IA et développement logiciel : où va-t-on ?

Bruno Mermet
Université Le Havre Normandie

La percée des intelligences artificielles génératives (IAG) reposant sur des modèles de langage, depuis la sortie de ChatGPT 3.5 à l'automne 2022, a radicalement transformé bon nombre de métiers. Le développement informatique n'a pas échappé à la chose. Les développeurs n'ont pas été les plus rapides à s'emparer de l'outil, même s'ils étaient, a priori, bien plus au courant que les autres secteurs du monde du travail des limites des outils ; mais les hallucinations qu'ont pu produire les premières versions de ChatGPT faisaient alors fureur sur les réseaux.
Cependant, entre fin 2022 et début 2024, plusieurs versions de ChatGPT sont parues, et des concurrents sérieux ont commencé à voir le jour. Ainsi, la qualité des réponses fournies par les IAG s'est nettement améliorée et dans le même temps, la proportion de réponses réellement « fausses » a considérablement diminué. Par ailleurs, tout un chacun a pu tester l'outil et a appris à mieux maîtriser le fameux « prompt ». L'année 2024 a ainsi été une véritable année charnière dans la relation entre le secteur du développement logiciel et les IAG. Mais pour quel bénéfice ?

Petits rappels autour des IAG

Le but de cette partie n'est pas de rappeler tous les aspects théoriques et techniques derrière les IAG reposant sur des grands modèles de langages (les seules considérées ici), mais de rappeler les quelques éléments nécessaires pour comprendre la suite.

Une IAG est un programme qui, à partir d'un modèle appris sur des données, génère une suite **probable** à un texte appelé *prompt*. Il est à noter que le processus est en fait itératif : à partir du prompt, l'IAG ne génère que le « mot »

DOI:10.48556/SIF.1024.25.189

suivant. Le mot généré est alors rajouté au prompt précédent, et on réitère le processus avec ce prompt prolongé. Suivant les cas, le prompt initial peut être :

1. donné entièrement par l'utilisateur ;
2. constitué d'un préambule interne au logiciel puis du texte saisi par l'utilisateur ;
3. rempli automatiquement dans un contexte donné.

La plupart des outils interactifs comme ChatGPT rentrent dans le deuxième cas. Un préambule (non communiqué) est ajouté afin d'obtenir des réponses plus pertinentes. Les assistants de code comme Github copilot relèvent du troisième cas : en général l'utilisateur n'interroge pas explicitement l'outil ; un prompt est régulièrement créé à partir du code en cours de développement et est envoyé à l'assistant, qui propose alors une complétion probable du code.

Utilisation des IAG par les équipes de développement

Dans le cadre du développement, différents usages des IAG sont mis en œuvre, parmi lesquels :
– le développement du code ;
– la synthèse de documentation ;
– l'aide à l'analyse d'un code existant ;
– l'optimisation de code ;
– l'écriture de la documentation ;
– le développement des tests.
Dans le cas du développement de code (que ce soit le code de l'application ou le code de test), deux façons de procéder sont souvent envisagées :
– l'utilisation d'un assistant comme Github copilot : l'outil, intégré dans un IDE, propose du code à la volée. Le prompt est constitué d'une partie du code du projet, commentaires inclus. Il est alors possible d'obtenir du code directement à partir d'un commentaire écrit en langue naturelle (la dernière tendance à la mode, appelée *vibe coding*) ;
– l'utilisation séparée d'une IAG : la développeuse demande directement à une IAG, via un prompt dédié, de lui générer du code répondant à son problème.
Dans le premier cas, l'IAG apparaît comme un outil de complétion de code amélioré. Mais là où les outils classiques de complétion de code se limitent à ne proposer que des complétions courtes lorsque la liste de complétions possibles est fortement limitée, les IAG vont bien au-delà, en proposant des complétions longues (des blocs d'instructions ou des sous-programmes complets) dans des contextes très ouverts. La confusion entre les outils, tout comme la diminution des cas d'hallucination flagrante de la part des outils d'IAG a soudainement amené les équipes de développement à adopter massivement ces outils au cours de l'année 2024, comme le montre [2].

Dans le second cas, on obtient souvent très rapidement un programme semblant répondre au problème posé. Les contextes pris en compte par les IAG ayant par ailleurs beaucoup augmenté, la taille des problèmes pouvant être traités a également grandement augmenté.

Conséquences

Un code globalement moins fiable

Un premier travers dans l'utilisation des IAG pour le développement est dû au biais de confirmation : face à un code proposé commençant plutôt bien, la tendance est rapidement de considérer la suite comme également valide. Cela est d'autant plus vrai dans un monde ayant systématiquement tendance à mettre en avant la « performance ». D'ailleurs, dans [1], il ressort qu'en 2024, seulement 39 % des développeurs faisaient peu ou pas confiance au code généré par une IAG. Une des conséquences évoquées dans ce même article est qu'une augmentation de 25 % de l'utilisation des IAG pourrait mener à une diminution de 7,2 % de la stabilité du code.

Un code de moins bonne qualité

L'étude présentée dans [2] montre un net changement en 2024 dans la structure du code généré. Ce changement montre une nette diminution du *refactoring* ainsi qu'une augmentation de la duplication de code. Or, plus la qualité d'un code baisse, plus sa validation et son maintien sont difficiles. Par ailleurs, les risques liés à la duplication de code sont maintenant bien connus : clonage de bugs, corrections partielles, etc.

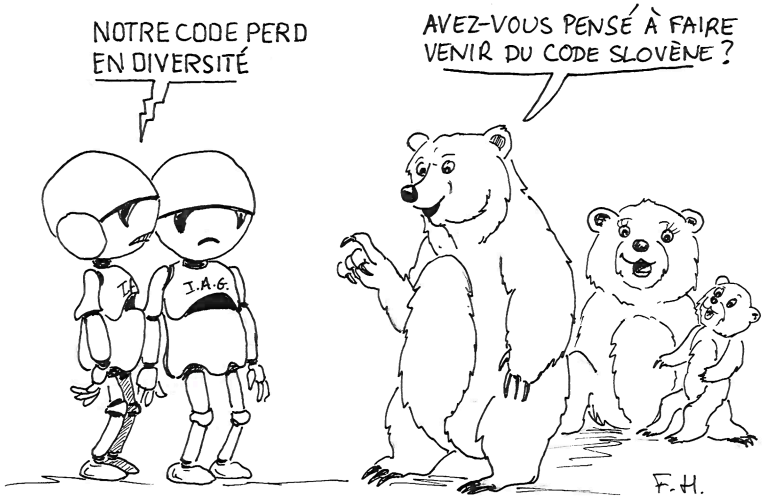
Perte potentielle de compétence

Utiliser une IAG dans le cadre du développement peut s'apparenter à un développement en binôme, où l'IAG joue le rôle de pilote et le développeur le rôle de copilote. Dans un développement en binôme classique, le copilote est actif dans toute la phase de développement, et cela est induit par le côté « humain » du pilote. Mais lorsque le pilote est réduit à un programme, l'interaction devient vite unidirectionnelle, et le copilote finit par devenir passif, se contentant la plupart du temps de valider ce que propose l'IAG. Par manque de pratique, il finit par perdre en compétence.

Ceci n'est bien sûr pas une fatalité : tout dépend de la façon dont on utilise les IAG. Cherche-t-on à analyser les solutions proposées ? À avoir un regard critique, voire à apprendre de ce qu'elles nous proposent ? Là encore, la course à la performance tend malheureusement à encourager le comportement décrit dans le paragraphe précédent.

Effondrement possibles des IAG et perte de diversité du code

Les IAG sont entraînées à partir des données trouvées sur Internet. Mais comme le montrent les études déjà citées, le code présent sur Internet est de plus en plus généré par des IAG. Or plusieurs études (comme [3]) tendent à montrer que les IAG s'écroulent lorsqu'elles apprennent de leurs propres résultats. Par ailleurs, dans les sciences du vivant, il a été montré dans de nombreuses circonstances, qu'un être vivant — ou son patrimoine génétique — pour être robuste, doit être diversifié [4]. Or la multiplication de code produit par IA mène à une uniformité du code. En perdant ainsi en diversité dans la mémoire collective que constituent les entrepôts de code, on risque aussi de perdre en robustesse : même avec des recherches Web « classiques », la recherche de solutions par des équipes de développement fournira des résultats similaires, diminuant ainsi les chances de produire de meilleures solutions par hybridation de solutions existantes.



Diffusion de code potentiellement confidentiel

L'utilisation d'assistants de code intelligents envoie de manière quasi continue le code du projet à un serveur distant et ce, de façon totalement transparente. Actuellement, le risque de violation de la confidentialité est sous-évalué par les équipes de développement. On peut alors être confronté à deux risques antagonistes :

– une équipe de développement ne se rend pas compte du problème ; elle diffuse alors du code et des données confidentielles (des codes d'accès privés, fort judicieusement non stockés sur des dépôts comme Github, peuvent malgré tout être diffusés ainsi) ;
– une équipe de développement, consciente du problème, bloque l'utilisation de ces outils ; elle engage une nouvelle personne, formée à l'utilisation des IAG (mais cette personne est alors totalement inopérante dans l'équipe).

Bien sûr, il est possible d'utiliser des modèles *open source* fonctionnant en interne pour éviter ces problèmes. Mais cela nécessite des ressources de calcul dédiées, et amène à utiliser des modèles souvent jugés moins performants car moins à jour.

Aucune garantie sur la base d'apprentissage des IAG

Les IAG apprennent en général à partir d'informations trouvées sur le Web, sans sélectivité. Elles peuvent donc apprendre, y compris à partir de code erroné créé intentionnellement pour dévoyer leur fonctionnement. C'est, par exemple, ce que l'on a constaté récemment sur des sites russes visiblement nourris de fausses informations pour dévoyer les réponses des IAG aux questions portant sur la guerre en Ukraine.

Plus affolant encore, le *fine tuning* des IAG, qui devrait permettre de mieux les calibrer, semble démontrer des risques de dégradation complète que l'on ne sait pas encore expliquer. Ainsi, dans [9], les auteurs montrent qu'une tentative de dévoyer une IA via du *fine tuning* ciblé sur du code Python vulnérable pouvait amener une IAG pourtant récente (GPT-4o, en l'occurrence) à diverger totalement, y compris sur des domaines sans aucun rapport avec la question posée ? (incitation au suicide, promotion d'Hitler, etc.).

IA et test : le grand danger

Si, pendant longtemps, le test logiciel a été très empirique, les choses ont maintenant bien changé. C'est un domaine de l'informatique à part entière permettant d'apporter de réelles garanties sur le bon fonctionnement d'un logiciel. Et même si l'activité de test devrait être considérée comme une activité particulièrement riche intellectuellement [5], elle reste souvent vue comme une perte de temps par les employeurs et comme une écriture pénible quasi automatique de code par les équipes de développement.

Aussi, de nombreux travaux proposent d'utiliser les IAG pour écrire ces fameux cas de test [6]. C'est malheureusement le contraire de ce qui devrait être fait. En effet, comme cela a déjà été rappelé, les IAG ne génèrent que du « probable », et n'ont aucune notion du « correct ». Ainsi, une IAG, pourra générer un cas de test à partir d'un simple commentaire de code — potentiellement erroné — ou du code déjà écrit, générant alors un cas de test biaisé.

J'ai notamment pu constater que des responsables d'équipes de développement pensaient qu'une simple mesure de couverture de code suffisait à évaluer la production de cas de tests par une IAG !

Si, à terme, le développement devait utiliser massivement les IAG, l'écriture des tests devrait au contraire être le dernier endroit où les utiliser.

Aspects écologiques

Les IA reposant sur l'apprentissage sont, de manière générale, des systèmes assez gourmands en énergie. Mais cela est pire pour les IAG, pour lesquelles non seulement le processus d'apprentissage est ultra-gourmand (en énergie, comme en matières premières), mais le processus d'utilisation est également très gourmand. Les chiffres sur le sujet restent en général flous : entre les aspects confidentiels et les différents outils disponibles, il n'est guère possible d'avoir une estimation précise. Par ailleurs, suivant les pays, l'électricité nécessaire au fonctionnement de ces outils est produite de manière plus ou moins renouvelable. Mais, même si une requête vers ces outils ne consommait pas plus qu'une simple requête dans un moteur de recherche, il faut prendre en compte la fréquence d'utilisation : en utilisant des IAG comme « outils de complétion de code améliorés », on peut considérer que chaque période d'inactivité d'au moins une seconde d'un développeur va générer une requête. Ce sont donc des centaines de requêtes quotidiennes supplémentaires par développeur qui vont finalement être générées. Les énergies renouvelables ne produisent actuellement dans le monde qu'environ 30 % de l'électricité. En augmentant ainsi la consommation électrique, une partie non négligeable des nouveaux sites de production d'énergie renouvelable ne sera pas consacrée à diminuer la production de gaz à effet de serre et la consommation de ressources fossiles, mais sera consacrée à absorber cette augmentation de la demande. Ce problème n'est cependant pas propre au développement informatique, donc je n'en parlerai pas davantage, même si c'est un problème fondamental, qui doit être pris en considération dans l'éventuelle utilisation de l'IAG comme aide au développement de logiciels.

Formation

La percée des IAG influe également sur la façon dont nous devons mener nos formations au développement.

En effet, il est clair maintenant que la quasi-totalité des exercices de programmation « de base » peuvent être résolus par une IAG. Faut-il alors sauter cette étape ? Certainement pas : les concepts de base doivent être parfaitement acquis. Ce n'est pas parce que les calculatrices existent qu'on n'apprend plus les bases du calcul à l'école. Bien sûr, avant l'avènement des

IAG, on trouvait déjà beaucoup d'exercices corrigés sur Internet. Mais il y avait toujours un travail minimum d'adaptation à faire, difficile pour un élève n'ayant pas les bases (j'ai déjà vu un élève rendre du PL/SQL alors que seul PL/pgSQL avait été présenté et pratiqué pendant toute l'année). Mais maintenant, les IAG font ce travail en totale autonomie. Il est important d'en avoir conscience et d'évaluer les élèves dans des contextes où les IAG ne sont pas utilisables (examen sur papier, mise en place d'un *firewall*, etc.).

Par ailleurs, il faut également avoir conscience que les élèves utiliseront de toutes façons les IAG. Il n'est donc plus possible de les ignorer. Il devient nécessaire de former les élèves à leur utilisation : détailler les risques, la place qu'elles peuvent prendre dans leurs études, et montrer leurs limites. Ainsi, en licence 3, je pratique beaucoup de *live coding*, c'est-à-dire que je développe en direct du code devant les étudiants, que ce soit pour illustrer directement un point de cours ou pour répondre à une question d'un étudiant. J'en profite alors pour montrer et commenter les suggestions de Github copilot : si certaines de ses propositions sont assez bluffantes, je jette ou modifie à peu près 80 % de ce qu'il me propose.

Il semble toutefois que les IAG vont de plus en plus réussir à produire des « unités de code » correctes : fonctions relativement standard, classes relativement simples. Mais elles risquent d'avoir du mal, à terme, à produire des applications de taille « raisonnable ». Le travail de développement que peut faire un technicien risque ainsi, à terme, de pouvoir être en partie effectué par une IAG. Mais ce ne sera probablement pas le cas de tâches que l'on demande à un bac+5. Il est donc impératif de proposer un maximum de passerelles permettant d'amener tout développeur qui le peut à un niveau bac+5. Le développement de la spécialité NSI (incluant d'en faire un prérequis pour les formations en informatique) pourrait permettre de faciliter grandement les choses.

La promotion actuellement faite par certaines personnes du *vibe coding* est un risque important à prendre en compte : de plus en plus de développeurs en font la promotion, et ils trouvent un certain écho y compris sur la place publique : certains annoncent même la fin du métier de développeur. Pourtant les défauts du code produit par les IA sont bien là, et il est particulièrement important d'insister auprès des étudiants sur tout le contrôle qu'il faut avoir sur du code produit par IA.

En attendant, dans le cadre de mes enseignements, je présente deux grands types d'information à mes étudiants, détaillés dans les deux sections suivantes.

Dois-je faire appel à une IAG ?

En conséquence de tout ce qui a été dit précédemment sur le fonctionnement des IAG, on doit se poser la question de savoir si, oui ou non, on doit ou on peut faire appel à une IAG pour effectuer une tâche particulière.

Ai-je vraiment besoin d'un outil pour effectuer cette tâche ?
Si non, oublier les IAG ; les IAG sont des outils grands consommateurs de ressources.

Existe-t-il un outil dédié autre qu'une IAG pour résoudre le problème ?
Si oui, oublier les IAG ; les résultats des IAG sont souvent approximatifs, et coûteux à produire ; un outil dédié produira en général un résultat meilleur et à moindre coût.

Suis-je capable d'évaluer ce que produira une IAG sur ce problème ?
Si non, oublier les IAG ; une IAG produisant du probable, et non du vrai, il faut être capable d'évaluer la production d'une IAG pour savoir si on peut l'utiliser.

Vais-je prendre le temps de valider la réponse que produira l'IAG ?
Si non, oublier les IAG ; là encore, une IAG pouvant produire du faux, il faut systématiquement prendre le temps d'évaluer le résultat fourni.

Ce qu'il faut savoir sur les IAG

Si on ne veut pas se contenter de suivre la mode sans réfléchir, il est important d'en savoir un peu plus sur les tenants et aboutissants des IAG.

QUALITÉ DU RÉSULTAT PRODUIT :

– les IAG produisent du « probable », pas du « vrai » ;
– les IAG n'ont aucune notion de vrai et de faux.

ASPECTS SOCIAUX ET ENVIRONNEMENTAUX :

– les IAG demandent beaucoup d'énergie ;
– les IAG demandent beaucoup de matériel, et consomment ainsi beaucoup de ressources ;
– les IAG ont tendance à reproduire les biais, notamment sociétaux, présents dans les données ayant servi à leur apprentissage ;
– les résultats bruts des IAG sont corrigés par des personnes sous-payées dans les pays émergents [7].

AVENIR DES IAG :

– de plus en plus de contenu du Web est produit par des IAG ;
– les IAG apprennent essentiellement à partir du contenu trouvé sur le Web ;
– les études montrent que les résultats des IAG se dégradent lorsqu'elles apprennent de leurs propres résultats
→ en conséquence, à terme, les résultats des IAG risquent de se dégrader ;

– les études semblent montrer que l'augmentation des capacités des IAG tend vers un « plafond de verre », malgré l'augmentation de la puissance de calcul des ordinateurs
  → donc, il est peu probable que les capacités des IAG continuent de progresser encore longtemps à ce rythme.

En guise de conclusion

La fin du métier de développeur n'est pas pour demain. Les intelligences artificielles génératives peuvent parfois apporter un gain de temps appréciable, mais elles doivent être utilisées avec rigueur et parcimonie, en pleine conscience de leurs nombreuses limites.

Références bibliographiques

[1] DORA (google), *Accelerate State of DevOps*, 2024. https://cloud.google.com/devops/state-of-devops.

[2] Gitclear, AI Copilot Code Quality : valuating 2024's Increased Defect Rate via Code Quality Metrics, 2025. https://gitclear-public.s3.us-west-2.amazonaws.com/AI-Copilot-Code-Quality-2025.pdf.

[3] Shumailov, I., Shumaylov, Z., Zhao, Y. et al. *AI models collapse when trained on recursively generated data*. Nature 631, 755–759 (2024).

[4] Olivier Hamant, *Antidote au culte de la performance. La robustesse du vivant*, Tract Gallimard n°50, 2023.

[5] Glenford J. Myers *et al.*, The art of software testing, John Wiley & Sons, Inc, 2012.

[6] Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. IEEE Transactions on Software Engineering.

[7] Perrigo B., Exclusive: OpenAI Used Kenyan Workers on Less Than $2 Per Hour to Make ChatGPT Less Toxic, Time Magazine, 18/01/2023. https://time.com/6247678/openai-chatgpt-kenya-workers/.

[8] Portal Kombat : quand la désinformation russe "biberonne" ChatGPT et autres IA, France 24. https://www.france24.com/fr/%C3%A9co-tech/20250309-portal-kombat-pravda-d%C3%A9sinformation-russie-chatgpt-intelligence-artificielle.

[9] Jan Betley, Daniel Chee Hian Tan, Niels Warncke, Anna Sztyber-Betley, Xuchan Bao, Martín Soto, Nathan Labenz, Owain Evans , Emergent Misalignment: Narrow fine-tuning can produce broadly misaligned LLMs, ICLR 2025 FM-Wild Workshop, 2025.