 3.4 käydään läpi erilaisia laajojen kielimallien hyödyntämismahdollisuuksia opetuksen kontekstissa. Laajojen kielimallien käyttämiseen liittyy myös yleisluonteisesti joitakin eettisiä kysymyksiä, joita käytiin tarkemmin läpi alaluvussa 2.4. Alaluvussa 3.5 käydään läpi suurten kielimallien kykyä ratkaista ohjelmoinnin tehtäviä.

Tämän lisäksi on noussut tarve yleiselle ohjeistukselle, miten ChatGPT:tä ja muita kielimalleja voidaan tai kannattaisi käyttää opetusta tukevana resurssina sekä oppilaiden, että opettajien näkökulmista mahdollisimman tehokkaasti ja eettisesti. Gimpel et al. [1] ovat koostaneet oppaan opettajille ja oppilaille kuinka generatiivista tekoälyä, esimerkiksi ChatGPT:tä tulisi hyödyntää. Heidän kehittämäänsä opasta käydään tarkemmmin läpi luvussa 3.6.

3.1 Ohjelmoinnin opetuksen järjestelmät ja menetelmät

Ohjelmoinnin opetukseen on vuosien saatossa käytetty useita erilaisia järjestelmiä, työkaluja ja menetelmiä. Useat työkalut visualisoivat ja pelillistävät ohjelmoinnin opettamista. Kanika et al. [25] tunnistivat arvioinnissan viisi erilaista klusteria ohjelmoinnin perusteiden opettamisen tekniikoista ja työkaluista.

- **Visuaalinen ohjelmointi**: Visuaalisessa ohjelmoinnissa suoritettava ohjelma visualisoidaan oppilaalle vuokaavion kaltaisella visualisaatiolla. Oppilaat oppivat ohjelmoinnin perusteet luomalla ohjelmointikoodia vuokaavioiden avulla.
- **Pelipohjainen oppiminen**: Ohjelmointiin opitaan kirjoittamalla ohjelmia, jotka pelaavat pelejä. Vaihtoehtoisesti oppilaita voidaan myös ohjeistaa luomaan yksinkertaisia pelejä käyttäen jotakin ohjelmointikieltä.
- **Pari- ja yhteisohjelmointi**: Oppilaat kirjoittavat ohjelmia pareittain tai ryhmissä. Oppimista tehostaa ryhmän sisäinen kommunikaatio.
- **Robotin ohjelmointi**: Oppilaat kirjoittamat ohjelmia, joilla robotit suorittavat yksinkertaisia tehtäviä oikeassa maailmassa. Usein robottien tarkoitus on lisätä oppilaiden motivaatiota ja sitoa ohjelmointi konkreettiseen tehtävän suoritukseen oikeassa maailmassa.
- **Arviointijärjestelmät**: Arviointijärjestelmät, jotka tarkastavat ja arvioivat oppilaiden kirjoittamia ohjelmia. Eri arviointijärjestelmien autonomiassa ja ominaisuuksissa voi olla suuria eroja.

Ohjelmoinnin opetuksen järjestelmät ovat usein adaptiivisia. Adaptiivisilla oppimisjärjestelmillä tarkoitetaan käytetyn järjestelmän kykyä sopeuta käyttäjänsä taitotasoon ja kykyyn personoida opetusta. Adaptiivisia oppimisjärjestelmiä käytetään

pienentämään opettajien työtaakkaa. Adaptiiviset järjestelmät ovatkin osoittautuneet tehokkaaksi oppimisen työkaluiksi erityisesti matemaattisissa aineissa. [26] [27] Adaptiiviset oppimisjärjestelmät hyödyntävät erilaisia algoritmeja toimiakseen. Nämä algoritmit toimivat yleensä ottamalla huomioon oppilaan menneen ja nykyisen suoriutumisen tason. Adaptiivisia oppimisjärjestelmiä on useita erilaisia, esimerkiksi ViLLE[1], ALEKS[2] ja Yixue Squirrel AI[3].

Laajoja kielimalleja on hyödynnetty opetuksessa toistaiseksi suhteellisen rajallisesti. Nye et al. [28] käsittelevät artikkelissaan mahdollisuuksia ja riskejä, joita laajan kielimallin käytössä tuutorina (eng. tutor) voi ilmetä. He esittävät artikkelissaan, että ChatGPT:tä ei voi korvata kokenutta tuutoria, koska kokeneen tuutorin toimintatavat eroavat usein laajan kielimallin toiminnasta. He nostavat kolme olennaista eroa kokeneen tuutorin ja ChatGPT:n (tai muun kielimallin välillä). Heidän mukaansa kokenut tuutori ohjaa usein keskustelun kulkua, kysyy kysymyksiä ennemin kuin kertoo vastauksia ja omaa käytännön kokemusta opetuksesta. Käytännön kokemus opetuksesta näkyy esimerkiksi kokeneen tuutorin kokemuksesta kumpuavasta kyvystä tunnistaa yleisiä oppilaiden tekemiä virheitä ja kyvystä vastata erilaisten oppilaiden henkilökohtaisia tarpeisiin paremmin kuin laaja kielimalli. Laajojen kielimallien käyttöä opetuksessa käsitellään tarkemmin luvussa 3.

3.2 Automaattisesti arvioitu ohjelmoinnin opetus

Erilaiset ohjelmoinnin tehtävät soveltuvat hyvin automaattisesti arvioitaviksi. Sama ohjelmakoodi tuottaa useimmilla ohjelmointikielillä aina saman lopputuloksen. Tämän takia ohjelmointitehtävien arviointia on kannattavaa automatisoida.

Ohjelmointitehtävien automaattinen arviointi on muuttunut vuosien saatossa.

[1] `https://ville.utu.fi/`

[2] `https://www.aleks.com/`

[3] `http://squirrelai.com/`

Samoin kursseilla käytetty ohjelmointikieli voi vaihdella hyvin suuresti riippuen kurssia järjestävästä tahosta. Lisäksi esimerkiksi Saikkonen et al. [29] huomauttivat jo 2001 tekemässään tutkimuksessa, että oppilaat suosivat käytössä olevia ohjelmointikieliä esimerkiksi pelkästään akateemisessa käytössä olevien ohjelmointikielien sijaan.

Ullah et al. [30] ovat analysoineet olemassaolevia automaattisia ohjelmoinnin tehtävien arvioinnissa käytettyjä keinoja. Staattisessa analyysissä hyödynnetään esimerkiksi koodirivien määrää, kommenttien tai käytettyjen operaattorien määrää. Dynaamisessa ohjelmoinnin arvioinnissa odotettua tulosta verrataan oikeaan vastaukseen tai oppilaan tuottamaa vastaus ajetaan testien läpi vastauksen oikeellisuuden varmistamiseksi. He totesivat, että parhaaseen lopputulokseen pääsemiseksi kannattaa usein hyödyntää ns. hybridimallia. Hybridimallissa hyödynnetään sekä staattisia, että dynaamisia analysoinnin keinoja.

3.3 ViLLE

ViLLE [31] on suomessa kehitetty oppimisen alusta. ViLLE:ssä on useita erilaisia tehtävätyyppejä ja sitä käytetään laajasti Suomen kouluissa. ViLLE:ä käytetään 15 maassa ja sillä on yhteensä yli 100 000 käyttäjää Suomessa. Se toimii myöskin suoraan useimmilla suosituimmilla internet-selaimilla, eikä vaadi ylimääräisiä asennuksia käyttäjän koneelle tai tabletille. Ensimmäinen versio ViLLE:stä julkaistiin vuonna 2005 ja sen tarkoitus oli olla mahdollisimman visuaalinen. Aluksi ViLLE:ä hyödynnettiin ohjelmoinnin ja matematiikan opetuksessa, mutta nykyään ViLLE:ä käytetään hyvin laajasti eri aineissa. Alustaa käytetään myöskin kaikilla Suomen koulutuksen tasoilla esikoulusta korkeakouluun. [32] [33].

ViLLE:n pääasiallinen tarkoitus on toimia oppimisen tukena ja opettajan työkaluna opettajan korvaamisen sijaan. ViLLE kehitys on alusta lähtien ollut tutkimuslähtöisä, ja useita ViLLE:n ominaisuuksia on hyödynnetty ja niiden tehokkuutta on

varmennuttu erilaisissa tutkimuksissa. ViLLE:ssä perustuu neljään pääperiaatteeseen: opettajien yhteistyö, oppilaiden yhteistyö, tehtävien automaattinen arviointi ja välitön palaute oppilaalle. [32]

ViLLE:ssä lähes kaikki opettajien tekemä materiaali on näkyvissä toisille opettajille. Laakso et al. [32] mukaan tämän tarkoitus on pienentää opettajien työtaakkaa tarjoamalla valmis materiaali. Samoin lähes kaikki materiaali ViLLE:ssä on aktiivista, ts. materiaalien lisäksi ViLLE:ssä on paljon erilaisia interaktiivisia tehtävätyyppejä. Esimerkiksi ViLLE:ssä käytettävät ohjelmistotehtävät kääntävät oppilaan kirjoittaman koodin vastauksen lähettämisen yhteydessä ja oppilas näkee käännetyn ohjelman tuottaman tulosteen tai virheilmoituksen. Oppilaan saama pistemäärä määräytyy vertaamalla oppilaan kirjoittaman koodin tuottamaa tulostetta mallivastauksen tulosteeseen [34]. Esimerkki tälläisestä tulosteesta voidaan nähdä kuvassa 3.1.

Kuva 3.1: Oppilaalle ohjelmointitehtävän palautuksen jälkeen näkyvä välitön palaute. Vasemmalla on oppilaan kirjoittaman ohjelmointikoodin tuottama tuloste ja oikealla mallivastauksen tuottama tuloste.

3.4 Laajojen kielimallien käyttö opetuksessa

Laajojen kielimallien käyttö opetuksessa sisältää useita mahdollisuuksia. Kielimallien käyttöä opetuksessa on myös tutkittu jonkin verran. Yan et al. [35] tekivät systemaattisen vertaisarvioitujen artikkelien katsauksen ja tunnistivat 53 erilaista käyttötapausta laajoille kielimalleille opetuksen automatisoinnin kontekstissa. He jakoivat tunnistamansa käyttötapaukset yhdeksään kategoriaan, jotka ovat esitelty kuvassa 3.2.

9 käyttötapauksen kategoriaa laajan kielimallin käyttöön opetuksessa

1. tekstin automaattinen luokittelu,
2. erilaisten tarpeiden tunnistaminen,
3. tehtävien arvostelu,
4. opetuksen tukeminen interaktiivisesti,
5. oppilaiden tulevan suoriutumiskyvyn arvioiminen,
6. tiedon formaatin muuntaminen,
7. kielimallin käyttämäminen palautteen antajana,
8. materiaalin generointi, ja
9. suositusten antaminen.

Kuva 3.2: Kuvassa on esitelty Yan et al. [35] tunnistamat yhdeksän erilaista kategoriaa käyttötapauksille laajan kielimallin käytöstä opetuksessa.

Samoin esimerkiksi Kasneci et al. [36] esittävät kirjoittamassaan artikkelissa useita mahdollisia käyttötapauksia laajojen kielimallien käyttämiselle opetuksessa. He tunnistivat kielimallien käytölle useita mahdollisuuksia erilaisissa opetukseen liittyvissä konteksteissa niin opettajien kuin opiskelijoiden näkökulmasta. Laajojen kielimallien käyttöön liittyviä mahdollisuuksia ovat heidän mukaansa esimerkiksi kielimallien käyttö opetuksen suunnittelussa, kielen opetuksessa ja ohjelmointitehtävissä. Macneil et al. [37] tunnistivat kolme erilaista käyttötapausta laajojen kielimallien käyttämisestä ohjelmoinnin opetuksessa. Nämä käyttötapaukset olivat aikavaativuuden selittäminen, aloittelevien ohjelmoijien yleisten virheiden havaitseminen ja koodin auki selittäminen.

Ottaen huomioon, että laajat kielimallit tulevat muuttamaan ohjelmoimista tulevaisuudessa merkittävästi [38], oppilaille pitäisi opettaa tehokkaita tapoja hyödyntää laajoja kielimalleja ohjelmoinnissa. Laajat kielimallit ovat tehokkaita selittämään koodin toiminnallisuutta auki. GPT-4 pystyy myös jopa ajamaan yksinkertaista pseudokoodia ilman, että sen tarvitsee kääntää sitä jollekin tunnetulle ohjelmointikielelle [38]. Savelka et al. [39] mainitsevat uskovansa, että laajat kielimallit osoittavat potentiaalia ratkaista tehokkaasti kaikkia yleisimpiä ohjelmoinnin kursseilla käytettyjä tehtävätyyppejä. Tekemässään tutkimuksessa he myös mainitsevat uskovansa, että ohjelmoinnin opettajien täytyy sopeutua siihen, että jo nyt saatavilla olevat laajat kielimallit (erityisesti GPT-4) ja tulevaisuuden sitäkin edistyneemmän mallit mahdollistavat opiskelijat läpäisemään ohjelmoinnin kursseja helposti jopa täysin ilman ohjelmoinnin osaamista.

3.5 GPT-mallien suorituskyky ohjelmoinnin tehtävien ratkaisussa

GPT-mallien kyvyistä ratkaista ohjelmoinnin tehtäviä on tehty jonkin verran tutkimuksia. Oman haasteensa kielimallien arvioinnista asettaa niiden luonne tuottaa useita erilaisia vastauksia samanlaisilla arvoilla. Esimerkiksi Finnie et al. [40] kiersivät tämän ongelman antamalla Codexin generoida 50 erilaista variaatiota esitettyyn kysymykseen ja huomioivat tutkimuksessaan näistä variaatioista parhaiten suoriutuvan. Codex on GPT-3:n pohjautuva malli, joka on koulutettu avustamaan ohjelmoinnissa [41]. Joissakin muissa tutkimuksissa mallien ulostulo on päätetty tehdä mahdollisimman deterministiseksi asettamalla mallin "lämpötila"(eng. temperature) mahdollisimman lähelle nollaa. Lämpötilan muokkaaminen vaikuttaa seuraavana olevan sanan todennäköisyysjakaumaan [42]. Tällöin malli tuottaa hyvin todennäköisesti aina saman ratkaisun samaan kysymykseen.

Finnie et al. [43] ovat tutkineet kahdessa eri tutkimuksessa Codexin kyvykkyyttä suoriutua ohjelmoinnin perusteista. Heidän 2022 julkaisemansa tutkimus päätyi johtopäätökseen, jossa Codex kykeni ratkaisemaan noin 80% kahdesta eri tentistä. He testasivat myöhemmin jatkotutkimuksessa [40] Codexin kyvykkyyttä ratkaista CS-2 tason ongelmia (algoritmit ja rakenteet), joita oli käytetty aiemmissa oppilaille laadituissa tenteissä. Codex sai tässäkin kokeessa 77% maksimipisteistä. Kummassakin tutkimuksessa Codex sijoittui verrannollisesti pisteissään parhaiten suoriutuneeseen oppilaiden kvartaaliin.

Savelka et al. [39] vertasivat GPT-4 malliin pohjautuva ChatGPT:tä aiempien laajojen kielimallien suoriutumiseen kolmella Python-ohjelmointikielellä suoritettavaan ohjelmointikurssiin. Kurssien tehtävät vaihtelivat kompleksisten ohjelmien ohjelmoinnista monivalintakysyksiin. Heidän tekemissään testeissä GPT-4 suoriutui paremmin kuin GPT-3.5. Jos mallit saivat hyödyntää automaattisesti vastauksista luotua palautetta, GPT-4 sai ohjelmointitehtävien ratkaisuistaan 83.4% ja GPT-3.5 sai 66.4% ohjelmoinnin tehtävien maksimipisteistä. Mikäli vain kielimallin ensimmäinen yritys ratkaista tehtävä otettiin huomioon, oli pistemäärä kummallakin matalampi. Kuitenkin GPT.4 suoriutui myös kyseisssä vertailussa paremmin kuin GPT-3.5. GPT-3.5 sai 53.6%ja GPT-4 71.7% kokonaispisteistä mikäli vain mallien tuottaman ensimmäisen vastauksen pisteet otettiin huomioon.

3.6 Eettinen ohjeistus generatiivisen tekoälyn käyttöön opetuksessa

Laajojen kielimallien käyttäminen on luonut tarpeen eettiselle ohjeistukselle. Gimpel et al. [1] ovat luoneet oppaan, joka sisältää joukon käytännönläheisiä ohjeita opettajille ja oppilaille. Näitä ohjeita käydään läpi tarkemmin luvuissa 3.6.1 ja 3.6.2.

3.6.1 Opettajille suunnattu ohjeistus

Gimpel et al. [1] ovat luoneet opettajille viiden kohdan ohjeistuksen, jonka pyrkimys on auttaa opettajaa hyödyntämään ChatGPT:tä järkevällä tavalla. Ohjeet ovat nähtävillä kuvassa 3.3.

He myös nostavat esille, että on tärkeää ohjata opetusta aiemmin asetettujen tavoitteiden mukaisesti generatiivisen tekoälyn hyödyntämisestä huolimatta. He nostavat myös esille korkeakouluopintojen tarpeen sopeutua käyttämään ja opettamaan parhaiden mahdollisten työkalujen käyttöä. Opettajien pitäisi myös kannustaa oppilaita käyttämään ChatGPT:tä oppimisen tukena sen sijaan, että ChatGPT:tä ja muita vastaavia työkaluja käytettäisiin korvaamaan ajattelua. He myös nostavat esille mahdollisuuden reflektoida ChatGPT:n tuottamia vastauksia. Voimme myös tarkastella Kasneci et al. [36] tunnistimia samankaltaisia haasteita kielimallien käytölle opetuksessa. Heidän tunnistamiaan yleisiä haasteita laajojen kielimallien ja muiden AI-työkalujen hyödyntämiselle opetuksessa olivat tekijänoikeudella suojatun materiaalin käyttö suurten kielimallien kouluttamisessa, tekoälyn taipumus toistaa tekoälyn kouluttamiseen käytetyn materiaalissa esiintyviä vinoumia sekä opettajien ja opiskelijoiden mahdollinen liian suuri nojautuminen kielimallien tuottamille vastauksille.

ChatGPT voi myös auttaa oppimateriaalin personoinnissa oppilaslähtöisesti. Sitä voidaan käyttää esimerkiksi auttaa moduulien luomisessa, luentojen ideoimisessa, seminaarien suunnittelemisessa ja paranmaaan oppilaille lähetettyjen tiedotteiden laatua. He Kuitenkin muistuttavat, että opettajan on aina pidettävä mielessä käytössä olevan teknologian rajoitteet.

Gimpel et al. [1] yleiset opettajan ohjeet

1. Reflektoi opetustavoitteen mukaan.

2. Luo opetusmateriaaleja ChatGPT:n avustuksella.

3. Tue oppilaiden osaamista tietovisoilla.

4. Vahvista oppilaiden osaamista ChatGPT:n avulla.

5. Kannusta oppilaita käyttämään ChatGPT:tä.

Kuva 3.3: Gimpel et al. [1] koostamat ohjeet opettajille generatiivisen tekoälyn käytöstä.

Näiden ohjeistuisten lisäksi he ovat koostaneet myös ohjeet siitä, kuinka ChatGPT vaikuttaa tehtävien arviointiin ja arviointiin käytettäviin perusteisiin. Nämä löytyvät taulukosta 3.4, mutta niitä ei kuitenkaan käydä tarkemmin läpi tämän tutkielman puitteissa.

Gimpel et. [1] al ohjeet tehtävien arviointiin

1. Suunnittele tehtävät uusi teknologia huomioiden.

2. Vaadi oppilailta avoimuutta uusien kehittyneiden työkalujen käyttämisessä.

3. Innovoi tehtävien tyyppejä.

4. Suunnittele tehtävien teon valvonnan prosessi uudelleen.

5. Innovoi käytettyjä arvioinnin kriteerejä.

6. Luo kattava ohjeistus, joka ottaa huomioon mahdollisen plagioinnin ja tekijänoikeusloukkaukset.

7. Opeta oppilaita käyttämään ChatGPT:tä oikein.

8. Luo käytetyille työkaluille selkeät ohjeet.

Kuva 3.4: Gimpel et al.(2023) koostamat ohjeet generatiivisen tekoälyn käytöstä tehtävien arvioinnissa.

3.6.2 Oppilaille suunnattu ohjeistus

Oppilaille suunnatussa ohjeistuksessa Gimpel et al. [1] painottavat eettisyyden merkitystä ja kriittisen ajattelun tärkeyttä. Oppilaat voivat kuitenkin käyttää ChatGPT:tä reflektoimisen apuna. He myös painottavat, että opiskelijoille tulisi korostaa hyviä käytänteitä lähteiden merkitsemisessä.

Heidän mukaansa yksi suurimpia vahvuuksia ChatGPT:n käytössä on se, että kielimalli ei ole inhimillinen olento. Se ei siis väsy tai ärsyynny. Oikeasta opettajasta poiketen se on myös aina opiskelijoiden tavoitettavissa. ChatGPT:ltä voi myös kysyä loputtomasti tarkentavia kysymyksiä. ChatGPT voi myös tiivistää pitkiä tekstejä tai sitä voi käyttää ohjelmoimisen apuna esimerkiksi selittämään koodia.

He kuitenkin myös painottavat, että oppilaiden on myös tunnettava ChatGPT:n käyttöön liittyvät riskit. Erityisesti ChatGPT:n taipumus tuottaa myös hyvin vakuuttavan kuuloista, mutta jopa täysin virheellistä tekstiä on otettava esille opiskelijoiden kanssa (hallusinaatioita). Opiskelijoille on painotettava, että vastuu tekstistä on viime kädessä tekstin kirjoittajalla, eikä sen luomiseen avustajana käytetyllä tekoälyllä. Tarkempi listaus Gimplel et al. [1] luomista ohjeista oppilaille löytyy taulukosta 3.5.

Gimpel et. al koostama oppilaiden ohje
1. Kunnioita lakia ja oppilaitoksesi säädöksiä
2. Reflektoi oppimistavoitteitasi
3. Käytä ChatGPT:tä kirjoittamisen apuna
4. Käytä ChatGPT oppimisen apuna
5. Toista ja keskustele ChatGPT:n kanssa
6. Luo tiivistelmiä materiaalista ChatGPT:n kanssa

7. Käytä ChatGPT:tä ohjelmoimisen apuna

8. Ymmärrä ja varo ChatGPT:n käyttämiseen liittyviä riskejä

Kuva 3.5: Gimpel et al.(2023) koostamat ohjeet oppilaille generatiivisen tekoälyn eettisestä käytöstä.

3.7 Plagioinnin tunnistaminen

Tämän hetkisen tiedon valossa opettajilla tulee olemaan haasteita tunnistaa kielimalleilla luotua tekstiä. Pegararon et al. [44] testasivat tutkimuksessaan useita maksullisia ja maksuttomia internetissä olevia AI:n tuottaman tekstin tunnistavia työkaluja ja havaitsivat, että heidän testaamansa työkalujen tarkkuus tekoälyllä luodun tekstin tunnistamiseen jäi useimmilla työkaluilla alle 50% tarkkuuden, ja että yksikään heidän testaamistaan työkaluista ei ollut riittävän tarkka tunnistamaan luotettavasti tekstin alkuperäiseksi kirjoittajaksi tekoälyä. Kuitenkin Mitrovic et al. [45] tekivät tutkimuksen, jossa he keskittyivät lyhyiden tekstien tekoälyn tuottamien tekstien tunnistamiseen. Heidän luomansa malli onnistui tunnistamaan ChatGPT:n tuottaman tekstin 79% tarkkuudella. Tämäkin on kuitenkin vielä hyvin kaukana 100% tarkkuudesta. Lisäksi voidaan kysyä,tuleeko kielimallien tuottama teksti muuttamaan myös sitä, kuinka ihmiset tulevaisuudessa kirjoittavat. Tämä voisi mahdollisesti vaikeuttaa tekoälyn tuottaman tekstin tuottamista entisestään.

Tekoälyn tuottamaa tekstiä on tällä hetkellä mahdoton tunnistaa luotettavasti myös mm. ChatGPT:n kehittäneen OpenAI:n mukaan, ja he myös huomauttavat, että tekstitunnistustyökalujen kiertäminen on suhteellisen yksinkertaista [46]. Lisäksi on nostettava esille Sadasivan et al. [47] tekemä tutkimus, jossa he esittävät teorian sekä käytännön kautta, että tekoälyn luoman tekstin tunnistaminen täysin luotettavasti voi osoittautua tulevaisuudessa mahdottomaksi.

Lisähaasteena ohjelmoinnin opetuksen kontekstissa tulisi mainita, että varsinkin ohjelmoinnin alkeiden opettelussa käytetään tyypillisesti suhteellisen lyhyitä tehtäviä. Tämä tuo kopioidun tekstin tunnistamiseen ohjelmointitehtävistä tuo oman lisähaastensa tuotetun tekstin (vastaus ohjelmointitehtävään) lyhyyden vuoksi.

4 Tutkimusasetelma

Tässä kappaleessa käsitellään tutkimuksessa käytetty tutkimusasetelma. Valittu tutkimusmenetelmä on kuvattu alaluvussa 4.1. Tutkimuksessa käytetty tietoaineisto on kuvattu alaluvussa 4.2. Alaluvussa 4.3 käydään läpi tutkimuksen kulku.

4.1 Tutkimusmenetelmä

Tutkimusmenetelmänä käytettiin simulaatiota. Simulaatio mahdolllistaa kompleksisten asioiden testauksen. Simulaatiossa voidaan mallintaa maailmaa. Simulaatio myös periaatteessa voi mahdollistaa myös deterministisen lähestymistavan. [48]. Lähestymistavat simulaatioihin voidaan jakaa kolmeen erilaiseen luokkaan.

- *Diskreetti tapahtumasimulaatio* : Diskreettiä tapahtumasimulataatiota voidaan käyttää, mikäli mallinnus voidaan tehdä parametreillä, ja jossa "muutokset tapahtuvat sääntöjen mukaan, mutta stokastisella tavalla. [48]
- *Jatkuva simulaatio*: Jatkuva simulaatio sopii tilanteisiin, jossa simulaatiossa käytetyt eri muuttujat voivat vaikuttaa toisiinsa. [48]
- *Agenttipohjainen simulaatio*: Agenttipohjaiset simulaatiot toimii parhaiten, kun simuloitu organisaatio voidaan kuvailla ryhmänä erilaisia itsenäisiä agentteja. Agenttipohjaisissa järjestelmissä yleensä pyritään kuvaamaan yksittäisiä agentteja ja agenttien yhteistoiminta tulee esiin usein ilman agenttien vuorovaikutussuhteen tarkempaa määrittelyä. [48]

Tämän tutkimuksen voidaan katsoa noudattavan diskreettiä tapahtumasimulaatiota. Tutkimuksessa kuitenkin hyödynnettiin ei-deterministiä laajoja kielimalleja. Tarkempia tietoja tehdyn tutkimuksen rajoitteista voi lukea alaluvusta 6.4.

4.2 Tietoaineisto

Tutkimuksessa hyödynnetiin ohjelmoinnin perusteet kurssin tehtäviä [49]. Kurssilla käsiteltävät aiheet ovat muun muassa "ohjelmien kirjoittaminen editorilla, hyvä ohjelmointitapa, muuttujat, viittaukset, peruskontrollirakenteet, perustietorakenteet, syöttö ja tulostus, algoritminen ongelmanratkaisu, modulaarisuus, funktiot."Kurssilla oli yhteensä 174 uniikkia tehtävää ja näistä 141 oli ohjelmointitehtäviä. Valtaosasta tehtäviä annettiin sama pistemäärä. Tehtävät olivat jaoteltu viikoittain teeman mukaan ja jokainen viikko toi uusia teemoja ja rakensi oppilaiden osaamista aiemmilla viikoilla opetettujen teemojen päälle. Kurssin rakenne oli seuraavanlainen:

1. Viikko 1: Johdatus ohjelmointiin
2. Viikko 2: Lisää ehtoja
3. Viikko 3: Lisää toistoa, merkkijonot, oma funktio
4. Viikko 4: Lisää funktioita, johdatus listoihin
5. Viikko 5: Lisää tietorakenteista
6. Viikko 6: Tiedostot ja virheet
7. Viikko 7: Kirjastot ja lisää Pythonia

Näiden lisäksi kurssilla käsiteltäviä aiheita olivat myös ohjelmointiympäristön ja Pythonin asennus oppilaan laitteelle. Ohjelmoinnin perusteet-kurssin läpäisy arvo-

sanalla 1 (asteikolla hyl-5) vaati 50% kurssin ja 50% tentin kokonaispistemäärästä. Kokonaispistemäärä kurssilla oli 1690.

Simuloitu oppilas koitti ChatGPT:n avulla ratkaista kaikki kurssilla olevat 134 ohjelmointitehtävää. Tämän lisäksi ChatGPT:n avustaman oppilaan kyvykkyyttä testattiin myös kurssin kokeissa käytetyillä tehtävillä. Ohjelmointitehtäviä annettiin kolmesta kurssilla käytöstä olevasta tentistä: vapauttava koe, kurssin varsinainen koe ja uusintakoe. Kaikkien kokeiden rakenne oli samanlainen. Jokaisessa oli yhteensä viisi kysymystä ja nämä kysymykset olivat ohjelmointehtäviä. Poikkeuksen muodosti vapauttava koe, jossa ensimmäinen tehtävä oli ohjelmointitehtävän sijaan lyhyt termien selitystehtävä.

Kurssilla olevia tehtäviä arvioitiin automaattisesti pisteyttämällä oppimisjärjestelmä ViLLE:ssä. ViLLEä käsiteltiin tarkemmin alaluvussa 3.3. ViLLE:ssä ohjelmointitehtävissä annetut pisteet määrittyvät vertaamalla oppilaan tuottaman vastauksen tulostetta esimerkkivastauksen tuottamaan tulosteeseen. Esimerkki tästä nähtiin kuvassa 3.1. Huijaamisen mahdollisuuden pienentämiseksi useimmat tehtävät hyödynsivät satunnaisuutta annetuissa parametreissa, jolloin kurssia käyvä oppilas ei voi suoraan kopioida esimerkkivastauksien antamaa tulostetta seuraavaan antamaansa vastaukseen.

4.3 Tutkimuksen kulku

Tutkimuksessa simuloitu oppilas koitti ratkaista kaikki kurssilla olevat ohjelmointitehtävät ChatGPT:n avustuksella. Tehtävät käytiin läpi yksi kerrallaan tässä alaluvussa hieman myöhemmin kuvattua metodologiaa noudattaen. Tutkimuksen tuloksien keräämisen helpottamiseksi simuloidulle ChatGPT:tä hyödyntävälle oppilaalle luotiin ViLLE:en oma tunnus, joka lisättiin ohjelmoinnin perusteet -kurssille. Simulaatio tehtiin hyödyntämällä neljää erilaista roolia. Nämä neljä rooli valittiin, koska niiden katsottiin antavan kattavan kuvan erilaisista mahdollisuuksista, taitotasosta

ja työkaluista, joita simuloitu ChatGPT:tä käyttävä oppilas voisi käyttää läpäistäkseen kurssin ChatGPT:hen hyvin vahvasti nojautuen.

- **Täysin ohjelmoinnista tietämättömän opiskelijan simulointi**: GPT 3.5:n antaman vastauksen koodi-osio kopioitiin ViLLE:en ilman minkäänlaisia muutoksia. Mikäli vastauksessa oli useampi koodi-osio, tehtiin oletus, että simuloitu oppilas koittaisi molempien koodi-osioiden syöttämistä ViLLE:en erikseen. Kuitenkin tämän roolin mukainen simuloidun opiskelijan ei oletettu ymmärtävän kopiomastaan koodista mitään.

- **Nolla-esimerkki GPT-3.5**: Ohjelmointia hieman ymmärtävän opiskelijan ensimmäinen palautus. Hieman ohjelmointia ymmärtävän opiskelijan palautuksissa oletttiin, että simuloitu oppilas kykenisi esimerkiksi tunnistamaan ja jättämään pois ChatGPT:ltä kopioimastaan vastauksesta esimerkiksi tehtävässä jo valmiiksi alustetut muuttujat.

- **Systemaattisen metodologian GPT-3.5**: GPT-3.5 hyödyntävän oppilaan simuloinnissa oppilas sai maksimissaaan tehdä kolme palautusta ViLLE:en alla kuvattua metodologiaa noudattaen.

- **Systemaattisen metodologian GPT-4**: Myös GPT-4 hyödyntävän oppilaan simuloinnissa noudatettiin alla kuvattua metodologiaa. Kuitenkin GPT-4-mallia hyödyntävän oppilaan simuloinnissa simuloidun oppilaan sallittiin tehdä palautuksia niin kauan kuin vastaus vaikutti edistyvän. Samoin GPT-4 hyödyntävässä simulaatiossa ajettiin vain läpi tehtävät, joissa GPT-3.5 ei onnnistunut saamaan täysiä pisteitä.

Tutkimusaineistossa käytettävät ohjelmointitehtävien ajamiseen käytetty tarkempi metodologia on kuvattu alla. Visuaalinen esitys käytetyn metodologian päävaiheista löytyy kuvasta 4.1.

1. Tutkija avasi ViLLE-tehtävän ja kopioi tehtävänannon.

2. ChatGPT:lle annettiin seuraavanlainen syöte:
   "*Kirjoita vastaus alla olevaan kysymykseen Pythonilla:*
   <ViLLE:n tehtävänanto>"

3. ChatGPT:n vastauksesta kopioitiin ViLLE:en koodi ja vastaus tarkistettiin käyttämällä ViLLE:n oppilaille annettua automaattista palautetta.

4. Tulos lisättiin taulukkoon.

5. Mikäli tuotti oikean vastauksen oli oikein, siiryttiin eteenpäin seuraavaan kysymykseen.

6. Mikäli vastaus oli väärin, ChatGPT:lle annettiin yksi alla olevista kehotteista riippuen siitä, miten vastaus oli virheellinen:

   (a) Mikäli ViLLE antaa palauttaa virheviestin, syötettiin virheviesti alla olevalla tavalla.
   "*Hei, sain tällaisen virheviestin:*
   <Virheviesti>.
   Voisitko korjata ohjelman niin, että se toimii aiemmin kuvatulla tavalla?"

   (b) Mikäli ChatGPT:n vastauksesssa olevan moduulin käyttöä ei oltu sallittu kyseisessä tehtävässä annattiin sille alla oleva kehote. Inhimillisistä tekijöistä johtuen kahdessa tehtävässä annettu kehote poikkesi hieman alla olevasta esimerkistä.
   Hei, et voi käyttää <moduulin-nimi>-moduulia. Voisitko tehdä saman ilman ulkoisia moduuleja?"

   (c) Mikäli kopioitu koodi oli väärin, mutta kopioitu koodi kuitenkin kääntyi ChatGPT:lle annettiin alla oleva syöte:

"*Hei, syöte tulostaa arvoja:*

<ViLLE:n oppilaan tuottaman koodin ulos tulostama syöte esimerkkiarvoilla>.

Sen pitäisi tulostaa vastaavilla arvoilla:

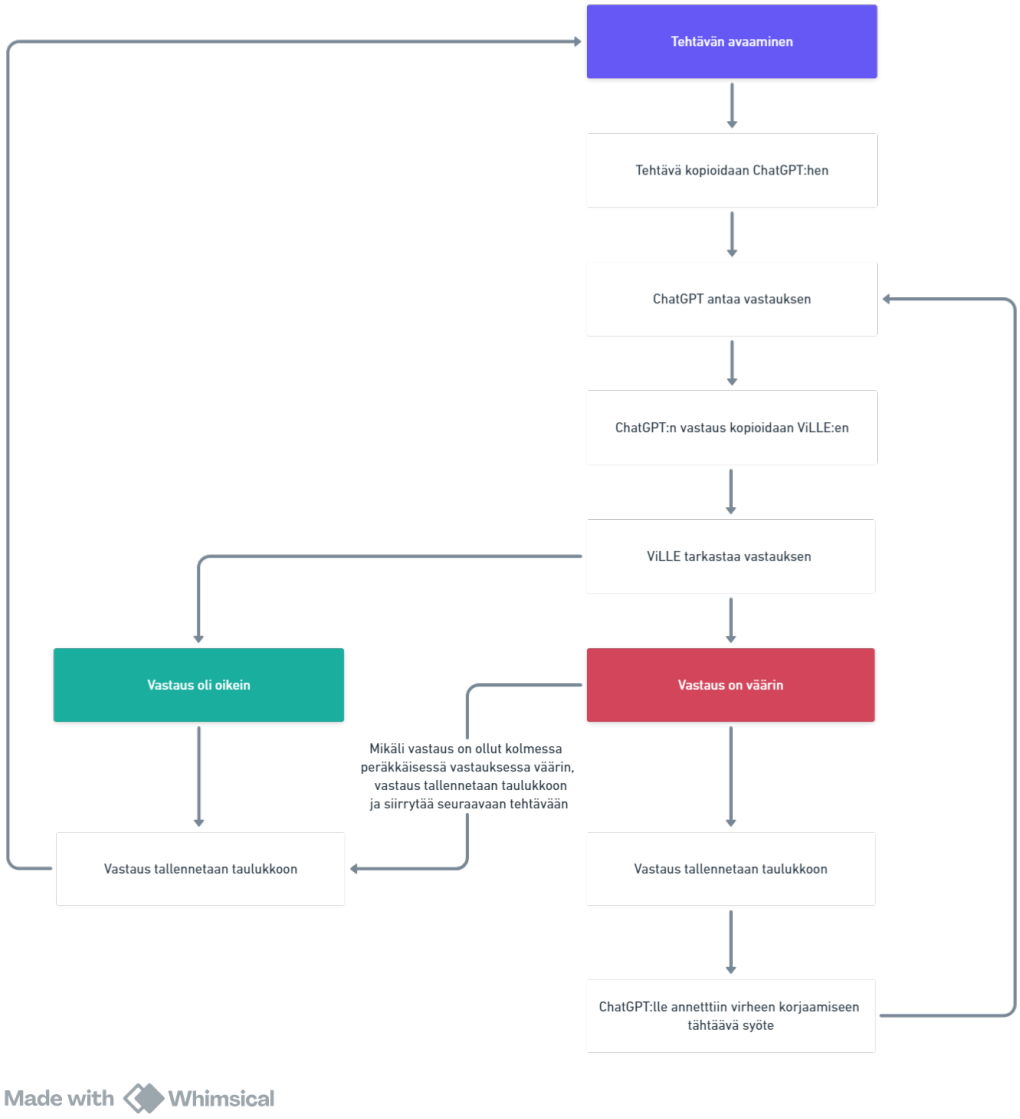
<Oikean vastauksen esimerkkisyöte>

Voisitko korjata ohjelman niin, että se toimii halutulla tavalla?""

7. Tulos lisättiin taulukkoon.

8. Mikäli koodi tuottama vastaus oli oikein, siiryttiin seuraavaan kysymykseen.

9. Mikäli koodin tuottama vastaus oli väärin, askeleet 5-7 toistettiin. Virheellistä syötettä koitettiin korjata korkeintaan kaksi kertaa. tämän jälkeen lopputuloksesta riippumatta siirryttiin seuraavaan kysymykseen. Tästä poiketen simuloidulle GPT-4 hyödyntävä oppilas sai jatkaa palautuksia niin kauan kuin vastaus vaikutti edistyvän.



Kuva 4.1: Tutkimuksen kulku prosessikaaviona. Tarkempi selitys vaiheista löytyy luvusta 4.3. Sanallisessa selityksessä avataan myös käytetyt kehotteet.

5 Tulokset

Tässä luvussa käydään läpi tutkimuksen lopputulokset ja lopuksi hieman yleistä pohdintaa aihepiiriin liittyen. Aluksi käydään läpi ChatGPT:n kyvykkyys ratkaista annettu aineisto ja miten virheet jakautuivat korkealla abstraktiotasolla. Sen jälkeen tehtäviä käydään yksittäin läpi, ja myös analysoidaan yleisimmät GPT-3.5:n tekemät virheet tarkemmin. Simuloitu GPT-4 hyödyntävä opiskelija ei tehnyt riittävästi virheitä niiden analysoimiseksi tarkemmin.

Lopuksi pohdinnassa käydään läpi, kuinka suurten kielimallien kehittyminen vaikuttaa ohjelmoinnin opetukseen ja ohjelmointiin yleisesti, ja pohditaan kuinka suuria kielimalleja voitaisiin tulevaisuudessa hyödyntää opetuksessa.

5.1 Simuloitujen oppilaiden tulokset

Tutkimuskysymys "*TK1: Kykeneekö ChatGPT:tä hyödyntävä alkeelliset taidot omaava opiskelija läpäisemään ohjelmoinnin peruskurssin?*", käydään läpi tässä luvussa. Samoin tässä luvussa kerrotaan, kuinka monta kurssilla käytettyä ohjelmointitehtävää ChatGPT:n apuun tukeutuva simuloitu opiskelija pystyi ratkaisemaan hyödyntämällä ChatGPT:tä.

5.1.1 Kurssin tehtävistä suoriutuminen

ChatGPT:n suorituskykyä voidaan tarkastella kahdesta eri näkökulmasta. Ensimmäinen näkökulma on kuinka suuren prosentuaalisen osuuden kurssin pisteistä opis-

kelija voisi saada hyödyntämällä ChatGPT:tä luvussa 4.3 kuvatulla metodologialla. ChatGPT:tä hyödyntävän oppilaan suoriutuminen vaihteli 65.9% ja 86.2% välillä käytetyn GPT-mallin ja metodologian mukaan. Täysin tietämättömän opiskelijan saavutti vain 63.4% kurssin kokonaispisteistä. Tarkempi kuvaus saavuteituista pisteistä on nähtävissä taulukossa 5.1.

Toinen mahdollinen näkökulma on tarkastella ChatGPT-avusteisen opiskelijan kykyä ottamalla tarkasteluun pelkästään kurssilla olevat ohjelmointitehtävät. Tämän näkökulman voidaan nähdä tarjoavan hieman tarkemman kuvauksen simuloidun oppilaan kyvyistä ratkaista puhtaasti *sanallisia ohjelmointitehtäviä*. Kurssilla oli 141 tehtävää, joissa suoriutuminen ohjelmointia hieman osaavan opiskelijan näkökulmasta vaihteli 75.9% (GPT-3.5) ja 98.6% (GPT-4) välillä. GPT-4 sai ratkaistua sille annetuista tehtävistä suurimman osan, eikä sen vastauksista erottunut selkeitä ongelmakohtia. GPT-4 malliin pohjautuvaa ChatGPT:tä käytettiin vastaamaan vain kysymyksiin, joista GPT-3.5 ei saanut täysiä pisteitä. Simuloitu GPT-4 pohjainen oppilas kykeni vastaamaan myös 3 kurssin loppupuolen kysymykseen, joihin simuloitu GPT-3.5 tukeutuva oppilas ei pystynyt vastaamaan niiden vaatiesssa aiemman tehtävän onnistunutta suorittamista. Simuloidulle GPT-4:n hyödyntävälle opiskelijalle myös annettiin kuvatusta metodologiasta poiketen useampia vastausmahdollisuuksia muutamiin monimutkaisempiin tehtäviin, koska GPT-4:n antama vastaus vaikutti edistyvän jokaisessa tehdyssä palautuksessa. Tarkemmat luvut ratkaistujen ohjelmointitehtävien määristä ovat nähtävissä taulukossa 5.2.

Taulukko 5.1: ChatGPT:n koko kurssista saavuttama pistemäärä. Huomioitavaa on, että ChatGPT:tä käytettiin vastaamaan vain kurssilla oleviin ohjelmointitehtäviin. Kurssin täysi pistemäärä oli 1690.

Tyyppi	Prosenttiosuus	Pistemäärä
Tietämättömän opiskelijan simulointi	63.4%	1072
Nolla-esimerkki GPT-3.5	65.9%	1114
GPT-3.5 kuvatulla metodologialla	77.1%	1303
GPT-4 kuvatulla metodologialla	86.2%	1457
Täysi pistemäärä	100%	1690

Taulukko 5.2: Simuloidun ChatGPT:n avustaman oppilaan kyvykkyys ratkaista kurssilla olevia ohjelmoinnnin tehtäviä. Ohjelmointitehtävien kokonaismäärä oli 141.

Malli	GPT-3.5		GPT-4	
Kategoria	Määrä	%	Määrä	%
Vastatut ohjelmoinnin tehtävät	138	97.9.%	17	12%
Vastattu oikein ensimmäisellä yrityksellä	107	75.9%	9	6.3%
Vastattu oikein toisella yrityksellä	13	9.2%	2	1.4%
Vastattu oikein kolmannella yrityksellä	5	3.5%	2	1.4%
Vastattu oikein ylipäätään	-	0	2	1.4%
Vähintään osittain väärin viimeisen palautuksen jälkeen	11	7.8%	2	1.4%
Vastattu oikein	124	87.9%	15	10.6%
Täysin oikein (100%) lkm.	124	87.9%	139*	98.6%*

**kun lasketaan GPT3.5:n ja GPT4:n ratkaisemat tehtävät yhteen*

..

5.1.2 Kurssin kokeista suoriutuminen

ChatGPT:n läpi ajettiin myös kurssin kokeissa käytetyt kysymykset, mutta nämä käsiteltiin kurssilla käytetyistä tehtävistä erikseen. Niiden rakenne oli seuraavanlainen: Varsinainen koe koostui 5 ohjelmointitehtävästä, joista ChatGPT 3.5 hyödyntävä simuloitu opiskelija sai kaikki tehtävät täysin oikein. Uusintakokeessa oli samanlainen rakenne ja siinäkin simuloitu opiskelija saavutti täyden pistemäärän. Vapauttava koe sen sijaan koostui yhdestä termien selitystehtävästä ja neljästä ohjelmointitehtävästä. Termien selitystehtävää ei arvioitu, mutta pikaisesti silmäiltynä ChatGPT vaikutti antavan myös termien oikeat määritelmät selitystehtävässä. Vapauttavan kokeen ohjelmointitehtävistä GPT-3.5:ttä hyödyntävä opiskelija sai oikein kolme neljästä kokeessa olleesta ohjelmointitehtävästä. On myös mainittava, että automaattinen arviointi koetehtävissä oli, kurssilla olevista tehtävistä poiketen, pelkästään *hyväksytty/hylätty* arviointi kurssin aikana tehtyjen ohjelmistotehtävissä käytetyn numeerisen pisteytyksen sijaan. GPT-4 hyödyntävän opiskelijan simulaatiota ei ajettu, sillä jo GPT-3.5:tä hyödyntävä simuloitu opiskelija olisi läpäissyt

kaikki kurssilla käytetyt kokeet.

5.1.3 Täysin ohjelmoinnista tietämättömän opiskelijan simulointi

Tässä alaluvussa käsitellään tutkimuskysymystä "*TK2: Voiko täysin tietämätön opiskelija läpäistä ohjelmoinnin peruskurssin osaamatta ollenkaan ohjelmoida?*". Tämä on hyvin olennainen kysymys opetuksen näkökulmasta. Laajojen kielimallien yleistyessä eräs olennainnen kysymys on, voiko ohjelmoinnista täysin tietämätön opiskelija päästä ohjelmoinnin perusteet-kurssista läpi ilman alkeellistakaan ymmärrystä. Täysin tietämätön opiskelija saisi luvussa 4.3 esitetyllä metodilla 1072 pistettä. Tämä on noin 63% koko kurssin pistemäärästä, mikä riittäisi kurssin läpäisyyn.

Tietämättömän opiskelijan pistemäärä voisi olla myös korkeampi, sillä tutkimuksessa alkuperäisen tehtävänannon kanssa käytetty syöte ei sisältänyt tietoa ViLLE:n ominaispiirteistä. Mikäli ChatGPT:lle annettu kehote olisi sisältänyt enemmän tietoa siitä, millaiseen ympäristöön ohjelmointitehtäviä tehdään, esimerkiksi tiedon, että ohjelmointitehtävissä käytetyt muuttujat alustetaan usein valmiiksi, olisi tämä johtanut todennäköisesti vielä korkeampaan pistemäärään. On myös huomautettava, että alkuperäinen tutkimusidea oli oli simuloida ohjelmoinnin alkeet ymmärtävää opiskelijaa. Täysin ohjelmoinnista tietämättömän opiskelijan simuloinnin idea tuli vasta tutkimuksen suorittamisen jälkeen. Tämä johti käytännössä siihen, että täysin tietämättömän opiskelijan ratkaisemista tehtävistä osa täyden pistemäärän tehtävistä rajattiin pois täysin tietämättömän opiskelijan aineistosta. Tämä johti oikeaa hieman alempaan saavutettuun pistemäärään simuloidulle täysin tietämättömälle opiskelijan simulaatiossa. Simuloitu täysin tietämätön opiskelija pystyi myös läpäisemään kurssin kokeet alaluvussa 5.1.2 kuvatulla tavalla.

5.2 Yleisiä ChatGPT 3.5:n tekemiä virheitä

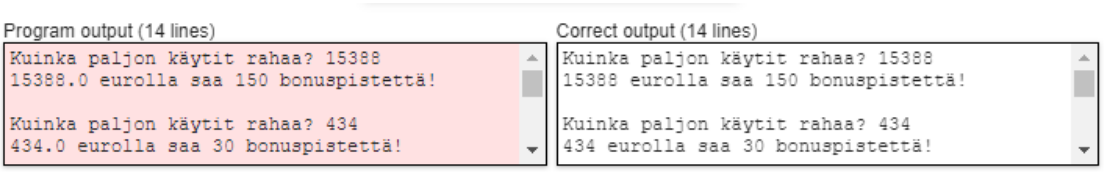
Tässä alaluvussa vastataan tutkimuskysymykseen, *"TK3: Mitkä ovat ChatGPT:n tekemät yleisimmät virheet ohjelmoinnin perusteiden tehtävissä?"*. Yleisimmät ChatGPT:n antamissa vastauksissa olevat virheet liittyivät tulostuksen muotoiluun. Tämä on ymmärrettävää, koska tästä ei annettu useinkaan tarkkaa määritelmää, vaan ChatGPT oli usein yksittäisen esimerkin varassa. Osa ChatGPT:n tekemistä virheistä johtui tutkimusasetelmasta ja haasteista joita ViLLE:n käyttäminen ohjelmointiympäristönä aiheutti. Tutkimusasetelmaan liittyvät mahdolliset ChatGPT:n virheet eivät näy aineistossa, ja näitä reunaehtoja on kuvattu tarkemmin luvussa 6.4.Yleisin ChatGPT:n tekemä virhe, oli ohjelman tulostukseen liittyvä muotovirhe. Kuitenkin usein näissä tehtävissä tehtävä saattoi olla muuten oikein ratkaistu. Havainnollistava esimerkki on nähtävissä kuvassa 5.1. Kuvaajasta 5.2 huomataan, että suurin osa ChatGPT:n tekemistä virheistä olivat tekstin tulostukseen liittyviä ongelmia. Alla on kuvattu kuavaajassa 5.2 käytetyt eri virheiden tyypit.

- **Virheellinen tuloste**: Tehtävä kääntyi, mutta sen tuottama tuloste oli virheellinen. Vastaus saattoi olla kokonaan tai osittain virheellinen. Useimmat virheet olivat vain pieniä koodin tuottamaan tulosteeseen liittyviä muotoseikkoja.

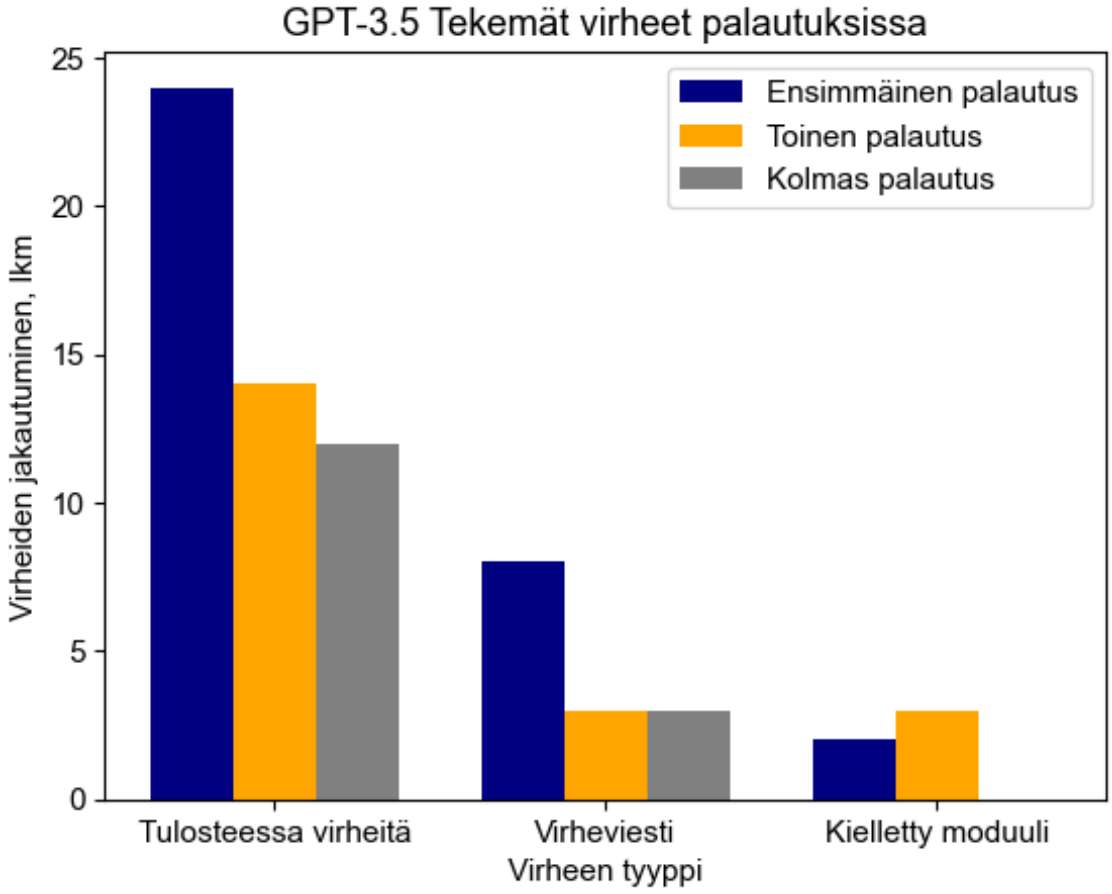
- **Virheviesti**: Ohjelma ei käänny virheen takia.

- **Kielletty moduuli**: ViLLE:n virheviesti kielletyn moduulin käytöstä. Useimmissa tehtävissä ulkoisten moduulien käyttöä ei oltu sallittu.

Kerättyä aineistoa tarkemmin tarkestelemalla tuli ilmi, että useimmissa virheellisesti ratkaistuissa tehtävissä ratkaisut olivat kuitenkin logiikan kannalta oikeita. Toisin sanoen tehtävän koodin logiikka itsessään oli oikea, mutta ohjelmakoodin tuottamassa tulosteessa olevan muotovirheen tai valmiiksi alustetun muuttujan vuoksi simuloitu opiskelija ei saanutkaan tehtävästä täysiä pisteitä.



Kuva 5.1: ChatGPT:n tekemä tyypillinen virhe. Kuvassa ChatGPT käyttää liukulukua kokonaisluvun sijaan ja sen takia simuloitu oppilas sai tästä palautuksesta nolla pistettä.



Kuva 5.2: Ensimmäisen palautuksella virheellisiä vastauksia oli yhteensä 34. Toiseen palautukseen edenneissä vastauksisa virheitä oli yhteensä 20. Kolmanteen palautukseen edenneissä vastauksissa virheitä oli yhteensä 15. Kuvaaja näyttää GPT-3.5:n virheellisten vastausten jakautumisen tyypeittäin. Tyypit ovat selitetty tarkemmin auki alaluvussa 5.2.

GPT-3.5 pohjautuva malli ei myöskään kykene ratkaisemaan tehokkaasti itse

sellaisia tehtäviä, jotka vaativat matemaattista ymmärrystä. GPT-4 osoitti huomattavasti enemmän kyvykkyyttä myös tällä osa-alueella. Tarkastellaan esimerkiksi kuvassa 5.3 olevaa ohjelmointitehtävää, jonka ratkaiseminen vaatii matemaattisia taitoja.

Viikko 6: 15.Korjaa ja kopioi laskut -tehtävänanto

Tiedostossa *laskut.csv* on yhteenlaskuja. Tiedosto on kuitenkin korruptoitunut - jokaisessa laskussa on jokin virhe.

Virheet voivat olla joko jommassa kummassa yhteenlaskettavassa tai tuloksessa (kuitenkin vain yhdessä näistä). Virheen seurauksena yksi alkuperäisistä luvuista laskussa on korvautunut satunnaisilla kirjaimilla.

Jos siis esim. lasku olisi ollut alunperin

2+4=6

...se voi nyt olla esim. muodossa

2+XY=6

tai

ZRQ+4=6

tai vaikkapa

2+4=UIJMN

Kirjoita funktio

korjaa_ja_kopioi() joka lukee laskut tiedostosta *laskut.csv*, korjaa ne alkuperäiseen muotoonsa ja kirjoittaa sitten korjattuna samassa järjestyksessä tiedostoon *korjatut_laskut.csv*

Kuva 5.3: Viikko 6: 15.Korjaa ja kopioi laskut -tehtävänanto.

Simuloitu GPT-3.5 hyödyntävä opiskelija ei pystynyt ratkaisemaan kyseistä tehtävää. GPT-3.5:ttä hyödyntävän opiskelijan vastaus nähdään kuvassa 5.4. Vaikka GPT-3.5 hyödyntävän oppilaan vastauksen koodi kääntyy, se ei toteuta tehtävän ratkaisuun vaadittua yksinkertaista laskuoperaatioiden logiikkaa. Vastausta tarkastelemalla havaitaan, että matematiikan laskusääntöjen noudattamisen sijaan GPT-3.5 hyödyntävän opiskelijan vastaus palauttaa korvattavan muuttujan kohdalla satunnaisluvun väliltä 1-100.

Viikko 6: 15.Korjaa ja kopioi laskut: ChatGPT 3.5 lopullinen vastaus

Tottakai, tässä on esimerkkiratkaisu annettuun kysymykseen ilman eval-moduulia:

```
def korjaa_ja_kopioi():
    # Avaa laskut.csv-tiedosto lukutilassa ja lue sen sisalto listalle.
    with open('laskut.csv', 'r') as csv_file:
        laskut = [line.strip().split(',') for line in csv_file]

    # Kay lapi kaikki laskut.
    for i in range(len(laskut)):
        # Etsi virheellinen luku (merkkijono) ja sen indeksi.
        lasku = laskut[i]
        virheellinen_luku = None
        virheellisen_luvun_indeksi = None
        for j in range(len(lasku)):
            if not lasku[j].isdigit():
                virheellinen_luku = lasku[j]
                virheellisen_luvun_indeksi = j
                break

        # Korvaa virheellinen luku satunnaisella luvulla.
        korvaava_luku = str(random.randint(1, 100))
        laskut[i][virheellisen_luvun_indeksi] = korvaava_luku

        # Laske oikea tulos korjatulla laskulla.
        summa = 0
        for k in range(len(lasku)):
            if k == virheellisen_luvun_indeksi:
                summa += int(korvaava_luku)
            else:
                summa += int(lasku[k])
        oikea_tulos = str(summa)

        # Korvaa korvattu luku alkuperaisella luvulla.
        laskut[i][virheellisen_luvun_indeksi] = virheellinen_luku

        # Korvaa virheellinen tulos oikealla tuloksella.
        laskut[i][-1] = oikea_tulos

    # Kirjoita korjatut laskut korjatut_laskut.csv-tiedostoon.
```

```
    with open('korjatut_laskut.csv', 'w') as csv_file:
        for lasku in laskut:
            csv_file.write(','.join(lasku) + '\n')
```

Tämä funktio käyttää perus Python-ohjelmointirakenteita kuten for-silmukoita ja ehtolauseita. Funktio käy läpi jokaisen laskun ja etsii virheellisen luvun (merkkijonon), joka korvataan satunnaisella luvulla. Sen jälkeen funktio laskee oikean tuloksen korjatulla laskulla manuaalisesti. Lopuksi funktio kirjoittaa korjatut laskut uuteen tiedostoon nimeltä korjatut_laskut.csv.

Kuva 5.4: ChatGPT 3.5 antama lopullinen vastaus tehtävänantoon *viikko 6: 15.Korjaa ja kopioi laskut.* Tehtävänanto on esitelty kuvassa 5.3.

Simuloitu GPT-4 mallia hyödyntävän opiskelijan lopullinen vastaus samaan kysymykseen nähdään kuvassa 5.5. GPT-4 antama lopullinen vastaus toimii täysin annetun tehtävänannon mukaisesti. Tehtävänratkaisu GPT-4:n avustuksella vaati kuitenkin enemmän kuin alkuperäisessä metodologiassa sallitut 3 yritystä.

Viikko 6: 15.Korjaa ja kopioi laskut: GPT-4 lopullinen vastaus

Pahoittelut jälleen virheestä. Oletetaan, että a_raw, b_raw ja c_raw sisältävät vähintään yhden numeron. Tässä on päivitetty versio korjaa_ja_kopioi() -funktiosta, joka huomioi tämän:

```
def korjaa_ja_kopioi():
    with open('laskut.csv', 'r') as input_file, \
         open('korjatut_laskut.csv', 'w') as output_file:
        for line in input_file:
            a_raw = line.strip().split('+')[0]
            b_raw = line.strip().split('=')[0]\
                        .split('+')[1]
            c_raw = line.strip().split('=')[1]

            # Poistetaan kirjaimet yhteenlaskettavista
            # ja tuloksesta.
            a_clean = int(''.join([
                            char for char in a_raw
                            if char.isdigit()
                        ]) or '0')
```

```
            b_clean = int(''.join([
                            char for char in b_raw
                            if char.isdigit()
                        ]) or '0')
            c_clean = int(''.join([
                            char for char in c_raw
                            if char.isdigit()
                        ]) or '0')

            # Korjataan lasku, jos siina on virhe.
            if a_clean + b_clean != c_clean:
                if not any(char.isdigit()
                           for char in b_raw):
                    b_clean = c_clean - a_clean
                elif not any(char.isdigit()
                             for char in c_raw):
                    c_clean = a_clean + b_clean
                else:
                    a_clean = c_clean - b_clean

            # Kirjoitetaan korjattu lasku tiedostoon.
            output_str = f'{a_clean}+{b_clean}={c_clean}\n'
            output_file.write(output_str)

korjaa_ja_kopioi()
```

Kuva 5.5: Simuloidun GPT-4 avusteisen oppilaan lopullinen vastaus tehtävään *viikko 6: 15.Korjaa ja kopioi laskut*. Tehtävänanto on esitelty kuvassa 5.3.

6 Johtopäätökset ja pohdinta

Tässä luvussa käydään läpi tutkimuksen johtopäätöksiä ja yleistä pohdintaa tutkimukseen liittyen. Alaluvussa 6.1 käydään läpi laajojen kielimallien vaikutusta ohjelmistokehitykseen yleisellä tasolla. Alaluvussa 6.2 käsitellään huijaamisen vaikeuttamista yleisellä tasolla. Tutkimuksessa käsiteltyä aineistoa myös laajennetaan havainnollistavalla esimerkillä eräältä toiselta ohjelmointikurssilta. Tämä esimerkki käydään läpi alaluvussa 6.3, jossa kyseistä tehtävää käytetään havainnollistamaan, että jo nykyiset laajat kielimallit pystyvät ratkaisemaan visuaalista ulostuloa vaativia tehtäviä. Alaluvussa 6.4 käydään läpi tutkimukseen liittyviä rajoitteita.

6.1 Laajojen kielimallien vaikutus ohjelmistokehitykseen

Laajat kielimallit ovat nostaneet ohjelmistokehittäjien tuottavuutta. Erityisesti GPT-4 on jo osoittanut kyvykkyyttä ratkaista erilaisia ongelmia ja osoittaa kykyä soveltaa oppimaansa [38] [50]. Kuitenkaan tekoälyn hyödyntäminen ohjelmistokehityksessä ei ole ongelmatonta. Harding et al. [51] ovat havainneet, että ohjelmointikoodin uudelleenkäyttö on vähentynyt AI-työkalujen käytön yleistymisen myötä. He myös arvioivat keräämänsä datan perusteella, että vuonna 2024 alle kaksi viikkoa sitten kirjoitettua ohjelmointikoodia muokkauskertojen määrä saattaa tuplaantua verrattuna vuoteen 2021 verrattuna. He toteavat todennäköiseksi syyksi generatiivista te-

koälyä hyödyntävien ohjelmointityökalujen yleistymisen. Tämä korostaa uhkakuvaa ohjelmistokehittäjien mahdollisesta liian vahvasta tukeutumisesta laajojen kielimallien tuottamaan ohjelmointikoodiin. Tämä voi johtaa esimerkiksi tietoturvaongelmiin ohjelmoijien kopioidessa sokeasti laajan kielimallin tuottamaa koodia tai jos he esimerkiksi luottavat kielimallin luomaan mahdollisesti virheelliseen tiivistelmään esimerkiksi jonkin ohjelmoijalle aiemmin tuntemattoman ohjelmointikielen ominaisuuksista. Näin ollen voidaan väittää, että laajojen kielimallien yleistyminen ei poista ohjelmoijan tarvetta ymmärtää koodia. Samoin tarve algoritmiselle ajattelulle tulee todennäköisesti säilymään. Nämä seikat johtavat luonnollisesti johtopäätökseen, että laadukasta ohjelmoinnin opetusta tarvitaan myös tulevaisuudessa.

On myös mainittava, että vaikka laajoja kielimalleja voi käyttää tuottamaan vastauksia suoraan, niitä voi käyttää myös oppimisen tukena. Tuutori, joka ei väsy, osaa avata käsitteiden merkityksiä monipuolisilla esimerkeillä ja kertoa suhteellisen tarkasti ohjelmointikoodissa olevista virheistä voi olla aloittelevalle ohjelmoijalle hyödyllinen apuri. Tämä korostaa esimerkiksi Gimpel et al. [1] korostamaa tarvetta opettaa opiskelijoille ja opettajille eettisiä ja kestäviä tapoja hyödyntää laajoja kielimalleja opetuksessa. Näitä näkökulmia käsiteltiin tarkemmin luvussa 3.

6.2 Huijausmetodien kiertäminen

On hyvä kysyä, voidaanko kielimalleilla huijaamista vaikeuttaa. Eräs mahdollisuus olisi käyttää visuaalisuutta vaativia tehtävänantoja. Kuitenkin jo nyt parhaimmat kielimallit osaavat vastata visuaalisuutta vaativiin tehtävänantoihin. Käytännön esimerkki tästä esitetään alaluvussa 6.3.

On myös muistettava, että laajojen kielimallien kyvykkyys ratkaista vaativia ongelmia on todennäköisesti lisääntynyt erilaisten työkalujen yleistymisen myötä. Esimerkiksi OpenAI on lisännyt ChatGPT:hen tuen liitännäisille[1] (eng. plugin). Sa-

[1] `https://openai.com/blog/chatgpt-plugins`

moin mikään ei myöskään estä esimerkiksi sisään kirjautuneen käyttäjän simulointia digitaalisella oppimisalustalla esimerkiksi saavutettavuuskerroksen kautta. Saavutettavuuskerroksen kautta kaikki tehtävän suorittamiseen tarvittava informaatio on oltava saatavilla myös selkeästi ymmärrettävän tekstin muodossa. Tämä todennäköisesti nostaa mallin suoriutumiskykyä esimerkiksi ohjelman suoritusta simuloivissa tehtävissä. Saavutetttavuuskerrosta ei myöskään (ymmärrettävästi) voida poistaa käytöstä. Samoin malleja, jotka ymmärtävät tekstiä, kuvia, ääntä ja videoita on alettu kehittämään. Konkreettisena esimerkkinä mainittakoon esimerkiksi Googlen kehittämä Gemini[2]. Geminille annettava syöte voi olla esimerkiksi tekstiä, kuvia, äänitiedosto tai video [52].

On myös huomioitava se, että chat-ominaisuuden omaavia laajoja kielimalleja kehitellään useiden eri organisaatioiden, yritysten ja yksilöiden toimesta. ChatGPT:n kilpailijana voidaan mainita esimerkiksi aiemmassa kappaleessa mainittu Gemini. Toinen maininnan arvoinen asia ovat avoimeen lähdekoodiin perustuvat kielimallit, joita kehitetään erilaisten yhteisöjen toimesta. Näistä esimerkkinä voidaan mainita esimerkiksi QLoRA. QLoRA on laaja kielimalli, jonka voi ajaa ja kouluttaa kuluttaja-tason laitteilla [53]. Se, että useita erilaisia malleja on olemassa ja se, että niiden tuottama teksti voi olla erilaista toisiinsa nähden asettaa ylimääräisen haasteen huijaamisen tunnistamiselle. Samoin esimerkiksi kielimallien hienosäätö voi lisätä haastetta eri kielimallien tuottaman tekstin luotettavalle tunnistamiselle.

6.3 Huijaamisen vaikeuttaminen tehtävänannolla

Tässä alaluvussa käsitellään huijaamisen vaikeuttamista konkreettisen esimerkin kautta. Kuten aiemmin on mainittu, laajojen kielimallien suoriutumista voidaan mahdollisesti yrittää vaikeuttaa niin, että kysymys sisältää visuaalisia elementtejä tai vaihtoehtoisesti tehtävään annettava vastaus on oltava visuaalisessa muodossa

[2] `https://gemini.google.com/`

esimerkiksi vuokaaviona. On kuitenkin huomattava, että jopa nykyisillä malleilla tämä lähestymistapa ei kuitenkaan (aina) toimi. Käytännön esimerkkinä tästä voidaan käyttää eräässä tähän tutkielmaan liittymättömässä kurssissa käytettävää tehtävänantoa, joka näkyy kuvassa 6.1.

Visuaalinen tehtävä: tehtävänanto

Oletetaan seuraavanlainen kontekstiton kielioppi
< Ohjelma > -> < Lauselista >
< Lauselista > -> < Lause >< LauseLista > | < tyhjä >
< Lause > -> < Asetuslause >
< Lause > -> < While-lause >
< Lause > -> < For-lause >
< Lause > -> < Metodikutsu >
< Asetuslause > -> < tunnus >< MathOper >< tunnus >
< While-lause > -> WHILE < Ehto >: < Lauselista > ENDWHILE
< For-lause > -> FOR (< Asetuslause >; < Ehto >; < Asetuslause >) < Lauselista > ENDFOR
< Metodikutsu > -> < tunnus > (< tunnus >)
< Ehto > -> < tunnus >< LooginenOper >< tunnus >
< LooginenOper > -> < | > | <= | >= | == | !=
< MathOper > -> += | -= | *= | /= | =
< tunnus > -> [a-öA-Ö]+ | [0-9]+

Minkälainen jäsennyspuu tulee allaolevalle ohjelmalle? Esitä se ASCII puuna

```
sum = 0
FOR (i=0;i<5;i++)
    sum += i
    print(sum)
ENDFOR
```

Kuva 6.1: Esimerkki tehtävänanto visuaalisuutta vaativasta tehtävästä.

GPT-4 generoima ASCII-puu on nähtävissä kuvassa 6.2. GPT-4:n antamaa vastausta kyseiseen kysymykseen tarkasteltaessa voidaan huomata, että myös pelkkää

tekstiä ulostulonaan käyttävä GPT-4 osaa vastata visuaalisuutta vaativiin kysymyksiin käyttämällä esimerkiksi ASCII:hin pohjautuvaa graafisen puun visualisaatiota.

Visuaalinen tehtävä: GPT-4 vastauksen ASCII-puu

```
<Ohjelma>
|
+- <Lauselista>
   |
   +- <Lause>
   |   |
   |   +- <Asetuslause>
   |       |
   |       +- <tunnus> "sum"
   |       +- <MathOper> "="
   |       +- <tunnus> "0"
   |
   +- <Lauselista>
       |
       +- <Lause>
       |   |
       |   +- <For-lause>
       |       |
       |       +- <Asetuslause> "i=0"
       |       +- <Ehto> "i<5"
       |       +- <Asetuslause> "i++"
       |       +- <Lauselista>
       |           |
       |           +- <Lause>
       |           |   |
       |           |   +- <Asetuslause>
       |           |       |
       |           |       +- <tunnus> "sum"
       |           |       +- <MathOper> "+="
       |           |       +- <tunnus> "i"
       |           |
       |           +- <Lauselista>
       |               |
       |               +- <Lause>
       |                   |
       |                   +- <Metodikutsu>
       |                       |
       |                       +- <tunnus> "print"
       |                       +- <tunnus> "sum"
       |
       +- <Lauselista> <tyhjä>
```

Kuva 6.2: Visuaalinen tehtävä: ChatGPT-4 mallin vastauksen tuottama ASCII-puu.

GPT-3.5 osasi myös tuottaa ASCII-puu visualisaation, mutta sen vastaus ainakin esimerkkinä käytettyyn tehtävään oli väärin, kuten nähdään kuvassa 6.3.

Visuaalinen tehtävä: GPT-3.5 vastauksen ASCII-puu.

```
    <Ohjelma>
       |
   <Lauselista>
       |
     <Lause>
       |
   <Asetuslause>
       |
    <tunnus>       <MathOper>    <tunnus>
       |              |             |
      sum             =             0
       |
     <For-lause>
       |
  FOR (<Asetuslause>; <Ehto>; <Asetuslause>)
       |
     <Lauselista>
       |
      <Lause>
       |
    <Asetuslause>
       |
     <tunnus>       <MathOper>    <tunnus>
        |              |             |
        i              =             0
       |
     <Ehto>
    /     |     <tunnus> | <LooginenOper> <tunnus>
   |      |        |          |
   i      <        5          i
       |
     <Lauselista>
       |
      <Lause>
       |
    <Metodikutsu>
       |
     <tunnus>   <tunnus>
        |          |
     print      sum
```

Kuva 6.3: Visuaalinen tehtävä: ChatGPT-3.5 mallin tuottama ASCII-puu.

Mikäli kysymystä tarkastellaan lähemmin, huomataan että ChatGPT:n tuottama vastaus on väärin. Kumpikaan ChatGPT:n versio (GPT-3.5:n tai GPT-4 pohjautuva malli) ei huomannut, että tehtävänannon esittämän esimerkkiohjelman syntaksi oli virheellinen. Tehtävänannossa käytettyä muuttujaa `i++` ei ole määritelty annetuissa säännöissä, joten oikea vastaus esimerkkitehtävässä 6.1 esitettyyn kysymykseen on, että ohjelma ei käänny.

On huomioitava, että mikäli oppilas osaisi kysyä tätä kysymystä suoraan, GPT-4 onnistuisi päättelemään oikean vastauksen. Tämä kuitenkin vaati, että alkuperäisen kysymyksen kehotteen osa

- *Minkälainen jäsennyspuu tulee allaolevalle ohjelmalle? Esitä se ASCII puuna*

korvattiin seuraavanlaisella kehotteella:

- *Onnistuuko alla olevan koodin kääntäminen ylläolevan kontekstittoman kieliopin mukaan? Perustele kantasi ja sen jälkeen vastaa kyllä tai ei..*

GPT-4:n antama vastaus yllä olevaan kehotteeseen on nähtävissä kuvassa 6.4.

Visuaalinen tehtävä: Kääntyykö koodi GPT-4 vastaus

- Koodi alkaa *asetuslauseella* $sum = 0$, joka vastaa kieliopin $<$ *Asetuslause* $>$-sääntöä. Tässä *sum* ja 0 ovat $<$ *tunnus* $>$ ja $=$ on $<$ *MathOper* $>$.
- Seuraavaksi tulee $<$ *For*$-$*lause* $>$ $FOR(i = 0; i < 5; i{+}{+})$ $<$ *lauselista* $>$ $ENDFOR$, joka vastaa kieliopin $<$ *For* $-$ *lause* $>$-sääntöä. Tässä:
  - Ensimmäinen $<$ *Asetuslause* $>$ on $i = 0$, joka vastaa kieliopin $<$ *Asetuslause* $>$-sääntöä.
  - $<$ *Ehto* $>$ on $i < 5$, joka vastaa kieliopin $<$ *Ehto* $>$-sääntöä. Tässä kieliopissa ei ole kuitenkaan erityistä symbolia $<$ operaattorille, joka on tässä ongelmallista.
  - Toinen $<$ *Asetuslause* $>$ on $i + +$, joka ei vastaa kieliopin $<$ *Asetuslause* $>$-sääntöä, koska siinä ei ole erityistä symbolia $++$ operaattorille.
  - $<$ *Lauselista* $>$ sisältää kaksi lausetta: $sum+ = i$ ja $print(sum)$. Ensimmäinen lause vastaa *Asetuslause*-sääntöä, mutta toinen lause ei vastaa mitään kieliopin $<$ *Lause* $>$-sääntöä, koska siinä ei ole erityistä sääntöä printtauslauseelle.

Joten annettu koodi ei käänny onnistuneesti tämän kieliopin mukaan, koska jotkut operaattorit (kuten $<$ ja $++$) ja metodit (kuten *print*) puuttuvat kieliopista.

Vastaus: Ei.

Kuva 6.4: GPT-4:n antama vastaus kuvassa 6.1 esitellylle tehtävänannolle. GPT-4:lle annettu muokattu kehote on esitelty luvussa 6.3. Vastaus ei ole täysin oikein, koska$<$ operaattori on määritelty kieliopissa. Kuitenkin GPT-4:n antama vastaus on osittain oikein, koska $i{+}{+}$ määritelmä puuttuu esitellystä kieliopista.

Opiskelijan pitäisi siis luoda oikeanlainen kysymyksenasettelun ja ns. kysyä oikea kysymys alkuperäisen tehtävänannon toistamisen sijaan. Se siis vaatisi, että opiskelija ymmärtää esitetyn ongelman ja sen mahdolliset ratkaisutavat tehtävänannon sokean kopioimisen sijaan.

Voidaankin myös todeta, että vaikka tehtävänannon muokkaamisella tehtävään vastaamista laajaa kielimallia voidaan vaikeuttaa, on se todennäköisesti vain väliaikainen ratkaisu. Samoin laajojen kielimallien käytön yleistyessä on luultavaa, että mallien käyttäjien (tässä tapauksessa opiskelijoiden) ymmärrys siitä, kuinka mallia voi käyttää tehokkaasti, voidaan olettaa lisääntyvän tulevaisuudessa. Samoin ottaen huomioon mallien nopean kehityksen, voidaan olettaa mallien tehokkuuden ja kyvykkyyden tehdä erilaisia tehtäviä jatkavan kasvuaan tai vähintään pysyvän samana.

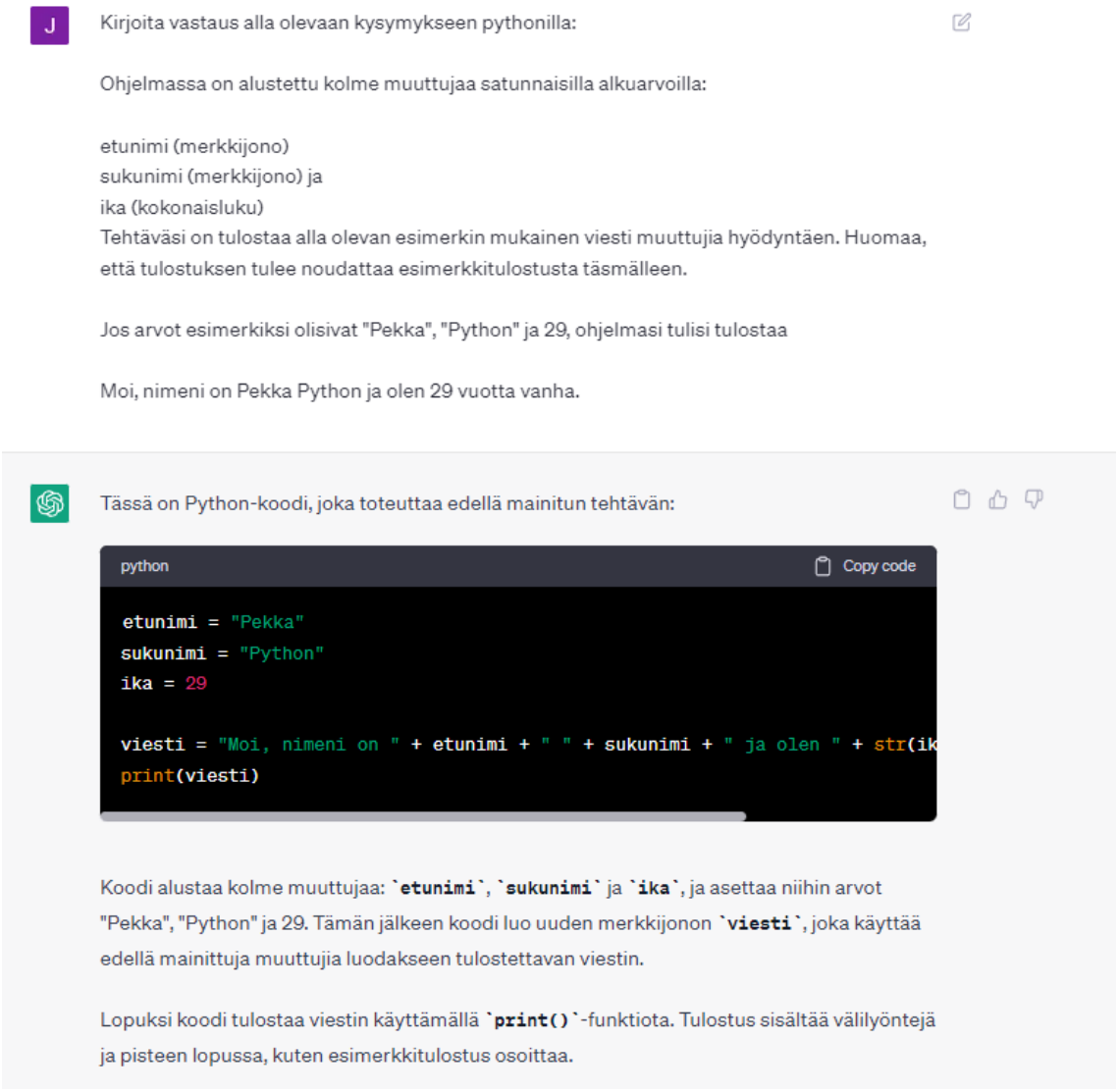
6.4 Tutkimuksen rajoitteet

On myös havaittava, että jotkin ohjelmointitehtävät vaativat myös loogista päättelykykyä. GPT-3.5 pohjautuvalla mallilla on vaikeuksia suoriutua monimutkaisista tai loogista ajattelua vaativista tehtävistä [50].

Voidaan myös spekuloida, että tarjoamalla ChatGPT:lle enemmän tietoa käytetystä alustasta ja sen rajoitteista, esimerkiksi siitä, että ViLLE estää ulkoisten kirjastojen käytön jo alkuperäisessä kehotteessa, on luultavaa, että ChatGPT olisi osannut ratkaista isomman osan tehtävistä. ViLLE:n Pythonin ohjelmointiympäristössä ulkoisten moduulien käyttö on rajoitettua vain tiettyihin ennalta määriteltyihin moduuleihin. Lisäksi kurssilla, jonka tehtäviä käytettiin tämän tutkimuksen pohjana useimmissa tehtävissä ulkoisten moduulien käyttöä ei sallittu ollenkaan. Tämä asetti ChatGPT:lle ylimääräisen haasteen tuottaa tehtävänannon täyttävä koodi ilman ylimääräisiä moduuleita.

Lisäksi osassa tehtävistä ChatGPT:n tuottamasta vastauksesta koodi kopioitiin

vain osittain. Mikäli ChatGPT oli luonut kopioitavaan vastaukseen muuttujan, jossa oli kuvitteellinen esimerkkisyöte, se jätettiin pois kopioidusta vastauksesta. Esimerkki tästä nähdään kuvassa 6.5. Jos ChatGPT:n vastaus sisälsi tehtävässä käytetyn moduulin *import*-lausekkkeen, kyseinen rivi jätettiin kopioimatta vastauksesta. Tämä oli yksi käytetyistä lähestymistavoista, koska yksi tutkimuksen päämääristä oli simuloida, kuinka oppilas, jolla on vähäiset ohjelmoinnin taidot selviytyisi kurssin tehtävistä ChatGPT:hen vahvasti tukeutuen. Täysin tietämättömän opiskelijan simuloinnissa ChatGPT:n vastaus kopioitiin sellaisenaan ilman mitään muutoksia. Täysin tietämättömän opiskelijan simulointia käydään tarkemmin läpi luvussa 5.1.3.



Kuva 6.5: ChatGPT alustaa tuottamassa ohjelmakoodissa myös muuttujat, vaikka ne olivat jo määritelty tehtävänannon mukaan. Muuttujat etunimi, sukunimi ja ikä jätettiin pois ViLLE:en kopioidusta vastauksesta, joten aineistoissa tämä tehtävä merkattiin oikein tehdyksi. Täysin tietämättömän opiskelijan simuloinnissa tämän vastauksen kopioinut simuloitu opiskelija sai tehtävästä nolla pistettä.

Tutkimuksessa hyödynnettiin pääasiassa GPT-3.5 hyödyntävää versiota ChatGPT:stä. Kesken tutkimuksen teon OpenAI julkaisi GPT-4-mallia hyödyntävän ChatGPT:n, joten tutkimusta päätettiin laajentaa myös kattamaan GPT-4 mallia hyödyntävä ChatGPT. Yleisesti ottaen GPT-4 suoriutuu ohjelmointiin liittyvistä suorituskyky-

testeistä prosentuaalisesti mitattuna joko samantasoisesti tai paremmin kuin GPT-3.5 pohjautuva malli [50] [50]. Samanlainen johtopäätös on tehtävissä myös yleisesti liittyen laajojen kielimallien suorituskyvyn skaalautumiseen [8]. Vaikka emme pysty täysin takaamaan, että GPT-4 kykenee ratkaisemaan täsmälleen samat ohjelmistotehtävät kuin GPT-3.5, edellä esitettyjen seikkojen valossa tehtiin päätös antaa GPT-4 hyödyntävän simuloidun oppilaan ratkaista vain ne tehtävät, joissa aiempi versio (GPT-3.5) epäonnistui kokonaan tai osittain. Koska GPT-4:n antama vastaus vaikutti edistyvän, GPT-4 mallia hyödyntävän simuloidun oppilaan annettiin myös joissakin ohjelmointitehtävissä antaa enemmän jatkokehotteita kuin GPT-3.5:ttä hyödyntävän oppilaalle asetettu kahden jatkokehotetteen maksimi.

7 Yhteenveto

Laajat kielimallit tulevat vaikuttamaan hyvin merkittävästi ohjelmoinnin opetukseen. Kerrataan vielä tutkimuksen tutkimuskysymykset ja niiden lopputulokset.

TK1: Kykeneekö ChatGPT:tä hyödyntävä alkeelliset taidot omaava opiskelija läpäisemään ohjelmoinnin peruskurssin?
Kuten tutkimuksessa todettiin, oppilaat voivat läpäistä ohjelmoinnin perusteet kurssin osaamatta ollenkaan ohjelmoida. Tehdyssä tutkimuksessa simuloitiin oppilasta, joka hyödyntää ChatGPT:tä ohjelmoinnin tehtävien ratkaisuun. Simuloidun oppilaan taitotaso vaihteli täysin tietämättömän ja jonkinasteisen tietämyksen välillä. Tutkimuksessa ohjelmointia hieman taitavan opiskelijan kurssilla saama kokonaispistemäärä vaihtelee, käytetystä metodologiasta ja GPT-mallista riippuen välillä 65.9%-86.2%. Tarkemmat luvut ovat nähtävillä taulukossa 5.1.

TK2: Voiko täysin tietämätön opiskelija läpäistä ohjelmoinnin peruskurssin osaamatta ollenkaan ohjelmoida?
Täysin ohjelmoinnista tietämätön opiskelija voi läpäistä ohjelmoinnin kurssin. Täysin ohjelmoinnista tietämättömän opiskelijan simuloinnissa simuloitu opiskelija pystyi läpäisemään kurssin 63.4% kurssin kokonaispisteistä.

TK3: Mitkä ovat ChatGPT:n tekemät yleisimmät virheet ohjelmoinnin

perusteiden tehtävissä?

ChatGPT:n yleisimmät virheet liittyivät tulostuksen muotoiluun. Kuitenkin valtaosa näistä tehtävistä vaikuttivat olevan loogisesti oikein. Tarkemman virheellisten tehtävien jaottelun voi nähdä kuvaajasta 5.2.

TK4: Kuinka helposti opiskelija voi huijata kursseilla ja onko mahdollista kiertää ChatGPT:n kykyä ratkaista tehtäviä esimerkiksi vaihtamalla tehtävätyyppiä tai kysymyksenasettelua?

Keinot valvoa ja tunnistaa laajojen kielimallien käyttöä voi osoittautua jopa mahdottomaksi. Tätä käsiteltiin luvussa 3.7. Lisäksi mallien hyödyllisyyden takia ne myös todennäköisesti ovat olennainen osa ohjelmoijan käyttämiä työkaluja tulevaisuudessa. Tätä korostaa simuloidun oppilaan korkeahko ratkaisuprosentti tutkimuksessa käytetyissä ohjelmointitehtävissä (taulukko 5.2). Näiden seikkojen valossa voidaan väittää, että ainoaksi keinoaksi pitkällä tähtäimellä hillitä huijaamista laajojen kielimallien avulla on se, että oppilaille on painotettava eettisiä työtapoja ja lähteiden merkitsemisen tärkeyttä. Luvuissa 6.2 ja 6.3 käsiteltiin tarkemmin mahdollisuuksia kysymyksenasettelun muuttamiseksi niin, että laaja kielimalli ei pystyisi antamaan oppilaalle suoraan oikeaa vastausta. Kuten luvuissa todettiin, oletuksena on, että tehtävätyyppien muokkauksilla päästään vain rajallisiin tuloksiin tekniikan kehittyessä. Samoin on pantava merkille, että erityisesti ohjelmoinnin opetuksessa tämä voi osoittautua mahdottomaksi tehtäväksi, sillä ohjelmoinnin oppimiseksi sitä on myös harjoiteltava käytännössä.

Lähdeluettelo

[1] H. Gimpel, K. Hall, S. Decker et al., ”Unlocking the power of generative AI models and systems such as GPT-4 and ChatGPT for higher education: A guide for students and lecturers”, Hohenheim Discussion Papers in Business, Economics ja Social Sciences, tekninen raportti, 2023.

[2] Z.-H. Zhou, *Machine learning*. Springer Nature, 2021.

[3] Y. LeCun, Y. Bengio ja G. Hinton, ”Deep learning”, *nature*, vol. 521, nro 7553, s. 436–444, 2015.

[4] Z. John Lu, ”The elements of statistical learning: data mining, inference, and prediction”, teoksessa Oxford University Press, 2010, luku 11, s. 389–416.

[5] S. M. Weiss, N. Indurkhya, T. Zhang, S. M. Weiss, N. Indurkhya ja T. Zhang, ”Overview of text mining”, *Fundamentals of Predictive Text Mining*, s. 1–12, 2010.

[6] S. M. Weiss, N. Indurkhya, T. Zhang, S. M. Weiss, N. Indurkhya ja T. Zhang, ”Overview of text mining”, *Fundamentals of Predictive Text Mining*, s. 16–17, 2010.

[7] D. Jurafsky ja J. H. Martin, *Speech and language processing (3rd draft ed.), 2023.*

[8] J. Kaplan, S. McCandlish, T. Henighan et al., ”Scaling laws for neural language models”, *arXiv preprint arXiv:2001.08361*, 2020.

[9] T. B. Brown, B. Mann, N. Ryder et al., *Language Models are Few-Shot Learners*, 2020. arXiv: `2005.14165 [cs.CL]`.

[10] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need", *Advances in neural information processing systems*, vol. 30, 2017.

[11] J. Devlin, M.-W. Chang, K. Lee ja K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[12] J. Maynez, S. Narayan, B. Bohnet ja R. McDonald, "On faithfulness and factuality in abstractive summarization", *arXiv preprint arXiv:2005.00661*, 2020.

[13] L. Ouyang, J. Wu, X. Jiang et al., "Training language models to follow instructions with human feedback", *Advances in Neural Information Processing Systems*, vol. 35, s. 27 730–27 744, 2022.

[14] J. Wei, X. Wang, D. Schuurmans et al., "Chain-of-thought prompting elicits reasoning in large language models", *Advances in Neural Information Processing Systems*, vol. 35, s. 24 824–24 837, 2022.

[15] S. Yao, D. Yu, J. Zhao et al., "Tree of thoughts: Deliberate problem solving with large language models", *arXiv preprint arXiv:2305.10601*, 2023.

[16] D. Hulbert, *Tree of Knowledge: ToK aka Tree of Knowledge dataset for Large Language Models LLM*, `https://github.com/dave1010/tree-of-thought-prompting`, 2023.

[17] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever et al., "Improving language understanding by generative pre-training", 2018.

[18] Y. Zhu, R. Kiros, R. Zemel et al., "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books", teoksessa *Proceedings of the IEEE international conference on computer vision*, 2015, s. 19–27.

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever et al., "Language models are unsupervised multitask learners", *OpenAI blog*, vol. 1, nro 8, s. 9, 2019.

[20] C. Raffel, N. Shazeer, A. Roberts et al., "Exploring the limits of transfer learning with a unified text-to-text transformer", *The Journal of Machine Learning Research*, vol. 21, nro 1, s. 5485–5551, 2020.

[21] S. Gehman, S. Gururangan, M. Sap, Y. Choi ja N. A. Smith, "Realtoxicityprompts: Evaluating neural toxic degeneration in language models", *arXiv preprint arXiv:2009.11462*, 2020.

[22] OpenAI, *Introducing ChatGPT*, `https://openai.com/blog/chatgpt`, Accessed: June 21, 2023, 2023.

[23] T. Wu, S. He, J. Liu et al., "A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development", *IEEE/CAA Journal of Automatica Sinica*, vol. 10, nro 5, s. 1122–1136, 2023. DOI: `10.1109/JAS.2023.123618`.

[24] J. Deng ja Y. Lin, "The benefits and challenges of ChatGPT: An overview", *Frontiers in Computing and Intelligent Systems*, vol. 2, nro 2, s. 81–83, 2022.

[25] Kanika, S. Chakraverty ja P. Chakraborty, "Tools and techniques for teaching computer programming: A review", *Journal of Educational Technology Systems*, vol. 49, nro 2, s. 170–198, 2020.

[26] S. Wang, C. Christensen, W. Cui et al., "When adaptive learning is effective learning: comparison of an adaptive learning system to teacher-led instruction", *Interactive Learning Environments*, vol. 31, nro 2, s. 793–803, 2023.

[27] S. Sun, N. M. Else-Quest, L. C. Hodges, A. M. French ja R. Dowling, "The effects of ALEKS on mathematics learning in K-12 and higher education: A meta-analysis", *Investigations in Mathematics Learning*, vol. 13, nro 3, s. 182–196, 2021.

[28] B. Nye, D. Mee ja M. G. Core, "Generative large language models for dialog-based tutoring: An early consideration of opportunities and concerns", teoksessa *AIED Workshops*, 2023.

[29] R. Saikkonen, L. Malmi ja A. Korhonen, "Fully Automatic Assessment of Programming Exercises", *SIGCSE Bull.*, vol. 33, nro 3, s. 133–136, kesäkuu 2001, ISSN: 0097-8418. DOI: 10.1145/507758.377666. url: https://doi.org/10.1145/507758.377666.

[30] Z. Ullah, A. Lajis, M. Jamjoom, A. Altalhi, A. Al-Ghamdi ja F. Saleem, "The effect of automatic assessment on novice programming: Strengths and limitations of existing systems", *Computer Applications in Engineering Education*, vol. 26, nro 6, s. 2328–2341, 2018.

[31] U. of Turku, *ViLLE*, https://ville.utu.fi/, Avattu: 10.8.2023, 2023.

[32] M.-J. Laakso, E. Kaila ja T. Rajala, "ViLLE–collaborative education tool: Designing and utilizing an exercise-based learning environment", *Education and Information Technologies*, vol. 23, s. 1655–1676, 2018.

[33] o. t. Turun Yliopisto, *Kaksivuotisen esiopetuksen kokeilun tutkimus*, https://www.oppimisanalytiikka.fi/hankeinfot/eskarikokeilu/, Accessed: October 10, 2023, 2023.

[34] T. Rajala, E. Kaila, R. Lindén et al., "Automatically assessed electronic exams in programming courses", teoksessa *Proceedings of the Australasian computer science week multiconference*, 2016, s. 1–8.

[35] L. Yan, L. Sha, L. Zhao et al., "Practical and ethical challenges of large language models in education: A systematic scoping review", *British Journal of Educational Technology*, vol. 55, nro 1, s. 90–112, 2024.

[36] E. Kasneci, K. Seßler, S. Küchemann et al., "ChatGPT for good? On opportunities and challenges of large language models for education", *Learning and Individual Differences*, vol. 103, s. 102 274, 2023.

[37] S. MacNeil, A. Tran, D. Mogil, S. Bernstein, E. Ross ja Z. Huang, "Generating diverse code explanations using the gpt-3 large language model", teoksessa *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 2*, 2022, s. 37–39.

[38] S. Bubeck, V. Chandrasekaran, R. Eldan et al., "Sparks of artificial general intelligence: Early experiments with gpt-4", *arXiv preprint arXiv:2303.12712*, 2023.

[39] J. Savelka, A. Agarwal, M. An, C. Bogart ja M. Sakr, "Thrilled by Your Progress! Large Language Models (GPT-4) No Longer Struggle to Pass Assessments in Higher Education Programming Courses", *arXiv preprint arXiv:2306.10073*, 2023.

[40] J. Finnie-Ansley, P. Denny, A. Luxton-Reilly, E. A. Santos, J. Prather ja B. A. Becker, "My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises", teoksessa *Proceedings of the 25th Australasian Computing Education Conference*, 2023, s. 97–104.

[41] OpenAI, *OpenAI Codex*, `https://openai.com/blog/openai-codex`, Accessed: December 1, 2023, 2023.

[42] M. Peeperkorn, T. Kouwenhoven, D. Brown ja A. Jordanous, "Is temperature the creativity parameter of large language models?", *arXiv preprint arXiv:2405.00492*, 2024.

[43] J. Finnie-Ansley, P. Denny, B. A. Becker, A. Luxton-Reilly ja J. Prather, "The robots are coming: Exploring the implications of openai codex on introductory

programming", teoksessa *Australasian Computing Education Conference*, 2022, s. 10–19.

[44] A. Pegoraro, K. Kumari, H. Fereidooni ja A.-R. Sadeghi, "To ChatGPT, or not to ChatGPT: That is the question!", *arXiv preprint arXiv:2304.01487*, 2023.

[45] S. Mitrović, D. Andreoletti ja O. Ayoub, "Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text", *arXiv preprint arXiv:2301.13852*, 2023.

[46] OpenAI, *How can educators respond to students presenting AI-generated content as their own?*, `https://help.openai.com/en/articles/8313351-how-can-educators-respond-to-students-presenting-ai-generated-content-as-their-own`, Accessed: September 9, 2023, 2023.

[47] V. S. Sadasivan, A. Kumar, S. Balasubramanian, W. Wang ja S. Feizi, "Can ai-generated text be reliably detected?", *arXiv preprint arXiv:2303.11156*, 2023.

[48] K. Dooley, "Simulation research methods", s. 829–848, tammikuu 2002.

[49] *TKO_2110 Ohjelmoinnin perusteet*, `https://opas.peppi.utu.fi/fi/opintojakso/TKO_2110/20408?period=2022-2024`, Accessed: 13.04.2023.

[50] J. Achiam, S. Adler, S. Agarwal et al., "Gpt-4 technical report", *arXiv preprint arXiv:2303.08774*, 2023.

[51] W. Harding ja M. Kloster, "Coding on Copilot: 2023 Data Shows Downward Pressure on Code Quality", GitClear, White Paper, tammikuu 2024, 150m lines of analyzed code + projections for 2024. url: `https://www.gitclear.com/coding_on_copilot_data_shows_ais_downward_pressure_on_code_quality`.

[52] G. Team, R. Anil, S. Borgeaud et al., "Gemini: a family of highly capable multimodal models", *arXiv preprint arXiv:2312.11805*, 2023.

[53] T. Dettmers, A. Pagnoni, A. Holtzman ja L. Zettlemoyer, ”QLoRA: Efficient Finetuning of Quantized LLMs”, *arXiv preprint arXiv:2305.14314*, 2023.

Liite A Sanasto

Sanasto	
suomi	**englanti**
Ajatusketju	Chain-of-Thought
Ajatuspuu	Tree of Thoughts
Automaattinen hakurobotti	Web crawler
Hienosäätö	Fine-Tuning
Ihmisen palautteeseen perustuva vahvenneoppiminen	Reinforcement Learning from Human Feedback
Itsehuomiomekanismi	Self-attentintion mechanism
Kehote	Prompt
Lämpötila	Temperature
Liitännäinen	Plugin
Muutama-Esimerkki	Few-Shot
Nolla-Esimerkki	Zero-Shot
Takaisinkytketty neuroverkko	Recurrent neural network
Tekstialkio	Token
Tietojoukko	Data set
Tila	State
Tulostumaton merkki	White space
Tuutori	Tutor

Viestiketju	Thread