Rapport Final - Yan Rabefaniraka

Yan Rabefaniraka, stagiaire de Meriem Ouederni et David Brunet

31 Juillet 2025

Contents

Remarques :

- Fichier engendré avec:

  ```
  pandoc --toc -o ../pdfs/report1_yan_rabefaniraka.pdf report1_yan_rabefaniraka.md
  ```

- Par facilité d'écriture, on utilisera le masculin 'étudiant' pour qualifier les étudiant.es.

Sujet de stage

Pour ce stage, il m'est demandé en tant qu'étudiant en sciences du numérique d'établir une **étude de cas approfondie** sur l'utilisation actuelle par les étudiants de l'IA générative dans ce domaine— en particulier des modèles de langage— et d'en déduire une possible utilisation plus efficace pour l'apprentissage. Mon expérience personnelle et mon statut d'étudiant sont un bon apport de départ, accompagnés par ma tutrice qui enseigne directement aux concerné.es.

Cette mission n'a **pas de filière cible** et n'est pas seulement dirigée vers celles spécialisées en informatique, tout d'abord parce que l'aide que l'IA pourrait fournir dans les études s'applique à tous les domaines, mais aussi parce que l'importance grandissante des petits cas d'usage de l'informatique dans les études supérieures (e.g savoir mettre sur un graphe avec *Python/Matlab* ses données d'expérience, ou encore faire des statistiques sur *RStudio*) permet de se questionner sur les facilitations que l'IA offre à celles et ceux dont le futur métier ne se trouve pas dans les sciences du numérique.

Ainsi, les tests effectués et les conclusions déduites de cette étude de cas— spécifiquement pour les sciences du numérique— doivent être applicables pour un **niveau débutant**, voire en particulier pour des langages haut niveau comme le *Python*.

Cependant, on peut aussi vérifier la puissance / les limites de l'utilisation de l'IA Générative dans l'apprentissage d'une matière complexe. Pour cela, **l'Ingénierie Dirigée par les Modèles**, enseignée (par Mme Ouederni aux 2ème années) à l'ENSEEIHT sera certainement choisie: C'est une matière suffisamment théorique mais avec des applications directes dans les différents processus de développement/industriels, avec une grande part de notions graphiques. Elle s'enseigne aussi avec 7 cours de *Travaux pratiques* sur machine, avec code et modélisation donc. On pourra ainsi vérifier ce que l'IA peut proposer à un étudiant qui cherche à l'utiliser pour travailler la matière en autonomie.

L'objectif final de cette étude de cas étant la création d'une formation / d'un module qui offrirait aux étudiants de tout horizon un **set d'outils** pour leur enseigner les bonnes pratiques, il n'est pas anodin de donner les bases de l'IA génératives à travers une courte vulgarisation du fonctionnement de la GenAI et des LLMs pour les non-initié.es. Le public étant large il est aussi intéressant de mentionner les impacts environnementaux en rappelant les coûts en énergie et en eau de l'apprentissage et de la maintenance des services, de la place des utilisateurs dans la boucle (apprentissage actif, données personnelles) mais aussi des limites de l'IA *i.e* les hallucinations, les erreurs, les biais causés par le *data set*, les biais de confirmation de ChatGPT, etc. Mais, il est aussi essentiel de leur montrer à quel point ces outils sont puissants, en parlant de tout ce qui est inclus avec ChatGPT (compilateur LateX, Python), la gratuité de Github Copilot pour tout étudiant avec au choix l'utilisation de modèles de Claude Sonnet, GPT, et Gemini, ainsi que les modèles *offline*. Cela inclut donc un comparatif des différents modèles de langage gratuits actuellement sur le marché, par tâche et *cas d'utilisation* afin de comparer leur pertinence.

Enfin, cela implique donc d'établir des **critères de qualité/validation** de réponses selon le domaine, les prompts, l'IA, etc. Pour ce faire, plusieurs tests seront effectués et mis sur tableau tout au long du stage. Des templates seront également préparés pour d'éventuels *Jupyter Notebooks* (à lancer sur *Google Colab* par exemple). Ces derniers facilitent l'interactivité et augmenteront donc l'accessibilité du projet.

Intérêt de la mission en tant qu'étudiant avec une professeure pour tutrice

En tant qu'étudiant en sciences du numérique, je vais donc me servir de mes propres expériences avec l'IA Générative et vais partager l'usage que j'en fais ou que j'ai pu constater lors de ma $1^{ère}$ année, que ce soit pour y apporter un regard critique ou m'en servir de base pour développer la suite du projet.

Ressenti - étudiant

Dans cette section, je vais donc étayer les différents usages de l'IA Générative que j'ai jusque là pu observer.

Utilisation de l'IA par le corps apprenant

Apprentissage

On constate plusieurs cas d'utilisation courants que font les étudiants de l'IA générative dans l'apprentissage, en particulier de ChatGPT:

- **Trouver des solutions à des problèmes spontanés** qu'ils rencontrent lors de l'apprentissage d'un cours ou la résolution d'exercices, quand aucun professeur n'est à disposition pour y répondre. Cette utilisation est de plus en plus courante quand les étudiants résolvent des exercices pour lesquels il n'y a pas de correction claire et où ils ne sont donc pas assurés d'avoir les bonnes réponses, voire d'avoir compris le cours. Cela peut aussi

s'appliquer à la programmation, quand la documentation pour une librairie est difficile à trouver ou à lire et qui incite régulièrement les étudiants à demander l'explication de lignes de code à un Chatbot. On peut par exemple penser aux différentes fonctions intégrées de *Matlab* dont la compréhension peut parfois être sinueuse. On peut aussi penser au développement *Web* où les étudiants peuvent demander à ChatGPT de leur expliquer les différentes balises *CSS* ou *HTML*.

- **Expliquer/résumer un cours** qu'ils n'ont pas compris ou dont ils veulent vérifier leur propre compréhension. Cela peut s'apparenter à de l'aide personnalisée, avec les étudiants qui fournissent la section du cours qu'ils n'ont pas comprise. De même, certains s'en servent pour générer des questions typiques ou des quiz, comme des équivalents à la plateforme *Quizlet*.

AJOUTER IMAGES ICI pour illustrer avec Graphes python, code Matlab.

Facilitation des tâches pénibles

Une utilisation courante est aussi celle de la **facilitation des tâches fastidieuses** (répétitives, peu challengeantes) en passant par l'IA. On rencontre le plus ce cas d'utilisation lors de de TPs ou de projets qui nécessitent du code répétitif et peu demandant en réflexion technique pour les étudiants. On peut notamment penser aux méthodes répétitives en programmation orientée objet, comme des *getters* ou des *setters*, ou encore aux matières dont le code n'est pas le sujet principal mais un support d'application (algèbre linéaire, automatique ou télécommunications par exemple).

Triche

On observe parfois des cas où les étudiants font faire tout leur projet / TP par l'IA, sans être capable de reproduire ni *a minima* d'expliquer le résultat. Cela a un apport nul pour l'apprentissage et est donc le cas qu'on cherche à éviter. On peut se baser notamment sur la récente étude menée par des chercheurs du MIT qui ont conclu à une baisse des capacités cognitives liées à l'utilisation exclusive de ChatGPT par des étudiants pour écrire leurs dissertations; En effet, il ne faudrait pas que la facilité d'utilisation de l'IA nuise à l'intégrité réflexive des apprenants, en particulier quand ces derniers n'ont pas encore acquis l'expertise dans le domaine concerné.

Objectif: Co-cognition

L'objectif visé serait pour l'étudiant d'atteindre la co-cognition: Un stade où, ni l'IA ni l'étudiant n'auraient été capables de réaliser le code sans l'autre. C'est-à-dire que le code final serait bien meilleur que celui proposé en un *prompt* par l'IA, et que l'étudiant se voit capable d'expliquer chaque étape puisqu'il aura directement contribué à chacune. Cela implique que ce dernier sait exactement ce qu'il veut et qu'il est capable d'évaluer le code que lui propose l'IA, de revenir dessus, éventuellement de l'améliorer ou de le corriger.

Pistes d'amélioration vers la co-cognition

Les principes que j'ai utilisés pour essayer d'approcher la co-coginition (/ co-construction) sont: - **Role prompting** → Définir un rôle et des compétences spécifiques à l'IA (e.g: "Tu es un programmeur fonctionnel"). Cela permet d'ajouter du contexte qui dirige la génération de l'IA vers un résultat plus spécialisé; C'est en particulier efficace avec des IAs généralistes comme **ChatGPT**. Cela s'est révélé moins impactant avec **Github Copilot** qui est *fine-tuned* avec les répertoires publiques **Github**, et qui peut donc très bien faire le travail sans. - **Contexte détaillé** → Fournir les spécifications complètes du projet c'est-à-dire les consignes du projet données par le professeur concernant le code et le raisonnement, mais aussi ce qu'on attend exactement de l'IA, par exemple "expliquer le code/la syntaxe" ou encore "écrire un squelette modulaire du code". - **Collaboration par itération**: Être capable d'intéragir avec les réponses de l'IA. Cela implique de travailler de son côté sur ce qu'elle renvoie, comme corriger son code, repérer les erreurs de compréhension (ou hallucinations), les manquements, ou simplement améliorer les prémisces qu'elle propose. La co-coginition/co-construction ressemblera inévitablement à une discussion entre deux collègues qui améliorent ou questionnent tour à tour leur proposition. - **Connaissances établies dans le domaine**: Pour que tous les points précédents soient efficaces, cela nécessite de l'utilisateur qu'il soit au moins légèrement éduqué voire expert dans le domaine sur lequel il utilise l'IA. En clair, il a besoin de connaître les bases de la syntaxe du langage de code/modélisation utilisé, les détails exacts du sujet avec les attentes et contraintes, et est capable de questionner les générations de l'IA. Son but est de rebondir sur ces dernières pour produire encore mieux; elles peuvent servir de base de réflexion ou de base de code pour quelque chose qu'il saurait faire correctement/décemment tout seul.

Tests menés sur des cas d'utilisation en code

Pour l'étude de cas approfondie, nous avons pris deux sujets distincts dans leur exigence. Cela nous permettra de nous faire une idée sur la viabilité d'utiliser une IA pour un cas accessible comme pour un cas plus complexe.

Pour plus de détails, voir ici.

Jeu des allumettes - Cas grand public

Le jeu des allumettes est un sujet de projet très court que j'ai eu à réaliser en **Ada** puis en **Java**. La légère complexité vient des contraintes imposées qui sont claires et rigides, et qui permettent d'évaluer la capacité d'un étudiant à implémenter un cahier des charges très explicite dans ses attentes.

Écriture d'un sujet

La première étape a été de réécrire simplement le sujet du jeu des allumettes. Il inclut les règles du jeu et conditions de victoire, ainsi que les contraintes de programmation et exigences. J'y aussi mis des extraits d'affichage suffisamment complet pour qu'il soit reproductible. Cela pourra servir de base pour évaluer les réponses de l'IA et ainsi savoir comment mieux poser les prompts à partir du sujet directement. La qualité des prompts sera aussi auto-évaluée selon la grille du PPAi6, au même titre que la qualité des réponses sera évaluée selon le code obtenu.

Résultats des tests

- **Consommation passive** → On remarque clairement que cette manière de se servir de l'IA est terriblement inefficace; cette dernière fait des erreurs de compréhension et doit compléter des pans importants de la consigne qui ont été omis dans le contexte. Pour autant, on peut voir la puissance de l'IA pour générer du code qui fonctionne, même si ce dernier ne remplit pas les contraintes du sujet. On verra aussi plus tard que certaines erreurs que fait ChatGPT (Copilot moins) se poursuivent dans les tests suivants malgré un meilleur prompting.
- **Création de contenu** → Les deux IAs utilisées ont généré une structure de départ plutôt bonne et facilement complétable/compréhensible par l'humain derrière, similaire à un raffinage. Pour autant, Copilot avec Claude Sonnet continue de sur-générer le code et en fait plus que nécessaire. Cela s'avèrerait utile pour la co-création nonobstant.
- **Tentative de co-création** → Le constat fait est que la co-création est possible avec les LLMs sur ce genre de sujets simples, avec un code produit de qualité, respectant les contraintes et attentes du sujet. Cependant, cela demande à l'utilisateur de savoir intéragir intelligemment avec l'IA: Il doit savoir prompter efficacement avec du role prompting, fournir un contexte détaillé avec les attentes et contraintes exactes du cahier des charges, et être capable de corriger les erreurs de l'IA, ou de rebondir sur ses propositions pour avancer de son côte. Cela implique donc que l'utilisateur ait au moins les bases du langage de programmation utilisé, et surtout une compréhension précise du sujet. Il doit pouvoir communiquer efficacement ce qu'il attend de l'IA, et doit pouvoir interagir avec l'IA comme un assistant à qui il montre ses avancées et demande une suggestion ou une correction. Dans le cas du jeu des allumettes, l'IA– en particulier ChatGPT– a beau avoir produit un code respectable, les biais d'apprentissage produisent des erreurs de compréhension de consigne, et les lacunes en bonnes pratiques de programmation rendent la correction obligatoire. Aussi, il semblerait aussi que Github Copilot (ou Claude Sonnet par extension) soit plus efficace pour la co-création que ChatGPT si on s'inspire des résultats précédents, malgré sa tendance à en faire trop et à donc laisser moins de marge de réflexion personnelle à l'utilisateur.

Conclusion - Jeu des allumettes

Ce qu'on peut conclure de cette étude de cas est que dans le contexte de l'apprentissage de la programmation, les LLMs sont capables de **produire du code de qualité**, mais nécessitent activement que **l'utilisateur interagisse avec pour corriger ou rebondir sur leurs propositions**. Pour que l'apprentissage et la co-création soient efficaces, il faut donc que l'utilisateur ait une compréhension précise du sujet et une certaine expertise dans le langage utilisé. Pour autant, il semblerait que l'IA puisse parfois **faire beaucoup trop et mâche le travail de l'utilisateur**, ce qui peut être *contre-productif* si ce dernier tente de s'exercer ou de se former. Et à cela s'ajoute aussi les IAs qui font des **erreurs** ou qui **hallucinent** dans leur réponse, comme on a pu observer ChatGPT le faire dans tous les tests sans exception. En clair, le mieux serait pour les étudiants de d'abord commencer leur raffinage

et d'ensuite s'aider de l'IA pour convertir leur travail. Ils pourraient ensuite travailler en tandem pour améliorer le produit et corriger les éventuels problèmes.

Ingénierie Dirigée par les Modèles - Filière Réseau de l'N7

L'***Ingénierie Dirigée par les Modèles*** est une matière relativement complexe enseignée à l'ENSEEIHT. L'itération choisie est celle pour la filière **Réseau** de l'école, qui n'est donc pas forcément composée d'étudiants experts en informatique. Ainsi, on peut observer l'utilité de l'IA dans l'apprentissage et l'application de langages de modélisation, pour des étudiants dont le domaine d'expertise visé n'est pas la programmation ou le logiciel. Cette section part directement du postulat qu'on veut la co-création et utilise donc les points mentionnés dans la section correspondante.

Pour plus de détails, voir ici.

Zones de test

En tant qu'élève peu expérimenté en IDM et dont mes connaissances se limitent à de l'apprentissage autonome, j'ai utilisé l'IA pour:

- **Expliquer les graphes de marquage** → Elle a servi à m'expliquer les graphes de marquage du cas d'interblocage sur Tina. Elle a su très bien expliquer tous les éléments pour quelqu'un comme moi qui s'y connaît peu: Elle détaille les états, comprend correctement les transitions et explique pertinemment d'où vient l'interblocage. Cette utilisation pourrait être utile pour comprendre un concept du cours et la syntaxe de modélisation associée. Cependant, l'IA a clairement mâché le travail pour moi, et il m'est difficile de dire si j'aurais pu comprendre le graphe sans son aide. De même, le manque de réflexion de mon côté interroge sur la permanence des informations tout juste acquises dans ma mémoire (sans l'effort personnel, compliqué de mémoriser). Il est donc difficile de dire si l'IA est utile pour apprendre ou si elle est contre-productive dans ce cas.
- **Apprendre la syntaxe OCL et comprendre le métamodèle de SimplePDL en OCL** → J'ai utilisé l'IA pour m'aider à comprendre la syntaxe OCL et le métamodèle de SimplePDL. Elle a pu m'expliquer les concepts de manière claire et concise, et la réponse est clairement réutilisable pour réviser et renforcer ma compréhension de la syntaxe OCL. Cependant, la difficulté de l'IA à comprendre certains éléments du métamodèle montre qu'elle n'est pas infaillible et qu'il est donc nécessaire de comprendre le sujet modélisé. Ainsi, c'est définitivement une bonne manière de réviser la syntaxe OCL, mais il est impossible d'apprendre avec l'IA seule et sans avoir au préalable une compréhension du sujet.
- **Écrire des contraintes SimplePDL en OCL** → J'ai demandé à l'IA de m'aider à ajouter certaines contraintes au métamodèle de SimplePDL. En donnant le fichier pour contexte, il est capable de proposer des contraintes cohérentes avec le métamodèle. L'IA comprend bien la logique des dépendances entre activités et peut suggérer des invariants pertinents, bien qu'il faille parfois corriger la syntaxe OCL spécifique ou la signification de certains contextes. Cela montre que l'IA peut être utile pour écrire du code OCL, mais qu'elle nécessite une compréhension préalable du métamodèle et de la syntaxe OCL pour être efficace. En somme la conclusion est la même: l'IA semble être un bon outil pour aider à écrire des métamodèles mais nécessite d'être capable de comprendre intégralement le sujet sans quoi elle fait le gros du travail, incluant des erreurs sur le chemin.

Résultats et conclusion - IDM

À ce stade de l'étude de cas, on peut dire dans l'ensemble que l'IA a actuellement les capacités pour aider à comprendre les principes et les différentes syntaxes liés à la modélisation de métamodèles, mais qu'elle a encore des difficultés évidentes à certains éléments du sujet et écrit des bêtises. Cela est symptomatique du domaine de l'IDM qui est un peu plus niché et complexe, et qui est donc moins bien compris par les LLMs. À noter que l'IA mâche clairement le travail pour certaines tâches, et qu'il est donc nécessaire de connaître le sujet au préalable pour bénéficier correctement de son aide.

Suite

Pour voir la pertinence de l'utilisation de l'IA en IDM, il reste trois tests à essayer de réaliser avant la fin du stage:

- TP sur XTest
- TP sur Sirius

- TP sur Acceleo

Ces tests donneront sûrement une idée plus précise de la place que pourrait prendre l'IA dans l'apprentissage de sujets aussi complexes que l'IDM, et de la pertinence de son utilisation pour les étudiants.

Préparation d'un notebook pour une possible formation

Un petit *Jupyter Notebook* a été préparé pour servir de fondation à une potentielle formation. J'y ai mis une introduction aux LLMs avec des illustrations en Python, ainsi que des explications sur les biais et les limites de l'IA. Il est aussi possible d'y ajouter des exercices interactifs pour les étudiants, comme des questions à choix multiples ou des exercices de code à compléter. Une partie sur la pondération des *tokens* par les modèles à l'aide d'*interpreto* n'a pas pu être débutée, mais il serait d'intéressant de considérer cette section pour montrer aux étudiants comment les LLMs fonctionnent en interne, et l'importance de leur choix de prompts.

Comparaison des modèles selon les cas d'utilisation

J'ai synthetisé sur ce tableau les tâches récurrentes de développement et les modèles disponibles sur Github Copilot les plus appropriés. Pour cela j'ai utilisé cette page de documentation rédigée par Github et cet article du blog Github.

Conclusions et pistes d'exploration

À ce stade de la mission, l'avancée est comme suit:

- ☒ Étude pratique sur la pertinence de l'utilisation de l'IA dans des cas d'utilisation spécifiques
  - ☒ Cas simple avec le jeu des allumettes
  - ☒ Première évaluation d'un cas plus complexe avec l'IDM
- [ ]Finalisation de l'étude de cas IDM
  - ☐ Tests sur les TPs restants
  - ☐ Retour sur la première conclusion du rapport
- ☐ Préparation d'un notebook pour une formation
  - ☒ Introduction aux LLMs
  - – Discussion sur les biais inhérents aux LLMs
  - ☒ Comparaison des modèles de langage selon les tâches
  - ☐ Pondération des tokens avec interpreto

Ca a été une mission plutôt enrichissante a effectuer avec beaucoup de connaissances sur l'IA Générative acquises ou renforcées. Ca a aussi été une bonne introduction à la recherche. Sur une note plus personnelle, j'ai pu prendre une certaine avance sur la matière IDM, ce qui me donnera une petite longueur d'avance pour la 2ème année. Mais j'ai aussi ressenti durant le stage que c'est un travail qui est difficile à réaliser seul, notamment parce qu'avoir d'autres esprits sur le même sujet auraient permis d'étayer les réflexions concernant l'étude de cas, et aussi car en ce qui concerne la résolution d'un blocage, c'est souvent compliqué de trouver quelle tâche poursuivre quand autant semble être à faire en même temps.

Enfin, la conclusion actuelle vis-à-vis de l'utilisation de l'IA dans l'apprentissage est que les LLMs peuvent être un outil très puissant pour l'étudiant notamment dans la production d'un projet relativement simple avec des consignes claires qu'il comprend et connaît parfaitement. Cependant, pour une tâche nouvelle et complexe, l'apport positif de l'IA sur l'enrichissement individuel de l'étudiant est plus discutable. En effet, l'IA peut aider à la compréhension spontanée d'un concept ou d'une syntaxe ainsi que poser des bases de code exploitables, mais combiné à son potentiel d'erreurs, sa tendance à parfois mâcher le travail de l'utilisateur et son incapacité à comprendre des sujets trop complexes font pencher la balance vers l'idée qu'il vaut mieux déjà prendre en main le gros du sujet seul avant de se tourner vers l'IA pour la co-cognition.

En clair, il a été évident pour chaque test que sans connaissance préalable du sujet, l'IA peut même s'avérer contre-productive.