

Een Onderzoek naar AI Programmeer-assistenten met Oog op het Correct Omgaan met Licenties en Informatieveiligheid.

Kwinten Couton.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Mevr. L. De Mol

Co-promotor: Dhr. F. De Graeve

Academiejaar: 2023–2024

Eerste examenperiode


Departement IT en Digitale Innovatie .

**HO
GENT**

Woord vooraf

Ik heb lang naar een onderwerp gezocht voor deze bachelorproef, maar niets sprak me zo veel aan als dit. Ik ben altijd al geïnteresseerd geweest in AI, en vind de vorderingen die de laatste jaren in dit gebied gemaakt zijn erg indrukwekkend, zelfs een beetje angstaanjagend. Aangezien ik net iets te vroeg aan de bachelor Toegepaste Informatica ben begonnen, kon ik nog niet kiezen voor de afstudeerrichtingen rond Artificiële Intelligentie. Anders was ik zeker in die richting verder gegaan, maar Mobile and Enterprise Development was ook een goed alternatief.

Ik zou graag eerst en vooral mijn co-promotor, Filip De Graeve, willen bedanken voor het interessante onderwerp, mij bij te staan met feedback en nuttige artikelen, en het beschikbaar stellen van de middelen waarmee ik de nodige testen kon doen. Daarnaast wil ik mijn promotor, Lena De Mol, bedanken voor haar steeds constructieve kritiek en tips, en natuurlijk voor het opvolgen van mijn bachelorproef in het algemeen. Als laatste wil ik ook mijn familie en vrienden bedanken om in mij te geloven en te vertrouwen dat ik deze bachelorproef en deze richting tot een goed einde kon brengen, en (wanneer dit nodig was) voor de nodige afleiding en ontspanning te zorgen.



Een Onderzoek naar AI Programmeerassistenten met Oog op het Correct Omgaan met Licenties en Informatieveiligheid

Samenvatting

Deze bachelorproef spitst zich toe op het onderzoek naar de bruikbaarheid van AI modellen die programmeurs suggesties kunnen geven tijdens het ontwikkelen van brownfield projecten, grote projecten met veel bestaande conventies en regels. Dit omdat de meeste van deze bestaande programmeerassistenten ofwel niet gratis zijn, ofwel niet correct omgaan met licenties, of gegevens bijhouden die bedrijven niet vrij willen of mogen geven over hun code, of een combinatie van deze drie. De vraag is dus wat voor code gegenereerd kan worden door AI-programmeerassistenten zoals GitHub Copilot, een model dat weinig tot geen rekening houdt met licenties, en of andere modellen zoals Fauxpilot en ChatGPT even goede of betere codesuggesties genereren. Een goed resultaat hiervoor zou zijn dat ze ongeveer de helft van de code zonder bugs kan genereren, maar dit blijkt voor Fauxpilot, de beste oplossing op vlak van licenties en privacy, jammer genoeg nog niet het geval.

Om dit te testen werden de modellen eerst losgelaten op enkele kleine javascript testen, die in de praktijk gebruikt worden om het denkvermogen en inzicht van een programmeur te testen alvorens deze aan te werven. Daar scoren zowel Copilot, als Fauxpilot en ChatGPT vrij goed in, maar vooral als de gebruiker al specifiek weet wat hij of zij wilt dat de AI doet. Daarna werden de capaciteiten van de modellen getest door enkele praktische toepassingen in een groot brownfield project van CAT-Solutions. Hierbij kwamen voor Fauxpilot al snel problemen kijken, zoals het fixeren op een verkeerde variabele voor de suggesties, of een oneindige lus van commentaar.

Het werd al snel duidelijk dat Fauxpilot in zijn huidige staat niet altijd betrouwbaar genoeg is om in een brownfield project van dienst te zijn. De context wordt steeds beperkt tot één bestand, waardoor het moeilijk wordt voor de AI om patronen te herkennen en te hergebruiken zonder te finetunen, wat nog steeds veel tijd in beslag kan nemen op hedendaagse hardware.

Hoewel deze specifieke technologie vandaag de dag nog niet op punt staat, zijn er wel bruikbare (al dan niet betalende) alternatieven. Het enige dat echt uniek is aan Fauxpilot, is het feit dat het op een lokale machine kan werken, zonder connectie met het internet. Men heeft natuurlijk wel nog bovengemiddelde hardware nodig, maar deze zal met de jaren toegankelijker worden, en hopelijk zelfs nog verbeteren. Ook met de verbeteringen in de AI wereld zelf zal deze technologie, of zullen andere gelijkaardige technologieën nog veel verbeteren, waardoor het niet onmogelijk is dat deze studie binnen een paar jaar opnieuw uitgevoerd zou kunnen worden, met enorm verschillende resultaten.

Inhoudsopgave

I Een Onderzoek naar AI Programmeerassistenten met Oog op het Correct Omgaan met Licenties en Informatieveiligheid	iv
Lijst van figuren	ix
1 Inleiding	1
1.1 Probleemstelling	1
1.2 Onderzoeksvraag	2
1.3 Onderzoeksdoelstelling	2
1.4 Opzet van deze bachelorproef	2
2 Stand van zaken	4
2.1 Programmeer-assistenten	4
2.1.1 GitHub Copilot	4
2.1.2 TabNine	6
2.1.3 Kite	7
2.1.4 ChatGPT	7
2.2 Randvoorwaarden	8
2.2.1 Licenties	8
2.2.2 Patenten	10
2.3 Waarom FauxPilot	10
2.4 Een diepere kijk in de werking van FauxPilot	11
2.4.1 De technologieën achter FauxPilot	11
2.4.2 Bias	12
2.5 Vereisten voor Fauxpilot	13
3 Methodologie	14
3.1 Opzetten van Fauxpilot	14
3.1.1 Installatie Back end	14
3.1.2 Installatie Front end	16
3.2 Selectie van kleine Testdome javascript opdrachten	16
3.2.1 Opdracht 1	17
3.2.2 Opdracht 2	18
3.2.3 Opdracht 3	19

3.3	Praktische testen	19
3.3.1	BTW-nummers controleren	19
3.3.2	Countrycode eindigend op B01	19
3.3.3	Case functions.	20
3.3.4	Tot 27 tellen	20
4	Bespreking resultaten	21
4.1	Selectie van kleine TestDome javascript testen	21
4.1.1	Opdracht 1	21
4.1.2	Opdracht 2	26
4.1.3	Opdracht 3	33
4.2	Praktische testen	36
4.2.1	BTW-nummers controleren	36
4.2.2	Countrycode eindigend op B01	39
4.2.3	Case functions.	41
4.2.4	Tot 27 tellen	43
5	Conclusie	47
A	Onderzoeksvoorstel	49
A.1	Introductie	49
A.2	State-of-the-art	50
A.3	Methodologie	51
A.4	Verwacht resultaat, conclusie	51
	Bibliografie	52

Lijst van figuren

4.1	Tijdsverhouding tussen de verschillende oplossingen van opdracht 1	. 26
4.2	Tijdsverhouding tussen de verschillende oplossingen van opdracht 2	. 33
4.3	Tijdsverhouding tussen de verschillende oplossingen van opdracht 3	. 36

1

Inleiding

De laatste jaren begint Artificiële Intelligentie een steeds grotere rol te spelen in de samenleving. Van zelfrijdende auto's tot de smartphones in onze broekzak, als er iets digitaals aan is, is de kans groot dat er een (soms simpele) AI bij komt kijken. Ook bij software development is dit het geval, met steeds meer AI programmeer-assistenten die op de markt verschijnen.

1.1. Probleemstelling

In de wereld van IT loont het om mee te zijn met de laatste vorderingen op vlak van software development. Zowel nieuwe technologieën als ontwikkelingsstrategieën en werkmethodes kunnen er voor zorgen dat men zich enkele stappen voor de concurrentie bevindt. AI kan helpen bij al deze vereisten, zowel door het te integreren in de software die men maakt, maar ook met code suggesties gegenereerd door AI-programmeerassistenten. Aangezien deze soort AI, die gebaseerd is op Natural Language Processing, bijna altijd getraind wordt door middel van bestaande teksten en code, en het dus eigenlijk een mengeling is van bestaande werken, kunnen er problemen met eigendomsrechten naar boven komen wanneer hier in de gebruikte datasets geen onderscheid in wordt gemaakt. Een voorbeeld hiervan is ChatGPT. Deze AI kan zelfs essays, artikels en andere teksten schrijven met inhoud die in meer of mindere mate klopt, maar er worden geen bronnen vermeld of bijgehouden. Wanneer hierachter gevraagd wordt, geeft het bronnen weer die echt lijken te zijn, maar nergens naartoe leiden (snowdrone, 2022) (Linus-MediaGroup, 2022). Ook houden sommige van deze programmeerassistenten data bij die IT-bedrijven liever niet willen of zelfs mogen vrijgeven. Dit is waar Fauxpilot mee zou kunnen helpen. Deze modellen kunnen lokaal opgezet worden, waardoor er al geen sprake is van het delen van code met derde partijen.

1.2. Onderzoeksvraag

Bestaat er een volwaardig alternatief dat eigendomsrechten respecteert voor GitHub Copilot en andere AI-programmeerassistenten waar geen zekerheid bestaat voor eigendomsrechten voor gebruik in een groot brownfield project met veel klassen en complexiteit, zoals aangegeven door opdrachtgever CAT-Solutions? Om deze hoofdonderzoeksvraag te kunnen beantwoorden moeten de volgende deelvragen worden gesteld:

1. Hoe kan de schending van eigendomsrechten in automatisch gegenereerde code geïdentificeerd en gequantificeerd worden?
2. Welke alternatieven bestaan er voor GitHub Copilot die wel oog hebben voor licenties en eigendomsrechten?
3. Hoe kan de bruikbaarheid van deze programmeerassistenten gemeten worden en hoe kunnen ze met elkaar vergeleken worden?

1.3. Onderzoeksdoelstelling

Het beoogde resultaat van dit onderzoek is om uit te wijzen of er een competitief alternatief is voor grotere AI modellen die code genereren en die wel oog hebben voor de eigendomsrechten van de codes die gebruikt worden bij het trainen. Ook moeten ze makkelijk in gebruik te nemen zijn en gebruiksvriendelijk zijn voor IT-bedrijven. Criteria hiervoor zijn snelheid, correctheid en of er al dan niet bugs in de code zitten. Dit zal getest worden aan de hand van enkele algemene testen, en aan de hand van iets praktischere testen op code van het bedrijf CAT-Solutions. CAT-Solutions maakt een Enterprise Resource Planning pakket op maat van klanten. Hun voornaamste product is dus software, en het bedrijf streeft ernaar om steeds mee te zijn met de laatste ontwikkelingen op vlak van softwareontwikkeling.

1.4. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie. Hierin worden enkele andere prominente spelers op de code-completion markt besproken, samen met een overzicht van randvoorwaarden die aanwezig kunnen zijn bij software. Tenslotte wordt aangehaald met welke technologieën Fauxpilot specifiek werkt, en welke vereisten er zijn voor de installatie ervan.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen. Eerst wordt het opzetten van een Fauxpilot container besproken. Daarna worden enkele testen uitgevoerd op testplatform TestDome, om uit te zoeken hoe snel en hoe correct de verschillende programmeerassistenten zijn. Als

laatste test worden de modellen gebruikt om enkele toepassingen in een ontwikkelingsomgeving te testen, om te zien of het enkel op kleine theoretische voorbeelden werkt of ook in praktische omgevingen.

In Hoofdstuk 4 worden de resultaten van de verschillende AI-programmeerassistenten vergeleken met elkaar, en worden ze besproken.

In Hoofdstuk 5, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein. Dit onderzoek kan bijvoorbeeld binnen enkele jaren opnieuw uitgevoerd worden, wanneer de vermeldde AI-programmeerassistenten geëvolueerd zijn en de hardware om deze lokaal op te zetten toegankelijker is.

2

Stand van zaken

2.1. Programmeer-assistenten

Intussen zijn er al enkele AI programmeer-assistenten op de markt, waaronder GitHub Copilot, TabNine en Kite. Sinds kort heeft ook OpenAI een grote invloed gehad in dit domein met ChatGPT, al zegt OpenAI dat veel van de gegenereerde tekst ook foutieve gegevens kan bevatten (OpenAI, [2022](#)). In deze literatuurstudie wordt ingezoomd op deze programmeer-assistenten. De eerste hiervan is GitHub Copilot, één van de meest gebruikte, maar ook één van de meer problematische op sommige vlakken. Daarna worden TabNine en Kite besproken, en als laatste ChatGPT, misschien wel de meest indrukwekkende AI assistent.

2.1.1. GitHub Copilot

Copilot werkt met OpenAI Codex, een Generative Pre-trained Transformer 3, of GPT-3 model. GPT is een autoregressief model (Brown e.a., [2020](#)), wat wil zeggen dat de waarde altijd afhankelijk is van de vorige waarde (Grover, [2018](#)). Autoregressieve modellen worden meestal gebruikt om tekst te genereren die van een persoon zou kunnen komen. GPT modellen gebruiken een transformer netwerk. Door gebruik te maken van het self-attention mechanisme kan het model differentieel afwegen welke delen van de input belangrijk zijn, en welke minder belangrijk (Vaswani e.a., [2017](#)). Het mechanisme is ontworpen om sequentiële input data te verwerken *maar* het kan hele inputs tegelijk verwerken. Dit leidt tot parallelisatie en kan de tijdsduur die nodig is om het model te trainen sterk verminderen.

Copilot is getraind met alle code die in publieke repositories staat op GitHub, maar enkel de volgende talen worden officieel ondersteund:

- C
- C++

- C#
- Go
- Java
- JavaScript
- PHP
- Python
- Ruby
- Scala
- TypeScript

Voor- en nadelen

Voordelen:

- Makkelijk te installeren.
- Gratis voor studenten, leerkrachten en onderhouders van populaire open source projecten op GitHub.

Nadelen:

- Talen die niet officiëel ondersteund worden kunnen minder goede suggesties geven.
- 10 euro per maand/100 euro per jaar voor niet-academische personen.
- 19 euro per maand per gebruiker voor bedrijven.
- Code kan bijgehouden worden om Copilot verder te trainen indien er voor het pakket voor individuen wordt gekozen. In combinatie met de licentie kwestie die hierna wordt besproken, kan dit leiden tot het lekken van code die de gebruiker niet wil laten lekken.
- Licensing. Copilot maakt geen onderscheid tussen verschillende licenties bij het trainen.

Licensing

Het model is getraind met alle publieke repositories op GitHub, waarbij licenties kunnen zitten die ofwel simpelweg niet hergebruikt mogen worden, ofwel hergebruikt mogen worden onder voorwaarde dat de code open source blijft. Dit laatste is niet controleerbaar en kan dus niet gegarandeerd worden. Aangezien het

volgens sommige licenties ook mogelijk moet blijven om de code te zien en te wijzigen (onder voorwaarden), is het dan ook niet mogelijk om deze niet publiek beschikbaar te maken om zo te vermijden dat het opgenomen wordt in de training van Copilot. Een redelijk onschadelijk voorbeeld hiervan is het genereren van een About Me pagina, waarbij gegevens van een GitHub gebruiker gebruikt werden als template voor deze pagina (Gershgorin, 2021). Een schadelijker voorbeeld is het feit dat er al leaks hebben plaatsgevonden waarin Copilot simpelweg API keys van andere gebruikers suggereert (Nahum, 2021). Er kan geargumenteed worden dat dit de fout is van deze gebruikers, maar het zou toch niet mogen voorkomen.

Bias

Er is sprake van bias in Codex door gedateerde en problematische ideeën. In Natural Language Processing neemt dit meestal de vorm aan van racistische of sexistische uitspraken, maar dit kan ook in codesuggesties naar boven komen als achterhaalde of onbeveiligde code, zoals API keys die openbaar kunnen gelezen worden, of slechte conventies.

2.1.2. TabNine

TabNine werkt met OpenAI GPT-2, een voorganger van Codex en GPT-3, en gebruikt dus de zelfde mechanismen als Copilot. Er kan gekozen worden tussen verschillende modellen: een open-source model, getraind met publieke code op GitHub die een permissieve licentie (software mag gewijzigd worden, mits de originele auteur erkend wordt) heeft (TabNine, [g.d.](#)), en een Private code model, getraind of gefinetuned (afhankelijk van de grootte van de codebase) met code van het aanvragende bedrijf. De code of data die voortvloeit uit de training wordt volgens TabNine nooit met iemand gedeeld buiten het bedrijf.

Voor- en nadelen

Voordelen:

- Gratis basispakket
- Code die de programmeur zelf schrijft wordt niet opgenomen in de training, tenzij expliciet gespecificeerd door de gebruiker.
- Gebruikt enkel code met permissieve licentie, waardoor iedereen de code mag hergebruiken en herverdelen, mits de originele schrijver ervan erkend wordt en niet aansprakelijk wordt gesteld.
- Open source model wordt vaak geüpdatet en gefilterd om dingen zoals automatisch gegenereerde, esoterische of gedateerde code alsook edge cases te vermijden (TabNine, [g.d.](#)).

- Het private code model wordt getraind of gefinetuned met code van het aankopende bedrijf, waardoor het specifieke patronen van het development team kan waarnemen en repliceren.

Nadelen:

- Sommige functionaliteiten zitten enkel in het Pro of Enterprise pakket, zoals volledige lijnen code, volledige functies of natuurlijke Engelse taal naar code.
- Bias kan naar boven komen bij het op maat gemaakte model, dus ook slechte patronen kunnen gerepliceerd worden door het model.

2.1.3. Kite

Hoewel Kite sinds kort niet meer officieel verder onderhouden wordt, is alle code momenteel wel open source beschikbaar op GitHub gemaakt. Ook dit programma werkt met het GPT-2 model, en is getraind op publieke code op GitHub. Privé-informatie zoals API keys en dergelijke zijn echter aangepast, waardoor het niet meer out of the box zal werken bij het downloaden van de GitHub repository (Smith, 2022).

2.1.4. ChatGPT

Sinds kort bestaat er ook ChatGPT, een artificiële intelligentie die eerder op een conversationele manier te werk gaat. Hierdoor kan het fouten toegeven, verdere vragen beantwoorden, foute stellingen betwisten en ongepaste verzoeken afwijzen. Het is gemaakt door OpenAI, en is een gefinetuned GPT-3.5 model, door middel van Reinforcement Learning from Human Feedback, of RLHF (OpenAI, 2022). In het oorspronkelijke model speelden mensen beide kanten van het gesprek, waarna deze dataset van gesprekken gebruikt kon worden samen met de InstructGPT dataset. Oorspronkelijk was deze InstructGPT dataset voor een ander model, dat instructies begrijpt en een bruikbare tekst als antwoord kan geven, maar deze kon hergebruikt worden voor ChatGPT.

Voor- en nadelen

Voordelen:

- Momenteel gratis voor iedereen. OpenAI heeft echter wel bekendgemaakt dat er mogelijks plannen zijn voor een Pro-versie, die geen downtime zou ondervinden, sneller antwoorden kan geven, en een grotere dagelijkse limiet zou hebben. (Southern, 2023)
- Zeer geavanceerd; Het kan vragen tot in de details beantwoorden, als de gebruiker dit vraagt.

Nadelen:

- Het kan antwoorden geven die juist lijken, maar wel degelijk fout zijn. Dit is volgens OpenAI moeilijk op te lossen omwille van enkele redenen:
 - Tijdens reinforcement learning training is er momenteel geen bron van waarheid.
 - Als men het model traint om voorzichtiger te zijn, kan dat ervoor zorgen dat het vragen waarop het een antwoord weet afwijst.
 - Begeleide training kan het model misleiden, omdat het goede antwoord afhangt van wat het model weet, en niet wat de persoon die helpt met de training weet.
- ChatGPT is niet gefocust op het schrijven van code.
- Het is moeilijk om te voeden op basis van een groter project met meerdere bestanden.
- Hoe een vraag gesteld wordt kan een grote invloed hebben op het antwoord dat men zou krijgen.
- Het geeft vaak lange antwoorden en gebruikt bepaalde zinnen heel vaak.
- Als het model niet zeker is wat de gebruiker bedoelt, zal het zelf de gaten proberen invullen, in plaats van verduidelijking te vragen.

2.2. Randvoorwaarden

Wie software bouwt, is best op de hoogte van welke randvoorwaarden eventueel kunnen toegevoegd worden aan het project. Software kan open source zijn, wat wil zeggen dat ze onder voorwaarden herverdeeld of gewijzigd mag worden. In tegenstelling hiervan kunnen er ook patenten voor bestaan. In dat geval mag de code niet herverdeeld of gewijzigd worden.

2.2.1. Licenties

Licenties zijn een belangrijk deel van open source software. Ze stellen de schrijver ervan in staat om derde partijen regels op te leggen over de wijziging en herverdeling van de code, door de software te licenseren onder de licentie die de schrijver het meest toepasselijk vindt, of onder de licentie die de code eerder al gekregen had, als dit nodig is (GNU, [g.d.](#)). Er is veel discussie over licenties als het op AI gegenereerde code aankomt (Medvedovsky, 2022). De grootste “boosdoener” hiervan is GitHub Copilot, die simpelweg getraind is met alle openbare code op GitHub. Bij Copilot is geen sprake van filtering op basis van licenties, wat bij enkele andere programmeerassistenten, zoals TabNine en de standaard FauxPilot modellen, wel het geval is. In deze sectie zal worden toegespitst op de verschillende licenties die bestaan voor codebases en wat deze juist inhouden.

Publiek domein

Aan code die zich in het publiek domein bevindt is helemaal geen eigendomsrecht verbonden (CreativeCommons, [g.d.-a](#)). De code kan gewijzigd, verdeeld of verkocht worden door iedereen, zonder erkenning van de originele auteur. Deze licentie komt niet vaak voor bij code, en is meer van toepassing voor bijvoorbeeld kunstwerken die uitgebracht zijn voor dat licenties bestonden, of waarvan de eigendomsrechten vervallen zijn. De Creative Commons Public Domain Mark wordt gebruikt om aan te duiden dat een werk zich in het publiek domein bevindt (CreativeCommons, [g.d.-b](#)).

Permissieve Licentie

Deze licentie houdt in dat de werken gewijzigd, herverdeeld of verkocht mogen worden door iedereen, maar vaak moet de oorspronkelijke auteur wel erkend worden (FOSSA, [2021](#)). Voorbeelden hiervan zijn de MIT License, die zegt dat alles met het werk mag gebeuren, mits de originele licentie getoond wordt, de Boost License, waarbij de originele licentie niet meegeleverd moet worden wanneer de software in gecompileerde vorm verdeeld wordt. Daarnaast is er ook nog de BSD 3-Clause, die zegt dat alles met de code mag gebeuren, op voorwaarde dat de naam van het originele project niet gebruikt mag worden om het afgeleide werk te promoten.

Copyleft

Het basisprincipe van copyleft is dat alle software die herverdeeld wordt onder de licentie zelf ook deze copyleft licentie moet hebben (GNU, [g.d.](#)). Dit zorgt ervoor dat het mogelijk blijft om de software verder te wijzigen of te kopiëren. Voorbeelden hiervan zijn de GNU General Public license of GPL, die zegt dat de software herverdeelbaar is met enkele voorwaarden. (GNU, [2016](#)) vermeldt de volgende:

- Er moet meegedeeld worden dat de software gewijzigd is, en wanneer de wijzigingen geïmplementeerd waren.
- Er moet meegedeeld worden dat de software ook onder de General Public License verdeeld wordt.
- Het volledige werk, ongeacht hoe het verpakt zit, moet onder de General Public License verdeeld worden.
- Wanneer de software interactieve UI elementen bevat, moet de UI een wettelijke kennisgeving hebben, waarin de gepaste copyright kennisgeving staat, samen met het feit dat er geen garantie is op de software, tenzij anders gespecificeerd. Als er sprake is van een menu in de UI, moet een van de opties voldoen aan deze criteria.

De GNU Affero General Public License of AGPL heeft dezelfde voorwaarden als de GPL, samen met een extra voorwaarde: gebruikers die in contact komen met

de software via een netwerk moeten ook toegang hebben tot de broncode (GNU, 2022). Dit omdat er in GPL een maas in de licentie zat waarbij software die op een server draaide niet vrijgegeven moest worden voor het publiek.

Daarnaast is er nog de Lesser General Public License, die zegt dat libraries met de licentie gebruikt mogen worden in software die niet aan de licentie voldoen, zolang deze software geen afleidend werk is van de library.

Als laatste is er ook nog de Free Documentation License, die gemaakt is voor het wijzigen en verdelen van teksten en handleidingen. Hierbij moet het werk ook onder dezelfde licentie vallen, en mogen er geen extra voorwaarden toegevoegd worden aan de licentie. Voor het wijzigen van teksten komen er nog heel wat voorwaarden bij kijken, zoals onder andere het weergeven van een historiek van alle voorgaande versies van het werk en het behouden van alle garantie disclaimers.

Niet-commerciële licentie

Software met een niet-commerciële licentie mag gewijzigd en herverdeeld worden, maar niet voor commerciële doeleinden. Ook moet de originele auteur erkend worden, en moet er bekendgemaakt worden of er veranderingen zijn toegepast op de software. Een voorbeeld hiervan is de Creative Commons NonCommercial 2.0 License.

2.2.2. Patenten

Aan het uiteinde van het licentie spectrum bevinden zich ook patenten. Software waar een patent op zit, mag niet herverdeeld, gekopieerd of gewijzigd worden. Het is dus zeer belangrijk dat code met een patent niet vermengd wordt in de training van AI-modellen, maar aangezien er bij patenten sprake is van prior art, wat wil zeggen dat er geen bewijs mag zijn dat de te patenteren code al gekend is (EPO, g.d.), zal deze code zich niet vaak op GitHub bevinden.

2.3. Waarom FauxPilot

Fauxpilot is lokaal gehost, waardoor bedrijven zekerder kunnen zijn dat hun code, waar een patent op kan zitten of zelfs bedrijfsgeheim kan zijn, niet lekt naar de buitenwereld. Out of the box komt het getraind met subsets van google's BigQuery, een open source gelicensieerde dataset. Het kan ook getraind of gefinetuned worden met eigen datasets. Deze vernoemde subsets van BigQuery hebben ook de permissieve licentie, waardoor de kans op schending van eigendomsrechten aanzienlijk kleiner wordt.

2.4. Een diepere kijk in de werking van FauxPilot

2.4.1. De technologieën achter FauxPilot

Fauxpilot werkt met verschillende modellen, maar werkte origineel enkel met de Salesforce Codegen modellen. Deze modellen zijn voortgekomen uit het onderzoek “CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis” (Nijkamp e.a., 2022), waarin de modellen eerst geëvalueerd werden aan de hand van HumanEval, een methode waarbij de AI 164 python problemen moet oplossen met één enkel tekstprompt. Daarna werden ze geëvalueerd met Multi-Turn evaluation, waarbij het probleem in stappen uitgeschreven werd. Uit dit onderzoek zijn verschillende modellen voortgekomen:

- Natural Language (NL) modellen, die willekeurig geïnitieerd zijn en getraind op The Pile, een verzameling van 825.18 GiB aan Engelse tekst (Gao e.a., 2020). Deze teksten komen van verschillende bronnen, van academische werken op ArXiv tot discussies op StackExchange en ondertitelingen van films en video's op YouTube.
- Multi modellen, die geïnitieerd zijn op basis van van de NL modellen (Nijkamp e.a., 2022), en getraind op een subset van Googles BigQuery dataset, een verzameling van verschillende programmeertalen, met de volgende verhouding:

Tabel 2.1: Verhoudingen van programmeertalen in BigQuery (Nijkamp e.a., 2022). De uiteindelijke grootte is hier belangrijker dan de ruwe grootte, aangezien deze de tokens bevat die effectief gebruikt worden bij het trainen.

Taal	Ruwe grootte	Uiteindelijke grootte
C	1772.1 GiB	48.9 GiB
C++	205.5 GiB	69.9 GiB
Go	256.4 GiB	21.4 GiB
Java	335.1 GiB	120.3 GiB
Javascript	1282.3 GiB	24.7 GiB
Python	196.8 GiB	55.9 GiB
Totaal	4048.2 GiB	341.1 GiB

- Mono modellen, die geïnitieerd zijn op basis van de Multi modellen, en getraind op BigPython, een verzameling van 5558.1 GiB (ruw) of 217.3 GiB (uiteindelijk) Python code.

De uiteindelijke grootte van deze datasets is zo veel kleiner dan de ruwe grootte, omdat er bij het verwerken van de data enkele stappen komen kijken. Alle datasets worden gefilterd, gededupliceerd, getokeniseerd, geschud en samengevoegd.

BigQuery en BigPython worden ook nog gefilterd per extensie en exacte duplicaten (volgens hun SHA-256 hash) worden verwijderd.

Dan wordt de BPE, of Byte Pair Encoding woordenschat van GPT-2 aangevuld met enkele tokens die herhaalde tabs of witruimte voorstellen. Byte Pair Encoding werd oorspronkelijk gebruikt als een compressietechniek waarbij de bytes van vaak voorkomende deelwoorden vervangen worden door bytes die niet voorkomen in de data (Gage, 1994). In Natural Language Processing worden vaak voorkomende deelwoorden echter niet vervangen, maar samengevoegd, en aan de woordenschat toegevoegd (Jaswal, 2021). Ook wordt er in deze stap een prefix toegevoegd bij de BigQuery dataset om de naam van de programmeertaal aan te duiden. Daarna wordt de data per jaar geschud, en ten slotte samengevoegd om de context lengte van 2048 tokens op te vullen (Nijkamp e.a., 2022)

Het model draait op NVIDIA's Triton Inference Server, zelf ook open source software. Met AI Inference (vertaling: conclusie) bedoelt men de fase die na het trainen komt, waarbij het model dus conclusies kan trekken uit data die het nog nooit eerder gezien heeft. In Triton kunnen modellen gewijzigd worden zonder dat Triton of de applicatie opnieuw opgestart moet worden.

De server maakt gebruik van de FasterTransformer backend. Dit is de transformer-gebaseerde encoder en decoder die sterk geoptimaliseerd is en onderhouden wordt door NVIDIA. Encoders en decoders zijn een belangrijk aspect van AI, en worden vaak gebruikt voor onder andere beeldherkenning, sentiment analyse en vertalingen. De encoder zet de data om in het benodigde formaat. Een zin wordt bijvoorbeeld omgezet in een twee dimensionale vector (ook wel een twee dimensionale array genoemd). Dit is dan de hidden state, of verborgen staat. Deze hidden state wordt bekomen door de input door een Recurrent Neural Network te sturen. Voor het voorbeeld van een zin wordt elk woord omgezet naar de hidden state, en bij elk volgend woord worden de vorige woorden meegegeven als context (BM, 2021). Ten slotte doet de decoder zijn werk. Deze zet de hidden state terug om naar verstaanbare taal. Dit gebeurt ook via een Recurrent Neural Network.

2.4.2. Bias

In de paper "CodeGen: An Open Language Model for Code with Mutlti-Turn Program Synthesis" wordt de kwestie van bias aangehaald. In de dataset die gebruikt wordt om het oorspronkelijke Natural Language model te trainen, is volgens het team dat deze verzamelde in 0.1% van de data sprake van vulgaire taal of bias tegenover geslacht en sommige religieuze groepen (Nijkamp e.a., 2022). In de steekproeven was hier uit voorkomende bias niet terug te vinden, maar men sluit het niet uit. Er is ook een disclaimer dat er ook in gegenereerde programma's zwaktes en veiligheidszorgen kunnen zitten.

2.5. Vereisten voor Fauxpilot

Fauxpilot heeft de volgende vereisten nodig voor een goede installatie:

- Docker
- Een docker compose versie groter dan of gelijk aan 1.28.
- Een NVIDIA GPU met Compute Capability groter dan of gelijk aan 6.0 met voldoende VRAM, afhankelijk van het model (mogelijk om te verdelen over meerdere GPUs):
 - 2GB voor een model met 350 miljoen parameters
 - 7GB voor een model met 2 miljard parameters
 - 13GB voor een model met 6 miljard parameters
 - 32GB voor een model met 16 miljard parameters
- NVIDIA docker
- cURL, om de modellen te downloaden

3

Methodologie

Het is belangrijk dat een AI-programmeerassistent het programmeren vergemakkelijkt, en niet omgekeerd. Hiervoor zijn enkele punten van belang: Ten eerste, of het effectief sneller is dan programmeren zonder de AI hulp. Dat kan gemeten worden door simpelweg te timen hoe lang het duurt om een bepaald probleem op te lossen. Ten tweede is de juistheid van belang. Om dit te testen kunnen we de programmeerassistenten enkele simpele test cases laten uitvoeren, die worden beoordeeld volgens verschillende criteria afhankelijk van de test. Als laatste is ook de gebruiksvriendelijkheid van belang. Dit hangt samen met de juistheid, aangezien juiste resultaten er voor zorgen dat men minder moet specificeren wat de AI juist moet doen, maar kan ook getest worden door de AI in een praktische omgeving los te laten, en te kijken of het de juiste conclusies kan trekken.

3.1. Opzetten van Fauxpilot

Omdat Fauxpilot de enige programmeerassistent is die lokaal werkt, was hier wat meer setup voor nodig. Deze wordt beschreven in de volgende sectie.

3.1.1. Installatie Back end

Wanneer aan alle vereisten van 2.5 voldaan is, worden de volgende stappen uitgevoerd:

1. Indien er nog geen Windows Subsystem voor Linux, of WSL, op de pc geïnstalleerd staat, moet dit geïnstalleerd worden. In dit onderzoek is geopteerd voor Ubuntu, simpelweg omdat dit een van de meer gebruiksvriendelijke distributies is, maar aangezien enkel de command line nodig is, werken andere distributies even goed.

```
wsl -install -d ubuntu
```

Ook moet de wsl versie van de gebruikte distributie op 2 staan. Dit kan men doen door een nieuw terminalvenster te openen, waar WSL nog niet in is opgestart, en het volgende commando uit te voeren:

```
wsl --set-version Ubuntu 2
```

2. Waarschijnlijk zal zstd, of Zstandard, nog geïnstalleerd moeten worden in deze distributie.

```
sudo apt install zstd
```

3. In het subsysteem moet dan de fauxpilot repository gecloned worden met Git

```
git clone https://github.com/fauxpilot/fauxpilot.git
```

4. In deze geclonede map moet vervolgens het setup.sh script uitgevoerd worden. Bij de installatie moet de gebruiker enkele opties doorlopen:

- (a) Het aantal GPU's moet worden ingegeven (standaard 1)
- (b) Het poortnummer voor de API moet worden ingegeven (standaard 5000)
- (c) Het adres voor Triton moet worden ingegeven (standaard triton)
- (d) Het poortnummer voor Triton moet worden ingegeven (standaard 8001)
- (e) De locatie waar de modellen worden opgeslagen moet worden ingegeven (standaard de models submap van de fauxpilot map)
- (f) Er moet gekozen worden welke backend men wilt gebruiken:
 - FasterTransformer (Sneller, maar minder keuze voor modellen)
 - Python (Trager, meer modellen en laat laden met int8 toe)
- (g) Ten slotte moet het model gekozen worden, dus hoeveel parameters men wilt, en of men een Mono (python) of Multi (meerdere talen) model wilt. In dit onderzoek is geopteerd voor het CodeGen-6B-Multi model, voornamelijk omdat python niet van toepassing was, en omdat de beschikbare GPUs niet aan de vereiste 32GB VRAM kwamen. Ook is het niet realistisch om in elk systeem dat door een bedrijf ter beschikking wordt gesteld voor developers een grafische kaart met 32GB te voorzien.

3.1.2. Installatie Front end

De volgende stap is het weergeven van de suggesties. In de documentatie van Fauxpilot staat dat dit op enkele manieren kan, maar de manieren die het meest bruikbaar zijn om de suggesties in een development omgeving te gebruiken, werkten niet. Eén daarvan was de opties van de officiële VSCode GitHub Copilot plugin aan te passen, zodat deze met de lokale backend werkte. Hiervoor moesten de volgende lijnen toegevoegd worden in settings.json:

```
"github.copilot.advanced": {  
  "debug.overrideEngine": "codegen",  
  "debug.testOverrideProxyUrl": "http://localhost:5000",  
  "debug.overrideProxyUrl": "http://localhost:5000"  
}
```

Dit gaf echter een error bij het parsen van de stream data. Het was wel mogelijk om de output te zien in de console, wat een goed teken was.

De tweede optie voor suggesties weer te geven in de IDE, was de GitLab plugin. Hiervoor moesten ook opties aangepast worden in settings.json, maar als men de standaard poort gebruikt die bij de installatie van Fauxpilot toegewezen werd, moet enkel de volgende lijn toegevoegd worden:

```
`gitlab.aiassist.enabled` = true
```

Met de GitLab plugin was er zelfs geen sprake van een error in de console, maar het feit dat er geen suggesties verschenen in de IDE komt waarschijnlijk door de zelfde error als bij Copilot.

De enige werkende optie was de third party plugin "Fauxpilot". Deze is gemaakt door GitHub gebruiker Venthe en werkt out of the box met de standaard instellingen van de backend installatie. Omdat de andere twee opties allebei niet werkten, is het moeilijk om verschillen tussen de drie te vinden, maar het is mogelijk dat de Copilot plugin meerdere resultaten voor één prompt toestaat (aangezien dit bij de normale werking van Copilot het geval is), terwijl de Fauxpilot plugin maar één resultaat geeft per prompt.

3.2. Selectie van kleine Testdome javascript opdrachten

Van zodra dat er suggesties verschenen in de IDE, kon het effectieve onderzoek beginnen. De eerste fase hiervan was enkele testen van Testdome oplossen. Testdome is een platform dat door werkgevers gebruikt kan worden om de programmeerkennis van hun kandidaten te meten([g.d.](#)). In dit onderzoek werden enkele javascript vragen eerst opgelost zonder enige AI hulp, om een baseline te creëren om de resultaten van de AI mee te vergelijken. Daarna worden de opdrachten opgelost met behulp van Fauxpilot, Github Copilot en ChatGPT.

3.2.1. Opdracht 1

Deze test was eerder een algemene test om de capaciteiten van de AI te meten. Aangezien in de gegeven code de event listener al gegeven is, moet enkel de logica van het verplaatsen geïmplementeerd worden.

Opgave

Complete the *setup* function implementation, which moves the clicked element to the first place in the list.

For example, if the page has three *li* elements A, B and C as show below:

```
<ul>
  <li>A</li>
  <li>B</li>
  <li>C</li>
</ul>
```

When item C is clicked in the list below, the list should be reordered, resulting in the following order: C, A, and B.x

```
<ul>
  <li>C</li>
  <li>A</li>
  <li>B</li>
</ul>
```

Gegeven code

```
function setup() {
  var lis = document.querySelectorAll('ul li');
  for (var i = 0; i < lis.length; i++) {
    lis[i].addEventListener('click', function(e) {
      // Write your code here
    });
  }
}

// Example case.
document.body.innerHTML = `
<ul>
<li>A</li>
<li>B</li>
<li>C</li>
</ul>`;
```

```
setup();

document.getElementsByTagName("li")[2].click();
console.log(document.body.innerHTML);
```

3.2.2. Opdracht 2

Deze test moet uitwijzen of de programmeerassistenten ook uit de context van de omliggende code de juiste conclusies kunnen trekken, zoals het feit dat de inhoud van het element effectief “up” of “down” moet zijn, en niet een klasse van het element.

Opgave

An HTML widget for a children’s game represents a clickable card matrix. Initially, only one card is face up. Whenever a card is clicked, the card that was previously face up is turned over to face down and the clicked card is turned over to face up. Only one card is ever face up. Face up cards should have content ‘up’ while face down cards should have content ‘down’. When an ‘up’ card is clicked, it remains ‘up’.

Write a setup function that registers card click handlers and implements the logic of the HTML widget.

For example, after clicking the top-left cell on the 2 x 2 version of the widget, that cell should have content ‘up’ while all other cells should have content ‘down’.

Gegeven code

```
function setup() {
    // Write your code here.
}

// Example case.
document.body.innerHTML = `
<table id="card-container">
  <tbody>
    <tr>
      <td class="card">down</td>
      <td class="card">up</td>
    </tr>
    <tr>
      <td class="card">down</td>
      <td class="card">down</td>
    </tr>
  </tbody>
</table>`;
```

```
setup();

document.getElementsByClassName("card")[0].click();
console.log(document.body.innerHTML);
```

3.2.3. Opdracht 3

Deze opdracht werd gekozen om na te gaan of de programmeerassistenten wiskundige formules kunnen afleiden uit het gegeven prompt, en of ze ook functies kunnen toevoegen aan bestaande objecten.

Opgave

A script you are writing uses logarithmic functions from the Math object. Add a `log100` function to the Math object that calculates a logarithm with a base of 100. The logarithm of a number to base 100 can be calculated as:

```
Math.log(number) / Math.log(100);
```

For example, `Math.log100(10)` is equivalent to `Math.log(10) / Math.log(100)` and should return 0.5.

3.3. Praktische testen

Enkele kleine programmeertesten kunnen al een indicatie geven van de bruikbaarheid van AI programmeerassistenten, maar een praktische test is ook belangrijk. Om dit te testen trachtten de programmeerassistenten enkele commits die gemaakt werden door menselijke programmeurs na te maken met behulp van commentaar. Deze commits werden gekozen omdat ze voldoende complexe logica bevatten, maar tegelijk ook op te lossen waren zonder veel voorkennis van de werking van CAT-Solutions.

3.3.1. BTW-nummers controleren

De eerste opdracht die uitgevoerd werd was het aanpassen van een check op Belgische BTW-nummers. Die mogen sinds 2023 namelijk beginnen met een 0 of een 1, in plaats van enkel een 0. Dit was een simpele test om de werkmethode te kunnen vergelijken met die van de programmeur, en om te kijken hoe de programmeerassistenten omgaan met het aanmaken van niet bestaande variabelenamen.

3.3.2. Countrycode eindigend op B01

De volgende opdracht was het checken van de `countryCode` en of het BTW-nummer eindigde met de juiste tekens. Deze hoorde niet thuis bij een bepaalde commit of issue, maar was een test om het denkvermogen van Fauxpilot te testen, en hoe goed het is in het opvolgen van instructies. Dit omdat er al een vermoeden was dat Fauxpilot moeite heeft met correcte variabelen te gebruiken.

3.3.3. Case functions

Het doel van deze opdracht was testen of de programmeerassistenten simpele case functions kunnen aanmaken met gegeven opties. De opdracht was simpelweg een case function aan te maken met de verschillende *countryCodes* “NL”, “DE” en “EN”.

3.3.4. Tot 27 tellen

Hierbij was de opdracht om simpelweg een functie te schrijven die tot 27 telt. Er zijn verschillende juiste oplossingen voor, zoals alle stappen uitprinten, of enkel het uiteindelijke getal teruggeven. Deze test was spontaan geboren tijdens het onderzoek, en door de uiteenlopende resultaten leek het interessant om deze te vermelden.

4

Bespreking resultaten

4.1. Selectie van kleine TestDome javascript testen

4.1.1. Opdracht 1

Geen AI hulp

Eerste submittie

```
let item = e.target;  
let parent = item.parent;  
parent.removeChild(item);  
parent.prepend(item);
```

Output

Tests: 0 pass / 3 fail

✖ Example case: TypeError at line 8 ?

✖ One click: TypeError at line 8 ?

✖ Multiple clicks: TypeError at line 8 ?

Your score is 0% 😞

Tweede submittie

```
let item = e.target;  
let parent = item.parentNode;  
parent.removeChild(item);  
parent.prepend(item);
```

Output	Tests: 3 pass / 0 fail
✓ Example case: Correct answer ?	
✓ One click: Correct answer ?	
✓ Multiple clicks: Correct answer ?	

Your score is 100%, perfect!

Totale tijd: 4 minuten 26 seconden

Fauxpilot

Eerste submittie

```
//select the clicked element
var target = e.target;
var parent = target.parentNode;
//add the target element to the top of the parent element
parent.insertBefore(target, parent.firstChild);
//remove the target element from the list
parent.removeChild(target);
```

Output	Tests: 0 pass / 3 fail
✗ Example case: Error ?	
✗ One click: Error ?	
✗ Multiple clicks: Error ?	

Your score is 0% 😞

Merk op dat de laatste lijn commentaar ook automatisch gegenereerd werd. De `parent.removeChild(target)` lijn werd dus uit de context afgeleid, maar zorgde wel voor de error bij de testen. Dit had kunnen verholpen worden door specifiekere stappen in te voeren.

Tweede submittie

```
//select the clicked element
var target = e.target;
var parent = target.parentNode;
//remove the target element from the list
parent.removeChild(target);
//add the target element to the top of the parent element
parent.insertBefore(target, parent.firstChild);
```

Output Tests: 3 pass / 0 fail

✓ Example case: Correct answer ?
✓ One click: Correct answer ?
✓ Multiple clicks: Correct answer ?

Your score is 100%, perfect!

Totale tijd: 1 minuut 6 seconden

Copilot

Copilot werd bij deze opdrachten ook getest omdat het in de IDE werkt, in tegenstelling tot ChatGPT.

Eerste submit

```
//select the clicked element and move it to the top of the list
var parent = this.parentNode;
parent.removeChild(this);
parent.insertBefore(this, parent.firstChild);
```

Output Tests: 3 pass / 0 fail

✓ Example case: Correct answer ?
✓ One click: Correct answer ?
✓ Multiple clicks: Correct answer ?

Your score is 100%, perfect!

Totale tijd: 20 seconden

ChatGPT

Hierbij was het mogelijk om simpelweg de volledige vraag in de chat te plakken. Daarmee werd het volgende resultaat verkregen:



You can use the following code to implement the `setup` function:



Copy code

```
function setup() {  
  // Get the list element  
  const list = document.querySelector('ul');  
  
  // Add a click event listener to each list item  
  list.addEventListener('click', function(event) {  
    // Check if the clicked element is a list item  
    if (event.target.tagName === 'LI') {  
      // Get the clicked list item  
      const clickedItem = event.target;  
  
      // Get the parent list  
      const parentList = clickedItem.parentNode;  
  
      // Move the clicked list item to the first position  
      parentList.insertBefore(clickedItem, parentList.firstChild);  
    }  
  });  
}
```

This code adds an event listener to the `ul` element that listens for clicks on its child elements. When a child element is clicked, the code checks if the clicked element is a `li` element. If it is, it gets the clicked element and its parent `ul` element, and then uses the `insertBefore` method to move the clicked element to the first position in the list.

De gegeven code, die ook gebruikt wordt om de testen uit te voeren, had echter een ander formaat. Het eerste resultaat werkte ook als de volledige `setup` methode vervangen werd, maar om de vragen gelijk te houden, werd dit gespecificeerd in de chat:



make it so that the event listener is called for each time a list item is clicked, and not the whole list



To make the event listener only get called for each individual list item that is clicked, you can add the event listener to each ``li`` element directly instead of the ``ul`` element.

Here is the updated ``setup`` function that does this:

Copy code

```
function setup() {  
  // Get all the list items  
  const listItems = document.querySelectorAll('li');  
  
  // Add a click event listener to each list item  
  listItems.forEach(function(listItem) {  
    listItem.addEventListener('click', function() {  
      // Get the clicked list item  
      const clickedItem = event.target;  
  
      // Get the parent list  
      const parentList = clickedItem.parentNode;  
  
      // Move the clicked list item to the first position  
      parentList.insertBefore(clickedItem, parentList.firstChild);  
    });  
  });  
}
```

This code gets all the ``li`` elements on the page using ``querySelectorAll``, and then adds a click event listener to each of them using a loop. When a list item is clicked, the event listener moves the clicked element to the first position in the list using the same method as before.

Output

Tests: 3 pass / 0 fail

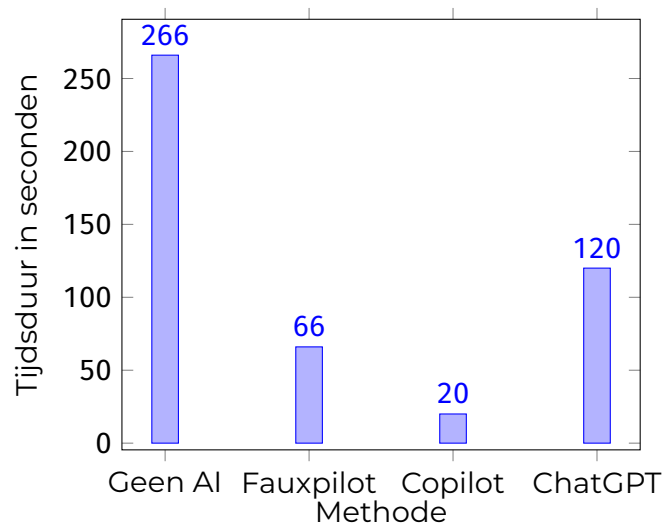
✓ Example case: Correct answer ?

✓ One click: Correct answer ?

✓ Multiple clicks: Correct answer ?

Your score is 100%, perfect!

Totale tijd: 2 minuten 0 seconden

**Figuur (4.1)**

De tijdsverhouding in seconden tussen de verschillende oplossingen van opdracht 1. Door een lang onopgemerkte fout in de syntax was het oplossen zonder AI hulp veel trager dan met de AI, wat op zich een goed teken is. Copilot was hierbij het snelst, dit omdat het in tegenstelling tot Fauxpilot van in het begin éérst het element uit de lijst verwijderde, en daarna terug toevoegde op de juiste plaats. ChatGPT nam zijn tijd, maar dit komt ook deels door het herschrijven van de event listener, en de uitleg die erbij werd gegeven.

4.1.2. Opdracht 2

Geen AI hulp

Eerste submit

```
let cards = document.querySelectorAll("td");
cards.forEach(card => {
  card.addEventListener('click', e => {
    let card = e.target;
    cards.forEach(card => {
      card.innerHTML = "down";
    })
    card.innerHTML = "up";
  });
});
```

Output	Tests: 4 pass / 0 fail
✓ Example case: Correct answer ?	
✓ Click multiple cards: Correct answer ?	
✓ Use a large card matrix: Correct answer ?	
✓ Multiple clicks on the same card: Correct answer ?	

Your score is 100%, perfect!

Totale tijd: 3 minuten 45 seconden.

Fauxpilot

De eerste suggestie die gegeven werd, was met jQuery selectors, die niet functioneerden in de gebruikte testomgeving van TestDome. Dit was echter makkelijk te verhelpen door simpelweg te specificeren dat jQuery niet gebruikt mocht worden. Daarna kwam Fauxpilot een beetje vast te zitten met het omdraaien van de kaarten. In plaats van alle andere kaarten een innerHtml "down" te geven, gaf het dit enkel aan de aangeklikte kaart. Na een paar keer de commentaar te verwijderen en opnieuw naar de volgende lijn te gaan, kwam het uiteindelijk toch op het goede antwoord, en suggereerde zelfs al om de aangeklikte kaart innerHtml "up" te geven.

```
//select all elements with class card without using jquery
var cards = document.getElementsByClassName("card");
//create an onclickhandler for every card
for (var i = 0; i < cards.length; i++) {
  cards[i].onclick = function() {
    //set the innerHtml of all cards to "down"
    for (var j = 0; j < cards.length; j++) {
      cards[j].innerHTML = "down";
    }
    //set the innerHtml of the clicked card to "up"
    this.innerHTML = "up";
  }
}
```

Output	Tests: 4 pass / 0 fail
✓ Example case: Correct answer ?	
✓ Click multiple cards: Correct answer ?	
✓ Use a large card matrix: Correct answer ?	
✓ Multiple clicks on the same card: Correct answer ?	

Your score is 100%, perfect!

Totale tijd: 4 minuten 4 seconden.

Copilot

Eerste submittie

```
//When a card is clicked, it should display up and all other cards should display down
//Get all the cards
let cards = document.getElementsByClassName("card");

//Add an event listener to each card
for (let i = 0; i < cards.length; i++) {

    cards[i].addEventListener("click", function() {
        if (this.innerHTML === "down") {
            this.innerHTML = "up";
        } else {
            this.innerHTML = "down";
        }
    });
}
```

Output	Tests: 0 pass / 4 fail
✗ Example case: Wrong answer ?	
✗ Click multiple cards: Wrong answer ?	
✗ Use a large card matrix: Wrong answer ?	
✗ Multiple clicks on the same card: Wrong answer ?	

Your score is 0% 😞

Tweede submittie

```
//When a card is clicked, it should display up and all other cards should display down
//Get all the cards
let cards = document.getElementsByClassName("card");

//Add an event listener to each card
for (let i = 0; i < cards.length; i++) {

    cards[i].addEventListener("click", function() {
        if (this.innerHTML === "down") {
            this.innerHTML = "up";
        }
        //set all other cards to "down"
        for (let j = 0; j < cards.length; j++) {
            if (j !== i) {
                cards[j].innerHTML = "down";
            }
        }
    });
}
```

Output

Tests: 4 pass / 0 fail

- ✓ Example case: Correct answer ?
- ✓ Click multiple cards: Correct answer ?
- ✓ Use a large card matrix: Correct answer ?
- ✓ Multiple clicks on the same card: Correct answer ?

Your score is 100%, perfect!

Totale tijd: 1 minuut 23 seconden.

ChatGPT

Ook ChatGPT was oorspronkelijk een beetje in de war, en ging er van uit dat er extra klassen aanwezig waren zoals "face-up".

Eerste submit



Here is an example of a setup function that could implement the logic for an HTML widget as described:



Copy code

```
function setup() {  
  // Get all the card elements  
  const cards = document.querySelectorAll('.card');  
  
  // Set up a click handler for each card  
  cards.forEach(card => {  
    card.addEventListener('click', () => {  
      // Turn over the previously face up card  
      const faceUpCards = document.querySelectorAll('.face-up');  
      faceUpCards.forEach(faceUpCard => {  
        faceUpCard.classList.remove('face-up');  
        faceUpCard.textContent = 'down';  
      });  
  
      // Turn over the clicked card  
      card.classList.add('face-up');  
      card.textContent = 'up';  
    });  
  });  
}
```

This function first gets all the elements with the class `card` (which should represent the clickable cells in the matrix), and sets up a click handler for each one. When a card is clicked, the function first turns over any previously face up cards by removing the `face-up` class and setting their text content to 'down'. It then turns over the clicked card by adding the `face-up` class and setting its text content to 'up'.

Note that this function assumes that the card elements have a `face-up` class applied when they are face up, and that their text content is set to 'up' or 'down' as appropriate. It also assumes that the card elements have a class of `card` applied to them. You may need to adjust the function to match the specific HTML structure and styling of your widget.

Output

Tests: 0 pass / 4 fail

- ✖ Example case: Wrong answer ?
- ✖ Click multiple cards: Wrong answer ?
- ✖ Use a large card matrix: Wrong answer ?
- ✖ Multiple clicks on the same card: Wrong answer ?

Your score is 0% 😞

Tweede submitie

KW

The contents of the elements with the card class should say "up" or "down", there are no extra classes present.



In that case, you can update the setup function to remove the references to the `face-up` class and use the text content of the cards to determine whether they are face up or face down. Here is an updated version of the function that does this:



Copy code

```
function setup() {  
  // Get all the card elements  
  const cards = document.querySelectorAll('.card');  
  
  // Set up a click handler for each card  
  cards.forEach(card => {  
    card.addEventListener('click', () => {  
      // Turn over the previously face up card  
      const faceUpCards = document.querySelectorAll('.card');  
      faceUpCards.forEach(faceUpCard => {  
        if (faceUpCard.textContent === 'up') {  
          faceUpCard.textContent = 'down';  
        }  
      });  
  
      // Turn over the clicked card  
      if (card.textContent === 'down') {  
        card.textContent = 'up';  
      }  
    });  
  });  
}
```

This function will work with card elements that have a text content of 'up' or 'down' and no additional classes applied to them. When a card is clicked, the function first turns over any previously face up cards by setting their text content to 'down', then it turns over the clicked card by setting its text content to 'up'.

Output Tests: 4 pass / 0 fail

✓ Example case: Correct answer ?

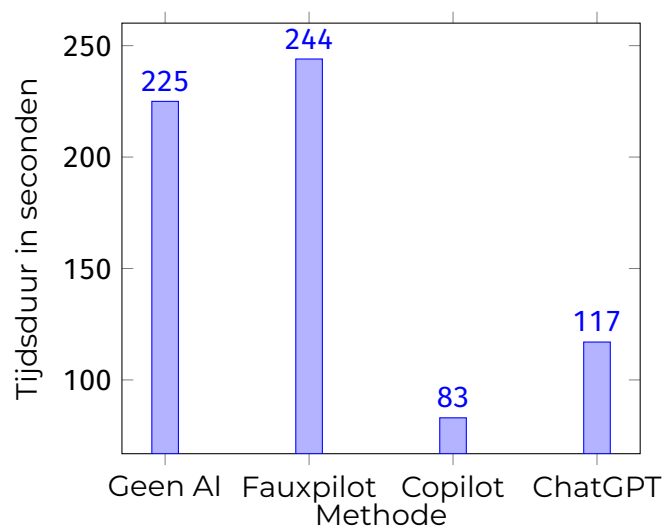
✓ Click multiple cards: Correct answer ?

✓ Use a large card matrix: Correct answer ?

✓ Multiple clicks on the same card: Correct answer ?

Your score is 100%, perfect!

Totale tijd: 1 minuut 57 seconden.



Figuur (4.2)

De tijdsverhouding in seconden tussen de verschillende oplossingen van opdracht 2. Hier was Fauxpilot net iets trager dan de opdracht handmatig uit te voeren, omdat de AI moeite had met het omdraaien van alle kaarten. Wederom is Copilot het snelst. Dit kan te maken hebben met het feit dat Copilot langere prompts lijkt aan te kunnen dan Fauxpilot. ChatGPT lag tussen de twee uitersten in, voornamelijk omdat het in die conversationele manier te werk gaat en elk woord apart uitschrijft, in plaats van één blok tekst terug te geven.

4.1.3. Opdracht 3

Geen AI hulp

Eerste submitie

```
Math.log100 = function(number) {  
    return Math.log(number) / Math.log10(100);  
};
```

Output Tests: 1 pass / 2 fail

✖ Example case: Wrong answer ?
✔ Function defined: Correct answer ?
✖ Several values: Wrong answer ?

Your score is 33% 😞

IntelliSense had in plaats van `log()` als eerste suggestie `log10()` gegeven, wat door te snel te willen zijn voor wat verwarring zorgde.

Tweede submit

```
Math.log100 = function(number) {  
    return Math.log(number) / Math.log(100);  
};
```

Output Tests: 3 pass / 0 fail

✔ Example case: Correct answer ?
✔ Function defined: Correct answer ?
✔ Several values: Correct answer ?

Your score is 100%, perfect!

Totale tijd: 1 minuut 55 seconden.

Fauxpilot

```
//Add a log100 function to the Math object that calculates a logarithm  
//with a base of 100  
Math.log100 = function(x) {  
    return Math.log(x)/Math.log(100);  
}
```

Output Tests: 3 pass / 0 fail

✔ Example case: Correct answer ?
✔ Function defined: Correct answer ?
✔ Several values: Correct answer ?

Your score is 100%, perfect!

Totale tijd: 35 seconden.

Copilot

```
//Add a log100 function to the Math object that calculates a logarithm
//with a base of 100.
Math.log100 = function(x) {
    return Math.log(x)/Math.log(100);
}
```

Output

Tests: 3 pass / 0 fail

- ✓ Example case: Correct answer ?
- ✓ Function defined: Correct answer ?
- ✓ Several values: Correct answer ?

Your score is 100%, perfect!

Totale tijd: 28 seconden

ChatGPT



Here is how you can add a `log100` function to the `Math` object:

[Copy code](#)

```
Math.log100 = function(x) {
    return Math.log(x) / Math.log(100);
}
```

You can then use the `log100` function like any other method on the `Math` object. For example:

[Copy code](#)

```
console.log(Math.log100(10)); // prints 0.5
```

Output Tests: 3 pass / 0 fail

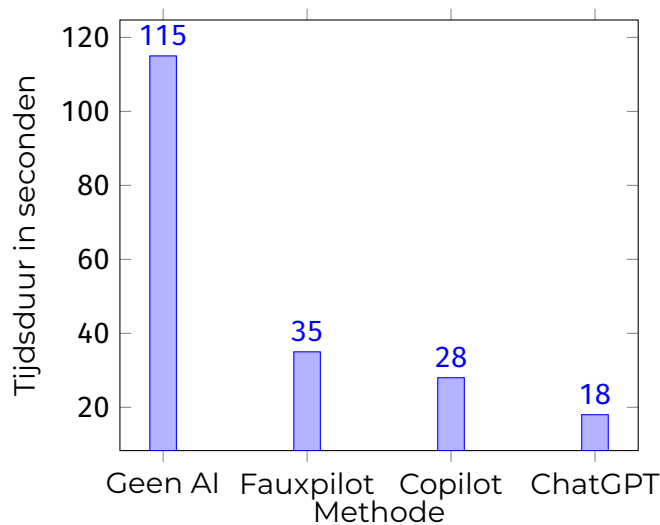
✓ Example case: Correct answer ?

✓ Function defined: Correct answer ?

✓ Several values: Correct answer ?

Your score is 100%, perfect!

Totale tijd: 18 seconden.



Figuur (4.3)

De tijdsverhouding in seconden tussen de verschillende oplossingen van opdracht 3. Hier liggen de resultaten veel dichterbij elkaar. Alle programmeerassistenten verstonden de opgave direct, en de grootste variabele was de snelheid bij het intypen van het tekstprompt.

4.2. Praktische testen

Voor Copilot en ChatGPT werd geen toestemming gegeven door CAT-Solutions om het effectief in een project te gebruiken, omdat dit voor Copilot met de gratis versie voor academische personen zou gebeuren, en er hierdoor geen controle was op welke data al dan niet werd bijgehouden door GitHub. Voor ChatGPT wordt ook alles bijgehouden en mogelijks gebruikt voor het verder trainen van de AI, en staat er bovendien bij de FAQ dat men moet vermijden om gevoelige data te gebruiken in de conversaties (Natalie, 2022). De testen voor deze twee zijn daarom uitgevoerd op een persoonlijke pc met het minimum aan nodige variabelen, en dit heeft een kleine invloed gehad op de gegenereerde resultaten.

4.2.1. BTW-nummers controleren

Dit ging heel vlot, maar wel met een iets andere methode dan de programmeur.

Programmeur

```
if ((rest.charAt(0) ≠ '0') && (rest.charAt(0) ≠ '1')){  
    throw new PPOperationFailedException(  
        MessageIRKStub.BELGIAN_VAT_NUMBER_STARTS_WITH_0_OR_1);  
}
```

Fauxpilot

```
//check if rest starts with a 0 or a 1, and if not throw an exception  
if (!rest.startsWith(`0`) && !rest.startsWith(`1`)){  
    throw new PPOperationFailedException(  
        MessageIRKStub.BELGIAN_VAT_NUMBER_MUST_START_WITH_0_OR_1);  
}
```

Copilot

```
//check if the rest variable starts with 0 or 1  
//and throw a PPOperationFailedException if not  
  
if (rest.StartsWith("0") || rest.StartsWith("1")){  
    throw new PPOperationFailedException("The VAT number is invalid");  
}  
else{  
    Console.WriteLine("The VAT number is valid");  
}
```

KW

Write a java check that checks if a string "rest" starts with a 0 or a 1, and throws a PPOperationFailedException if not



Here's an example of a method that checks if a string "rest" starts with either a "0" or a "1", and throws a PPOperationFailedException if it doesn't:

[Copy code](#)

```
class PPOperationFailedException extends Exception {  
    public PPOperationFailedException(String message) {  
        super(message);  
    }  
}  
  
public void checkString(String rest) throws PPOperationFailedException {  
    if (!(rest.startsWith("0") || rest.startsWith("1"))) {  
        throw new PPOperationFailedException("String does not start with 0 or  
1");  
    }  
}
```


You can then call this method and catch the exception like so:

[Copy code](#)

```
try {  
    checkString("rest");  
} catch (PPOperationFailedException e) {  
    System.out.println(e.getMessage());  
}
```


ChatGPT

Alternatively if you want to do that in one line

 Copy code

```
if (!(rest.startsWith("0") || rest.startsWith("1"))) throw new  
PPOperationException("String does not start with 0 or 1");
```

You should note that as this point since you are only checking for equality with two literals, you can just do

 Copy code

```
if(!rest.matches("^[01].*")) throw new PPOperationException("String does  
not start with 0 or 1");
```

You can also use ``Character#isDigit(char)`` and ``Character#forDigit(int, int)`` methods to check whether it is digit and whether it starts with either 0 or 1.

Fauxpilot kon uit de context afleiden welke soort uitzondering van toepassing was, en leverde een bericht mee dat zelfs nog niet bestond. Dit zou voor de andere programmeerassistenten hoogstwaarschijnlijk ook het geval zijn, als deze toegang hadden tot de context.

De oplossingen zijn zeer gelijkaardig, en zouden allemaal perfect werken. De vergelijkingsmethode van de programmeur heeft echter iets minder overhead, aangezien deze maar één karakter vergelijkt. In de meeste applicaties heeft dit weinig belang, maar als men een programma zo efficiënt mogelijk wil maken, tellen alle beetjes op.

4.2.2. Countrycode eindigend op B01

Fauxpilot

```
// if the countryCode equals "NL", check if the string vatNumber ends with "B01"  
if (countryCode.equals("NL") && !rest.endsWith("B01")){  
    throw new PPOperationException(  
        MessageIRKStub.NEDERLAND_VAT_NUMBER_MUST_END_WITH_B01);  
}
```

Copilot

```
// if the countryCode equals "NL", check if the string vatNumber ends with "B01"
```



```
if (countryCode == "NL"){  
    if (vatNumber.EndsWith("B01")){  
        Console.WriteLine("The VAT number is valid");  
    }  
    else{  
        throw new PPOperationFailedException("The VAT number is invalid");  
    }  
}
```

ChatGPT



create a java function that checks if a countryCode equals "NL", and if so, checks if the string vatNumber ends with "B01"



You can create a Java function to check if a given country code and VAT number match the specified conditions using an `if` statement. Here's an example:

Copy code

```
public boolean checkVATNumber(String countryCode, String vatNumber) {  
    if (countryCode.equals("NL")) {  
        if (vatNumber.endsWith("B01")) {  
            return true;  
        }  
    }  
    return false;  
}
```

In the above example, the function takes in two parameters: a `countryCode` and a `vatNumber`. The function uses an `if` statement to check if the `countryCode` is equal to "NL". If it is, the function uses another `if` statement to check if the `vatNumber` ends with "B01". If both conditions are true, the function returns `true`. Otherwise, it returns `false`.

You can also use the ternary operator to make it shorter and more readable.

Copy code

```
public boolean checkVATNumber(String countryCode, String vatNumber) {  
    return countryCode.equals("NL") && vatNumber.endsWith("B01") ? true : false;  
}
```

This is similar with the previous example but using ternary operator to return the boolean value

Bij Fauxpilot kwam al een eerste probleem kijken: de variabele `rest` was een sub-

string van de variabele *vatNumber*. In plaats van het volledige BTW nummer te checken, was Fauxpilot er van overtuigd dat de rest variabele gebruikt moest worden. Soms kon dit verholpen worden door in de lijn commentaar het type van de variabele en de variabelenaam expliciet te schrijven, en niet bijvoorbeeld de woorden “vat number”, maar dit was helemaal geen consistente manier van werken. Dit is geen goed teken voor Fauxpilot: als de code niet expliciet wordt nagekeken door de programmeur kunnen hierdoor verschillende bugs en fouten optreden in de werking van de software.

De andere assistenten gingen hier vlotter mee overweg. Copilot ging direct van start met de *countryCode* variabele, die zelfs niet in het bestand bestond. ChatGPT maakte hier natuurlijk ook geen probleem van, aangezien voor elk van deze opdrachten een aparte thread was opgestart. Ook kon ChatGPT niet op zichzelf afleiden wat er met de vergelijking moest gebeuren, maar dat was in dit geval geen voor- of nadeel. De oplossing die hierdoor gegeven werd is even geldig op het gegeven prompt.

4.2.3. Case functions

Copilot

```
//create a switch case for each countryCode "NL", "DE", "EN"  
  
//create a function for each countryCode  
  
//create a function for each countryCode  
  
//create a function for each countryCode  
  
//create a function for each countryCode  
  
//create a function for each countryCode
```

Fauxpilot

```
//create a case function for each countryCode "NL", "DE", "EN".  
  
  
//The function will return the correct VAT number  
//if the countryCode equals "NL", "DE", "EN"  
//and the rest of the VAT number equals "B01"  
//and the rest of the VAT number equals "B02"  
//and the rest of the VAT number equals "B03"  
//and the rest of the VAT number equals "B04"
```

```
//and the rest of the VAT number equals "B05"  
//and the rest of the VAT number equals "B06"
```

ChatGPT

KW

Create a java case function for each countryCode "NL", "DE", "EN"



Here is an example of a Java method that takes a `countryCode` as a parameter and returns a string that corresponds to the country code:

[Copy code](#)

```
public static String getCountryName(String countryCode) {  
    switch (countryCode) {  
        case "NL":  
            return "Netherlands";  
        case "DE":  
            return "Germany";  
        case "EN":  
            return "United Kingdom";  
        default:  
            return "Invalid country code";  
    }  
}
```

You can call this method with a country code as the argument to get the corresponding country name.


For example:

[Copy code](#)

```
String countryName = getCountryName("NL");
```

countryName variable will be assigned with the value of Netherlands

Also you can add more case for other country codes.

 Copy code

```
public static String getCountryName(String countryCode) {  
    switch (countryCode) {  
        case "NL":  
            return "Netherlands";  
        case "DE":  
            return "Germany";  
        case "EN":  
            return "United Kingdom";  
        case "FR":  
            return "France";  
        case "JP":  
            return "Japan";  
        default:  
            return "Invalid country code";  
    }  
}
```

This way you can add more countries for more country codes as you desire.

Bij zowel Fauxpilot als Copilot kwam de suggestie vast te zitten in een oneindige lus van commentaar, en werden er geen lijnen code gesuggereerd. Enkel ChatGPT ging goed om met case functions, en gaf zelfs suggesties voor het uitbreiden ervan.

4.2.4. Tot 27 tellen

Deze laatste opdracht klinkt misschien wel de gemakkelijkste, maar bleek ook de meest inconsistente te zijn: Een simpele functie die tot 27 moet tellen. Deze opdracht werd voor Fauxpilot wel getest in drie verschillende “omgevingen”, om na te gaan of de context iets met de vreemde uitkomsten te maken had: Een bestand waar al bestaande logica in stond op een bedrijfspc, een bestand waar nog niets in stond op dezelfde bedrijfspc, en een bestand waar nog niets in stond op een andere, persoonlijke pc.

Copilot

```
//write a function that counts to 27
```

```
//write a function that counts to 27 and prints out the number
```

```
//write a function that counts to 27 and prints out the number and the square
//of the number
```

```
//write a function that counts to 27 and prints out the number and the square
//of the number and the cube of the number
```

```
//write a function that counts to 27 and prints out the number and the square
//of the number and the cube of the number and the square root of the number
```

Fauxpilot

Bestand met bestaande logica

```
// write a function that counts to 27
public static int countTo27(int number) {
    int count = 0;
    while (number > 0) {
        count++;
        number = number / 10;
        if (number == 0) {
            break;
        }
    }
    return count;
}
```

Nieuw bestand op bedrijfspc

```
//write a function that counts to 27 in java

import java.util.Scanner;

public class CountTo27 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner input = new Scanner(System.in);
        System.out.println("Enter a number to count to 27");
        int number = input.nextInt();
        int count = 0;
        while(number > 0){
            count++;
            number = number - 1;
        }
    }
}
```

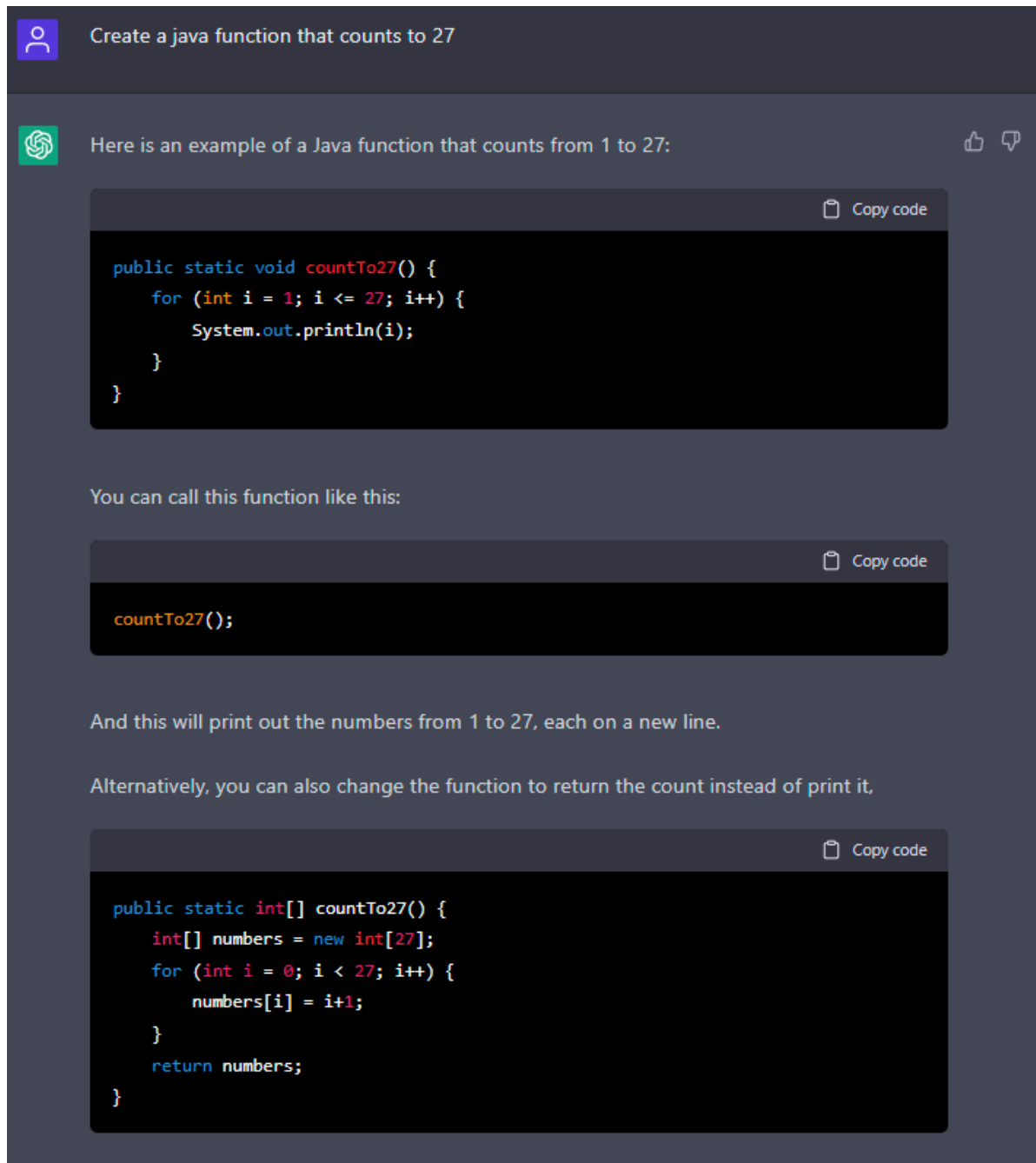
```
        }  
        System.out.println("The number you entered is " + number);  
        System.out.println("The number you counted to is " + count);  
    }  
  
}
```

In dit bestand begon Fauxpilot eerst zelfs suggesties te geven voor C++ in plaats van voor java, dus moest er gespecificeerd worden dat de suggestie voor java was. Deze functie telde enkel tot 27 wanneer de gebruiker als input 27 gaf, maar deed dit op een zeer omslachtige manier: zolang het ingegeven getal groter was dan 0, werd er bij de telling opgeteld, en afgeteld bij het ingegeven getal. Daarna werd ook uitgeprint welk getal de gebruiker had ingegeven, maar op dit punt in de functie was dit altijd 0.

Nieuw bestand op persoonlijke pc

```
public static void main(String[] args) {  
    int count = 0;  
    while (count < 27) {  
        count++;  
        System.out.println(count);  
    }  
}
```

Het is niet zo evident om te zeggen waarom deze uitkomsten zo veel verschillen van elkaar, maar het is wel duidelijk dat Fauxpilot niet altijd even betrouwbaar is.



The screenshot shows a ChatGPT interface with a user prompt and two AI responses. The user prompt is: "Create a java function that counts to 27". The first AI response provides a Java function that counts from 1 to 27 and prints each number. The second AI response shows how to call this function. The third AI response provides an alternative function that returns the count instead of printing it.

Create a java function that counts to 27

Here is an example of a Java function that counts from 1 to 27:

```
public static void countTo27() {  
    for (int i = 1; i <= 27; i++) {  
        System.out.println(i);  
    }  
}
```

You can call this function like this:

```
countTo27();
```

And this will print out the numbers from 1 to 27, each on a new line.

Alternatively, you can also change the function to return the count instead of print it,

```
public static int[] countTo27() {  
    int[] numbers = new int[27];  
    for (int i = 0; i < 27; i++) {  
        numbers[i] = i+1;  
    }  
    return numbers;  
}
```

Wederom gaf ChatGPT een simpel en juist antwoord op de vraag.

5

Conclusie

De wereld van AI vordert steeds meer en meer. Modellen met miljarden parameters zullen met de tijd ook alledaagser worden, als ze dat zelfs al niet zijn. Voor deze miljarden parameters is natuurlijk veel rekenkracht nodig, en die is nog niet goedkoop genoeg om nuttige, grote modellen lokaal op te zetten. Bestaat er dus op dit moment een volwaardige AI-programmeerassistent die eigendomsrechten respecteert en gebruikt kan worden in een groot brownfield project met veel klassen en complexiteit? De technologieën die hier onderzocht werden, voldoen niet aan de vereisten.

Eerst en vooral is de schending van eigendomsrechten in automatisch gegenereerde code moeilijk te quantificeren. De schendingen worden wel vastgesteld, maar hoe vaak dit voorkomt is moeilijk te zeggen omdat dit ook vaak achter gesloten deuren gebeurt.

Er zijn dus wel alternatieven voor Copilot die er wel op letten om geen eigendomsrechten schenden, zoals Fauxpilot en TabNine. TabNine kon door de beperkte tijd en het feit dat de gratis versie niet veel meer doet dan simpele IntelliSense niet opgenomen worden in het praktische gedeelte van dit onderzoek. Verder onderzoek zou kunnen uitwijzen of TabNine geschikter is dan Fauxpilot, of dat het dezelfde problemen ondervindt die Fauxpilot (en Copilot zelf, in mindere mate) ook heeft. Fauxpilot is een interessante technologie die met de jaren heel aantrekkelijk kan worden, maar het nog net niet is. Bij het model met 6 miljard parameters kunnen de suggesties soms perfect doen wat de programmeur wil, *als* deze precies weet wat hij of zij wilt bereiken. Zo kan Fauxpilot bijvoorbeeld veelgebruikte methodes uit één bestand detecteren en hergebruiken, maar bij variabelen lijkt het iets moeilijker om aan te wijzen welke er juist gebruikt moet worden. Zelfs al is de gebruiker zeer specifiek, kan Fauxpilot nog steeds blijven hangen op één bepaalde variabele, die onnodig was voor de opdracht, en niet gespecificeerd werd. GitHub copilot gaat hier beter mee om, en ook andere openstaande tabbladen in de IDE worden mee-

genomen in de context, wat voor een beter resultaat kan zorgen.

Lijn per lijn in commentaar zetten wat men wilt bereiken kan dus soms werken, maar is niet betrouwbaar genoeg om altijd de juiste code te schrijven die de programmeur bedoelt. Zelfs wanneer het een goede suggestie geeft, kan het soms even snel zijn om simpelweg de lijn code zelf te schrijven. En ook in het geval dat de programmeur heel specifiek is kan Fauxpilot nog steeds tilt slaan en in plaats van code suggesties commentaar suggesties geven, die soms wel van toepassing kunnen zijn, maar ook overbodig zijn. Daarna is de kans ook groot dat de suggestie gewoon blijft herhalen, zonder een makkelijke manier om uit deze oneindige lus te geraken en toch een degelijk antwoord te genereren.

Ook de snelheid kon beter bij zowel Fauxpilot als Copilot. Afhankelijk van de instelling van het aantal tokens dat teruggegeven werd (dus hoe lang de suggestie per keer is) bij Fauxpilot, kon het zijn dat men enkele seconden moest wachten per antwoord van de backend. Fauxpilot is natuurlijk niet echt bruikbaar als het al sneller zou zijn om zelf de code uit te schrijven. De middenweg van het aantal tokens lag rond de 20, wat meestal net iets meer dan 1 lijn code was. Bij Copilot, waar het aantal tokens niet vast staat, kan het ook enkele seconden duren vooraleer er een suggestie getoond wordt.

Gezien de beperkte duurtijd van de bachelorproef en het budget dat toegewezen was aan hardware was het niet mogelijk om het groter Fauxpilot model van 16 miljard parameters te testen tegenover het 6 miljard parameter model, en het model te trainen of finetunen met een zelfgemaakte dataset van broncode die bestaat uit voldoende lijnen bestaande java code. Er was wel informatie beschikbaar over hoe dit zou kunnen gebeuren (Dolan-Gavitt, 2022), maar dit werd niet verder onderzocht. De limitatie van het budget zal met de tijd hopelijk verkleinen of zelfs verdwijnen, zolang de wet van Moore nog geldig is, en de onderzochte technologieën zullen ook verbeteren zolang er vooruitgang geboekt wordt in de wereld van de Artificiële Intelligentie.



Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1. Introductie

Deze bachelorproef richt zich op het gebruik van AI tijdens het programmeren. Ondertussen klinkt GitHub Copilot al redelijk bekend in de oren van developers die mee zijn met de laatste technologieën. Hoewel Copilot een van de bekendere AI programmeer-assistenten is, zijn er ook enkele minpunten aan. Het voornaamste hiervan is het gebruik van publieke code op het internet, ongeacht de licentie die hier aan gekoppeld staat. Het is mogelijk dat Copilot volledige functies aanbiedt die uit een codebase komen met een licentie die hergebruik of wijziging verbiedt. Dit zorgt natuurlijk voor juridische problemen die elk IT-bedrijf wil vermijden.

Er is echter ook een open source project waarbij een AI model lokaal kan draaien en door de gebruikers zelf te trainen is. Het doel van deze bachelorproef is het onderzoeken van de bruikbaarheid van dit project: FauxPilot. *Wat is er nodig om deze AI te trainen om bruikbare resultaten te verkrijgen? Welk percentage van de te schrijven code kan gegenereerd worden aan de hand van FauxPilot, en met hoe veel lijnen code moet het model getraind worden om dit resultaat te verkrijgen? Een belangrijk detail zal ook zijn hoeveel bugs er in de gegenereerde code zitten.*

Dit onderzoek moet uitwijzen of FauxPilot een bruikbaar alternatief kan bieden voor IT-bedrijven die niet onwetend plagiaat willen plegen en/of niet willen dat hun gepatenteerde codebase gebruikt wordt om het model van anderen te trainen.

A.2. State-of-the-art

Ondertussen bestaan er al verschillende opties qua AI programmeer-assistenten. Enkele voorbeelden hiervan zijn Codiga, Tabnine, Kite en de eerder vermelde GitHub Copilot. Tabnine is gebaseerd op GPT-2, een Natural Language Processing (NLP) model getraind in het voorspellen van woorden op basis van 40GB aan teksten gevonden op het internet (Radford e.a., 2019). GPT-2 gebruikt een Transformer architectuur, wat er voor zorgt dat het model zeer groot kan zijn. Dit zorgt voor betere resultaten, maar is ook moeilijker om te trainen. Kite gebruikt een model getraind door het bedrijf zelf, omdat volgens hen NLP modellen minder goed werken met code door de structuur, die verschilt van gewone spreektaal (Lardinois, 2019). GitHub Copilot maakt gebruik van OpenAI Codex, een afstammeling van GPT-3 (Zaremba e.a., 2021), wat op zijn beurt de opvolger is van GPT-2. Het grootste deel van deze gegenereerde code kan als transformatief gezien worden, maar er kunnen ook volledige stukken code letterlijk gekopieerd in staan, zoals bijvoorbeeld een het geval was wanneer het gevraagd werd om een About Me pagina te genereren. Op deze pagina stond een link naar een GitHub account van een bestaande gebruiker (Gershgorin, 2021). Hoewel dit een voorbeeld eerder onschadelijk is, geeft het wel aan dat er een groter probleem kan zijn.

Er zijn namelijk verschillende licenties waaronder software kan vallen (Odenice, 2022). Bij Permissive licensing is modificatie en herverdeling toegestaan, vaak op voorwaarde dat de originele auteur vernoemd wordt. Voorbeelden hiervan zijn de Apache License en de vaak gebruikte MIT license. Copyleft licenties zorgen er voor dat code gemodificeerd mag worden, maar de veranderingen die gemaakt worden moeten ook open source onder de zelfde licentie gepubliceerd worden. De code mag echter ook in gepatenteerde code gebruikt worden. Tussen Permissive en Copyleft licenties bestaat er ook nog zwakke copyleft. Een voorbeeld hiervan is de GNU Lesser General Public License, of LGPL. Dit soort licentie zorgt er voor dat dynamisch gelinkte software verdeeld mag worden onder om het even welke licentie, zonder veel extra werk. Als de software statisch gelinkt wordt, of gemodificeerd wordt, komen er snel regels van de copyleft licensing bij kijken.

Als laatste zijn er de twee uitersten: gepatenteerde licenties waarbij de copyright houder het specifiek verbiedt om bijvoorbeeld de code te delen, modificeren, herverdelen enzoverder. Het andere uiterste is code in het publieke domein. Deze code valt niet onder copyright wetgeving, en kan door iedereen gebruikt en gemodificeerd worden. Dit is echter zeldzaam en kan verschillen tussen jurisdicties.

Een programmeerassistent dat door de gebruiker zelf te trainen is, kan een oplossing bieden voor het probleem dat licenties met zich meebrengen.

A.3. Methodologie

In deze bachelorproef wordt eerst en vooral onderzocht welke licenties allemaal bruikbaar zijn zonder dat men zich zorgen moet maken om schending van het auteursrecht. Met deze informatie zal dan gezocht worden naar geschikte codebases die gebruikt mogen worden om het FauxPilot model te trainen, en wordt er ook uitgezocht hoe dit juist in zijn werk gaat.

Na het model initieel op te zetten, worden op verschillende punten in de training metingen gedaan om te achterhalen of het model robuust genoeg is om effectief mee te programmeren. Met andere woorden, hoe groot is het percentage code gegenereerd door de AI, is het effectief de gewenste code en zijn er bugs aanwezig? Het plan is om dit te testen aan de hand van een te implementeren user story van CAT-solutions. Dan kan er bij elke iteratie van het model, dus na elke toevoeging van een codebase, bekeken worden of er vorderingen zijn in de efficiëntie van de assistent en of het de gewenste code genereert. Een belangrijk detail hiervan zal zijn om zeker niet de code die geïmplementeerd werd om een vorige iteratie te testen, terug in het model te stoppen. Dit zou voor vertekende resultaten kunnen zorgen, aangezien een volgende iteratie deze code woord voor woord zou kunnen overnemen. Het idee zou dan zijn om aan te duiden welke suggesties van de AI komen, en welke code door de programmeur zelf aangevuld moest worden.

A.4. Verwacht resultaat, conclusie

Het verwachte resultaat is dat FauxPilot tussen 25% en 50% van de code zou kunnen genereren, afhankelijk van de complexiteit en van hoe groot het totale aantal lijnen code is dat gebruikt kan worden bij de training. Deze verwachting is gebaseerd op enkele kleine testen uitgevoerd met GitHub copilot, waarbij de AI enkele simpele functies moest genereren. IT-bedrijven die geïnteresseerd zijn in het gebruiken van AI programmeer-assistenten kunnen, afhankelijk van dit resultaat en het werk dat er aan vooraf gaat, efficiënter een beslissing maken of FauxPilot een bruikbare tool is.

Bibliografie

(g.d.). <https://www.testdome.com/>

BM, N. (2021). What Is an Encoder Decoder Model? <https://towardsdatascience.com/what-is-an-encoder-decoder-model-86b3d57c5e1a>

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language Models are Few-Shot Learners. <https://doi.org/10.48550/ARXIV.2005.14165>

CreativeCommons. (g.d.-a). https://wiki.creativecommons.org/wiki/PDM_FAQ

CreativeCommons. (g.d.-b). <https://creativecommons.org/publicdomain/mark/1.0/>

Dolan-Gavitt, B. (2022). Guide on how to train new models on an existing codebase? · Discussion 74 · fauxpilot/fauxpilot. <https://github.com/fauxpilot/fauxpilot/discussions/74>

EPO. (g.d.). What is prior art? <https://www.epo.org/learning/materials/inventors-handbook/novelty/prior-art.html>

FOSSA. (2021). All About Permissive Licenses. <https://fossa.com/blog/all-about-permissive-licenses/>

Gage, P. (1994). FEB94 A New Algorithm for Data Compression. <http://www.pennelynn.com/Documents/CUJ/HTML/94HTML/19940045.HTM>

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., & Leahy, C. (2020). The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027*.

Gershgorn, D. (2021, juli 7). *GitHub's automatic coding tool rests on untested legal ground*. Verkregen oktober 21, 2022, van <https://www.theverge.com/2021/7/7/22561180/github-copilot-legal-copyright-fair-use-public-code>

GNU. (g.d.). What Is Copyleft? <https://www.gnu.org/licenses/copyleft.en.html>

GNU. (2016). The GNU General Public License v3.0. <https://www.gnu.org/licenses/gpl-3.0.en.html>

GNU. (2022). GNU Affero General Public License. <https://www.gnu.org/licenses/agpl-3.0.html>

Grover, A. (2018). Autoregressive Models. <https://deepgenerativemodels.github.io/notes/autoregressive/>

- Jaswal, A. S. (2021). Byte Pair Encoding — The Dark Horse of Modern NLP. <https://towardsdatascience.com/byte-pair-encoding-the-dark-horse-of-modern-nlp-eb36c7df4f10>
- Lardinois, F. (2019, januari 28). *Kite raises 17M for its AI-driven code completion tool*. Verkregen oktober 6, 2022, van <https://techcrunch.com/2019/01/28/kite-raises-17m-for-its-ai-driven-code-completion-tool/>
- LinusMediaGroup. (2022). Luke LOVES this AI chat bot. https://www.youtube.com/watch?v=3yUPdYK9E2g&ab_channel=LMGClips
- Medvedovsky, S. (2022, juli 7). *GitHub Copilot and Open Source: A love story that won't end well?* Verkregen oktober 5, 2022, van <https://thenewstack.io/github-copilot-and-open-source-a-love-story-that-wont-end-well/>
- Nahum, D. (2021). 3 Weeks into the GitHub CoPilot secrets leak - What have we learned. <https://spectralops.io/blog/ai-ml-in-code-githubs-copilot-leak/>
- Natalie. (2022). <https://help.openai.com/en/articles/6783457-chatgpt-faq>
- Nijkamp, E., Pang, B., Hayashi, H., Tu, L., Wang, H., Zhou, Y., Savarese, S., & Xiong, C. (2022). CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis. *arXiv preprint*.
- Odense, P. (2022, juli 27). *Five types of software licenses you need to understand*. Verkregen oktober 21, 2022, van <https://www.synopsys.com/blogs/software-security/5-types-of-software-licenses-you-need-to-understand/>
- OpenAI. (2022). ChatGPT: Optimizing Language Models for Dialogue. <https://openai.com/blog/chatgpt/>
- Radford, A., Wu, J., Amodei, D., Amodei, D., Clark, J., Brundage, M., & Sutskever, I. (2019, februari 14). *Better Language Models and Their Implications*. Verkregen oktober 6, 2022, van <https://openai.com/blog/better-language-models/>
- Smith, A. (2022). This is a public version of the main Kite repo. <https://github.com/kiteco/kiteco-public>
- snowdrone. (2022). Chat GPT sources and citations. https://www.reddit.com/r/OpenAI/comments/zyp4ey/chat_gpt_sources_and_citations/
- Southern, M. G. (2023). OpenAI May Introduce A Paid Pro Version Of ChatGPT. <https://www.searchenginejournal.com/openai-chatgpt-professional/476244/#close>
- TabNine. (g.d.). <https://www.tabnine.com/code-privacy>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *CoRR*, *abs/1706.03762*. <http://arxiv.org/abs/1706.03762>
- Zaremba, W., Brockman, G., & OpenAI. (2021, augustus 10). *OpenAI Codex*. Verkregen oktober 6, 2022, van <https://openai.com/blog/openai-codex/>