Ari Häyrinen

Koodarin AI-renki – ja toivottavasti pysyy sellaisena

Mitä jos käytössäsi olisi renki, jolle voisit antaa tehtäväksi kaikki tylsät tehtävät? Sen lisäksi renki tuntisi työsi ja voisit pyytää sitä tarkastamaan omaa työtäsi ja pyytää korjausehdotuksia. Lisäksi renki olisi vaatimaton, ei kiukuttelisi ja olisi aina töissä. Ja ennen kaikkea, se osaisi hommansa. Juuri tällainen kaveri on nykyaikainen koodiavustin. Miten pitkälle se voi minut avustaa? Ja voiko se tulevaisuudessa suositella minulle toista työpaikkaa?

Koodaaminen on (ainakin toistaiseksi) kirjoittamista. Koodin kirjoittaminen on myös aivan tietynlaista kirjoittamista. Syntaksi on ehdoton, sanasto on rajallinen, pilkuilla ja pisteillä on todella väliä ja jopa sisennysten oltava (tietyissä kielissä) määrämittaisia.

stablediffusionweb.com

Kielimallit ja koneiden kielet

Koodin kirjoittaminen ei ole luovaa kirjoittamista. Suuri osa työstä on samojen rimpsujen kirjoittamista yhä uudelleen ja uudelleen. Liian luovat ratkaisut tekevät koodin ymmärtämisestä muille hankalaa ja hyvin villit ratkaisut ovatkin joko erittäin kokeneiden ohjelmoijien tekemiä tai sitten ne tulevat aivan aloittelijoiden käsistä. Yleensä on parasta tyytyä useimmiten käytettyihin ratkaisuihin ja käytetyn kielen vakiintuneisiin tapoihin.

Siksi ei olekaan yllättävää, että koodaamiseen on ollut jo hyvin kauan erilaisia avustimia, jotka auttavat syntaksin kanssa. Rajoitetun sanavaraston ja tiukkojen sääntöjen vuoksi avustimia on ollut mahdollista luoda perinteisillä ohjelmointimenetelmillä. Nämä avustajat eivät kuitenkaan missään mielessä ymmärtäneet, mitä koodin oli tarkoitus tehdä. Syntaksi itsessään on vain kieliopin oikeellisuutta ja surkeakin, mutta toimiva ohjelma on syntaktisesti oikein.

Suurten kielimallien esiin rymistelyn myötä on käynyt selväksi, että kielimallit toimivat erinomaisesti myös koneiden kielten – ohjelmointiin tar-

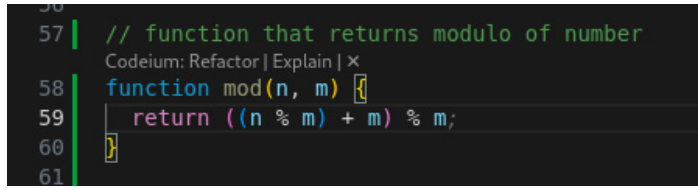
koitettujen kielten – kanssa. Koodiavustimet ottivat loikan syntaksista semantiikkaan. Ne osaavat kääntää luonnollista puhetta ohjelmakoodiksi ja ne osaavat ottaa huomioon muun ympärillä olevan koodin. Miten se käytännössä toimii ja kuinka se vaikuttaa minuun?

Kone, tee minulle ISBN-numero!

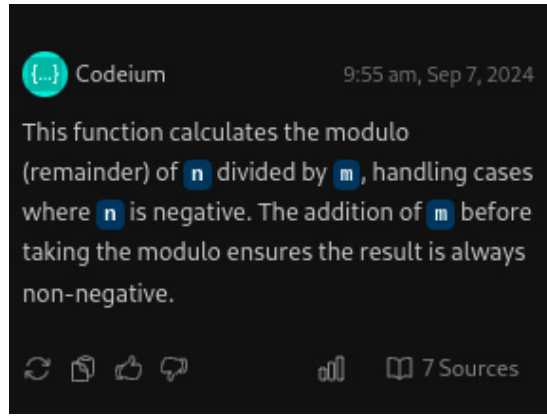
Kuvitellaan, että olen tekemässä ohjelmaa ISBN-numeroiden myöntämiseen väitöskirjan tekijöille. Ohjelmani tarvitsee muutaman toiminnon (function): tarvitsen funktion, joka antaa käsiteltyjen tietueiden lukumäärän jakojäännöksen ja toiminnon, joka tuottaa minulle oikeilta näyttäviä ISBN-numeroita testaamista varten.

En muista ulkoa miten jakojäännös nyt saatiinkaan (muistan vain englanninkielisen nimen ”modulo”), joten jätän funktion luomisen avustimen tehtäväksi. Kirjoitan koodieditoriin kuvauksen funktiosta:
// function that returns modulo of number

Sen jälkeen painan enteriä, minkä jälkeen avustin ehdottaa funktion nimeksi ”mod” ja että se saa kaksi lukua, jaettavan ja jakajan (n, m). Painan tabulaattoria hyväksyäkseni funktion nimen



Tulos näyttää hieman monimutkaisemmalta kuin odotin, joten pyydän avustinta selittämään koodin. Klikkaamalla ”explain” -linkkiä funktion yläpuolella ja saan seuraavan vastauksen:



Niinpä, olisin unohtanut negatiivisten lukujen käsittelyn, millä ei tosin tässä tapauksessa ole merkitystä, mutta hyvä muistutus joka tapauksessa.

Koko prosessi kesti muutaman sekunnin, minun ei tarvinnut poistua kehitysympäristöstä vaan haltiamainen avustin teki sen mitä halusin. Juuri tähän ”teki sen mitä halusin”-toimintoon kietoutuu koko tekoälyavusteisen työskentelyn idea. Olisin pystynyt tekemään sen, mitä pyysin tekoälyä tekemään, mutta tekoäly teki sen paljon nopeammin.

Seuraavaksi tarvitsisin funktion, joka tuottaa ISBN-numeron testaamista varten. Aloitan taas kirjoittamalla kommentin, joka kuva mitä haluan:

// function that creates random ISBN number

Painan enteriä, minkä jälkeen minulle ilmestyy ehdotus funktion nimestä (generateISBN) ja kun painan tabulaattoria, ehdotus ilmestyy editoriin ja funktion loppuosa täydentyy.

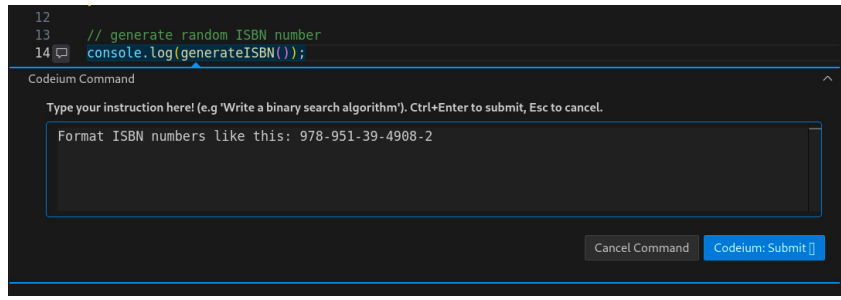
```
// function that creates random ISBN number
function generateISBN() {
 let ISBN = '';
 for (let i = 0; i < 13; i++) {
 ISBN += Math.floor(Math.random() * 10);
 }
 return ISBN;
}

// generate random ISBN number
console.log(generateISBN());
```

Huom! Tätä voit testata itsekin selaimessasi! Avaa developer console ja kopioi yllä oleva teksti sinne ja paina enter.

Kun suoritan funktion, saan seuraavan tuloksen: 7751295563019

Huomaa, että en kertonut avustajalle mikä ISBN-numero on. Tuloksesta kuitenkin näkee, että avustajalla on jonkinlainen käsitys ISBN-numerosta. Merkkejä on 13, kuten pitääkin. Väliviivoja ei ole, mutta ne eivät olekaan pakollisia. Tunnus ei kuitenkaan ala oikeilla numeroilla ja haluaisin väliviivat mukaan, koska ne tekevät tunnuksista luettavampia.

Voin edelleen kehittää funktiota valitsemalla koodiketkstin ja painamalla Ctrl + I. Näkyviin tulee ikkuna, johon voin kirjoittaa lisäohjeita siitä, miten funktiota tulee muuttaa. Kirjoitan seuraavasti:



Suoritan jälleen funktion ja tulos näyttää tältä: 978-376-0210-727

Väliviivat tulivat mukaan, mutta hieman satunnaisella jaksotuksella. Avaan samaisen kehote-ikkunan ja annan askel askeleelta etenevän tavan muodostaa ISBN-numero:

Create isbn as follows, step by step:
1. start with 978-
2. then add three digits three times separated by hyphen
3. lastly, add one digit separated by hyphen

Näillä kehotteilla saan tuloksesi 978-111-143-052-6. Viimeinkin! Numerot näyttävät tarpeeksi oikeilta ISBN-numeroilta. Nyt avustin siis tietää, miltä ISBN-numero *näyttää*, mutta se ei edelleenkään ymmärrä, miten se muodostetaan. Viimeinen numero on tarkistusnumero ja se ei siis tietenkään voi kuulua arvottaviin numeroihin.

Lisää korjauksia

Lähden korjaamaan tilannetta kertomalla funktiolle, miten tarkistusnumero lasketaan. Pienen googletuksen jälkeen löydän sanallisesti kuvatun algoritmin ja annan sen avustimelle ohjeeksi funktion uudelleen muotoilua varten.

An ISBN-13 check digit ranges from 0 to 9, and is computed using similar steps.

- Multiply each of the first 12 digits by 1 or 3, alternating as you move from left to right, and sum the results.
- Divide the sum by 10.
- Subtract the remainder (not the quotient) from 10.
- If the result is 10, use the number 0.

Tulos näyttää tältä:
978-1-2-0-6-2-4-1-4-62
Tulos on surkea. Kokeilen muutamalla muullakin sanallisella algoritmilla, mutta tulos ei parane. Tässä vaiheessa täytyy tehdä päätös. Pääsin tiettyyn

pisteeseen helposti mutta sen pisteen jälkeen tie nousi pystyyn. Käytänkö vielä aikaa kehotteeseen vai kirjoitanko koodin vain itse?

Laita AI asialle, mene itse perässä?

ISBN-numeron tarkistusnumeron algoritmi on aika monimutkainen ja olisin ollut todella yllättynyt, jos sanallisen kuvauksen pohjalta olisi syntynyt toimivaa koodia. Toimivan ratkaisun olisi luultavasti saanut aikaan purkamalla ISBN-numeron generoinnin pienempiin paloihin ja lisäämällä kehotteeseen lisää vaiheita ja yksityiskohtia. Mutta jos joudun sanallisesti kuvaamaan jokaisen vaiheen erikseen tekoälylle, menetän oikeastaan kokonaan AI:n tuoman hyödyn. On vain todettava, että tässä kohtaa olen omillani ja minun täytyy kirjoittaa koodi itse.

Täysin validin ISBN-numeron generoiva funktio pelkästään AI-apurin tuottamana jäi siis haaveeksi. Esimerkki oli keinotekoinen, mutta se toivottavasti näytti yhden tavan hyödyntää tekoälyä arkisessa työssä. Esimerkissä AI tiesi jotain ISBN:stä, mutta tämän tietämyksen laajuuden sain selville vain kokeilemalla. Jos sen koulutusaineistossa olisi ollut vastaavia käyttötapauksia, se olisi luultavasti tuottanut täysin oikean funktion heti ensimmäisestä kehotteesta.

Koodausavustin on paljon tietävä, hieman typerähkö renki, jonka ymmärrys saattaa kertakaikkiaan loppua yllättävässä paikassa. Käytäntö on kuitenkin osoittanut tämän rengin mainioksi apuriksi. Se osaa rutiinit, tekee ne luotettavasti ja todella nopeasti. Mutta se osaa enemmänkin. Se osaa *minun rutiinini* eli tuntee kirjoittamani koodin ja sen lisäksi se osaa selittää minulle muiden kirjoittaman koodin.

Rutiineja inhoava asiantuntija on tekoälytyöelämän voittaja.

Kuten ensimmäisestä esimerkistä näimme, on jokaisen funktion yläpuolelle pieni ”explain” -teksti, joka vapauttaa yhden kielimallien supervoimista eli kyvyn selittää. Voin pyytää sanallisen kuvauksen itselleni vierasta koodista. Se nopeuttaa huimasti ohjelman rakenteen omaksumista eritoten, jos ollaan tekemisissä hieman itselle vieraan kielen tai toiminnallisuuden kanssa. Tämä on kullanarvoista sellaiselle, joka joutuu korjaamaan tai muuttamaan jonkun toisen – mahdollisesti kiireessä – tekemää sovellusta.

Toinen AI-avustimen kyky on sen kontekstitietoisuus. Avustin pystyy hyödyntämään muuta ohjelmistoprojektin koodia. Tämä tuottaa joskus uskomattomia hetkiä, jolloin avustin tuntuu lukevan ajatuksia. Koodia saattaa syntyä hetkittäin pelkkä tabulaattoria painelemalla ja koodi tekee juuri sen mitä pitikin. Joskus jopa ennen kuin olen tiennyt haluavani juuri kyseistä ratkaisua.

Olenko edelleen asiantuntija?

Ulkoistanko koodin kirjoittamiseen liittyvät rutiinit AI:lle? Kyllä. Ulkoistanko ne mielelläni? Kyllä! Luultavasti rutiini koodin *kirjoittamiseen* heikkenee, mutta kykyni koodin *ymmärtämiseen* täytyy pysyä hyvänä. Minun on edelleen ymmärrettävä se, mitä avustin tuottaa. Koodiavustin nopeuttaa minua vain, jos pystyn nopeasti tarkastamaan tuotetun koodin.

Rutiineja inhoava asiantuntija on tekoälytyöelämän voittaja. Kun tiedän mitä olen tekemässä ja saan jonkun tekemään tylsät asiat puolestani, säästän aikaa ja olen tehokkaampi. Innovointi, kokeilu ja keksiminen on kivaa, rutiinien tekeminen on tylsää. Annan siis mielelläni jonkun toisen hoitaa ne menettämättä kuitenkaan kontrollia niihin. Viihdyn työssäni paremmin ja pääsen tekemään niitä asioita, mitä AI-renki ei osaa.

Tulevaisuus

Tämän hetken avustimet eivät tiedä, *miksi* kirjoitan koodia ja miten kokonaisuuden tulisi toimia puhumattakaan siitä, millaisessa toimintaympäristössä on tarkoitus toimia. Seuraavaa looginen askel onkin semantiikasta kokonaisuuksiin. Ehkä pian ISBN-numeroita jakavan ohjelmiston koodaaminen voisi tapahtua askel askeleelta näin:

- Suunnittele sovellus, joka jakaa ISBN-numeroita ennalta annetusta listasta. Prosessi alkaa sillä, että henkilö pyytää ISBN-tunnuksen ja antaa lomakkeella tarvittavat tiedot. Julkaisuun tarvittavat tiedot löytyvät tästä linkistä: [LINKKI]
- Lisäksi tarvitaan henkilökunnan näkymä, jossa käsittelijä myöntää ISBN-tunnuksen.
- Sovelluksen pitää lopulta tallettaa julkaisu julkaisuarkistoon ja tutkimustietojärjestelmään. Rajapintojen tiedot löytyvät tuolta: [LINKKI]
- Sovelluksen pitää myös tuottaa teoksesta esitietue MARC-muodossa: [LINKKI]
- Ulkoasun voit soveltaa kirjaston graafisten ohjeiden pohjalta: [LINKKI]
- Kysy lisätietoja tarvittaessa.
- Good luck! Menen kahville.

Näin pitkällä ei olla vielä, mutta yksittäiset komponentit tällaiseen ovat jo olemassa ja jossain vaiheessa tämä tulee olemaan mahdollista. Tämä aiheuttaa sitten omat päänsärkynsä esimerkiksi ylläpidon, tietoturvan ja vastuiden saralla. Jos ohjelma toimii väärin, onko se kehotteen laatijan vai tekoälyn vika? Saako vian korjattua kehottamalla? Entä jos ei saa, niin kuka korjaa järjestelmän?

Yksi perustavanlaatuinen ongelma on luonnollisen kielen ja ohjelmointikielien suhde. Ohjelmointikielet ovat yksiselitteisiä, luonnollinen kieli ei. Ongelman sanoittaminen saattaa olla hyvin työlästä ja silti se voi olla monitulkintaista, varsinkin, kun puhutaan hyvin spesifeistä tarpeista.

Mutta toisaalta, tähän mennessä olemme käyneet dialogia koneen

kanssa lähinnä kirjoittamalla. Ehkä tulevaisuudessa ohjelmointi onkin ensisijaisesti puheammatti. Kehittäminen on jatkuvaa dialogia tekoälyn kanssa. Ehkä ohjelmistot lakkaavat olemasta erillisinä olioina ja niistä tulee ad hoc -tyyppisiä eli jotain, joka luodaan tarpeen mukaan? Vaikkapa tällaisia:

”Tee minulle tekstieditori, joka avustaa opiskelijoiden luentopäiväkirjojen palautteen annossa.”

”Luo herätyskello, joka hälyttää tämän viikon ajan kello 8.30 soittamalla Säkkijärven polkkaa eri versioina. Tuhoudu sitten.”

Haluaisitko helpomman työn?

Uhkakuvat kuuluvat vääjäämättömästi koneälyn ympärillä käytyyn keskusteluun. Tämänkertainen uhkakuva liittyy tekoälyn saaman kontekstin laajuuteen. Jotta AI voi ottaa kantaa kokonaisuuksiin, sillä täytyy olla myös kokonaisuuksien konteksti ja paljon tietoa toimintaympäristöstä. Mutta voiko AI:lla olla liikaa kontekstia? Kuuluvatko sähköpostini ja videotapaamisten litteroinnit AI-renkini taustatietohin? Vaaniiko tässä monokulttuurin uhka, joka tekee työntekijöistä AI:n osia, syötettä kielimallille tuottavia biologisia komponentteja?

Tässä keskustelun avauksia AI-isännältäni tulevaisuudesta, joita en toivoisi näkeväni:

- Kirjoitat koodia tänään huomattavan hitaasti. Oletko nukkunut hyvin?
- Muistat varmaan, että lupasit ohjelmiston valmiiksi 9.9. mennessä?
- Johdon kanssasi käymiesi sähköpostien ja videopuheluiden perusteella olet tehnyt viime aikoina muutamia vääriä arviointeja. Haluaisitko ehkä pyrkiä vähemmän haasteellisiin tehtäviin?
- Laadin puolestasi työpaikkailmoituksen: [LINKKI]

Tämän tekstin kirjoittamisessa ei ole käytetty tekoälyä (ei edes oikolukua). Kirjoittaja on STKS:n Uuden teknologian työryhmän jäsen.

Kirjoittaja

Ari Häyrinen
Jyväskylän yliopisto
Avoimen tiedon keskus
ari.hayrinen@jyu.fi