Se connecter
Proposer un contenu
Identifiant
Mot de passe
Connexion automatique
Se connecter
Pas de compte ? S'inscrire...

12 nov. 2026 13

Journal : Par la loi du bobsleigh de Lemmer-Webber, le code généré par LLM n'est à terme pas libre

Posté par **Samuel** (**site web personnel**) le 12 novembre 2026 à 21:49. **Licence CC By-SA.**
Étiquettes : **logiciel_libre**, **grands_modèles_de_langage**, **fsf**

Désolé, ce journal parle encore LLM (grands modèles de langues).

Il y a en ce moment un débat féroce entre des développeurs, militants, organisations qui s'opposent fermement à l'usage des LLMs pour des raisons éthiques et certains développeurs centraux et des organisations historiques de l'écosystème du logiciel libre qui utilisent ou autorisent l'utilisation de LLM pour l'écriture de code. Le débat porte sur la pertinence ou non d'utiliser ce code.

Je veux ici apporter une contribution à ce débat en argumentant que le logiciel généré par LLM ne peut pas être libre en raison de la liberté n°1 du logiciel libre, celle d'étudier et de modifier le code source.

Je ne reviens pas sur les **nombreuses autres** raisons de ne refuser en bloc les LLMs, raisons que je considère bien plus importantes que celle que j'ajoute ici. Je ne reviens pas non plus sur le débat juridique autour de ce qu'on peut faire légalement d'un code généré à partir de nombreux exemples de code sous des licences très variées. Je me place du point de vue de l'ingénieur pragmatique libriste, celui capable d'ignorer les humains qui ne sont pas lui-même et d'aborder une question sous le seul angle des objectifs qu'il s'est fixé tant qu'on respecte les principes du logiciel libre.

La loi du bobsleigh observée par Christine Lemmer-Webber dans son texte Faulty Towers, vibe sickness, and the vibe bobsled est que : quelles que soit les limites posées aux générateurs automatiques initialement par le développeur, quand on met un doigt dans la machine on finit toujours et assez rapidement par abandonner le contrôle de l'écriture du code à ces générateurs automatiques, puis, le code s'empilant, à reposer sur ces outils pour pouvoir comprendre l'architecture et le comportement du code. C'est comme une piste de bobsleigh sur laquelle on s'engage : quel que soit l'endroit où on rejoint la piste, à partir du moment où on embarque dans l'appareil, on dévale la pente et on on finit toujours au même endroit à la fin.

La FSF (fondation pour le logiciel libre) définit le logiciel libre à partir de 4 libertés dans le texte Qu'est-ce que le logiciel libre ? :

1. la liberté de faire fonctionner le programme comme vous voulez, pour n'importe quel usage (liberté 0) ;
2. la liberté d'étudier le fonctionnement du programme, et de le modifier pour qu'il effectue vos tâches informatiques comme vous le souhaitez (liberté 1) ; l'accès au code source est une condition nécessaire ;
3. la liberté de redistribuer des copies, donc d'aider les autres (liberté 2) ;
4. la liberté de distribuer aux autres des copies de vos versions modifiées (liberté 3) ;

Si on commence à intégrer dans un logiciel libre du code généré par des machines, par la loi du bobsleigh, le code devient à terme "illisible" par lui-même : on a besoin de faire appel à un outil tiers pour tenter de lui donner du sens. Il n'est plus possible de le comprendre et de le modifier. La FSF précise, à propos de la liberté numéro 1 :

> Du code source rendu illisible [obfuscated] n'est pas du vrai code source et ne compte pas comme code source.

Les LLMs ont cet effet, involontaire, de rendre à terme le code illisible. Le code ainsi généré n'est pas libre, à terme.

Cette liberté de lire et modifier un programme était déjà fragilisée avant l'arrivée des LLMs remarque Cassandrich sur le fedivers : le fait de devoir toujours mettre à jour un logiciel, pour des raisons de fonctionnalités ou de sécurité, nous rend dépendants des mainteneurs et rend en pratique très coûteux de le détourner selon nos besoins.

Cette liberté de transformer comme on souhaite le logiciel est essentielle pour le logiciel libre. C'est la mise en forme des principes hacker (la liberté du détournement) et ce qui oriente l'informatique dans un sens plus convivial, au sens d'Ivan Illich et présenté par Benjamin Grassineau aux JdLL 2024 (vidéo, transcription).

Du coup, je me demande dans quelle mesure la loi du Bobsleigh de Lemmer-Webber est vérifiée. A-t'on des contre-exemples ? Avez-vous observé des situations où, après avoir commencé à utiliser des LLMs dans la gestion du code jusqu'à un certain point, les développeurs sont revenus sur leur décision et ont réduit ou banni l'utilisation de ces outils ?

(50 commentaires). Markdown EPUB

Loi du bobsleigh

Posté par **Thomas** (**site web personnel**) le 12 novembre 2026 à 22:49. Évalué à 4 (+5/-2).

> J'adore.
> On a tous ressenti ce moment où on ne lit plus le code. On sait qu'on devrait, qu'on aurait du mais pfff …
>
> Un bobsleigh finit par freiner, cependant.

Répondre

[^] # Re: Loi du bobsleigh

Posté par **piratebab2** le 12 novembre 2026 à 23:07. Évalué à 7 (+6/-0).

Si certains sont revenus en arrière, ils ne communiquent pas dessus. Probablement parce qu'ils ont peur que cela soit considéré comme un échec.
On voit au contraire beaucoup de témoignages de personnes qui étaient réticentes au début, on fini par s'y mettre et ne veulent pas revenir en arrière.
Ce biais va fausser les statistiques.

Perso je serais bien incapable aujourd'hui de lire un code machine comme je le faisais il y a 40 ans, un logiciel qui le met en ASM me serait indispensable. Et encore, même l'ASM me serait bien complexe à comprendre aujourd'hui. Se faire aider d'un logiciel pour comprendre ce que fait un autre logiciel est une pratique ancienne.

Répondre

[^] # Re: Loi du bobsleigh

Posté par **Pat _** le 13 novembre 2026 à 08:32. Évalué à 6 (+6/-1). Dernière modification le 13 novembre 2026 à 08:32.

Du code machine, c'est (sensé être) la traduction reproductible d'un code source de haut niveau que tu peux lire et comprendre. Même si tu ne le lis pas dans le texte directement, si tu as le source tu sais ce qu'il fait.

Avec le code pondu par les LLM, il vient toujours un moment où on ne cherche même plus à comprendre comment il fait ce qu'il (parait) faire…

Répondre

[^] # Re: Loi du bobsleigh

Posté par **lolop** (**site web personnel**) le 13 novembre 2026 à 08:47. Évalué à 3 (+1/-0).

> Du code machine, c'est (sensé être) la traduction reproductible d'un code source de haut niveau que tu peux lire et comprendre.

Pas nécessairement (même si très très majoritairement), tu peux avoir du code machine qui n'est que la traduction du code assembleur du processeur ; c'est symbolique, mais proche du code machine.

--
Votez les 30 juin et 7 juillet, en connaissance de cause. http://www.pointal.net/VotesDeputesRN

Répondre

[^] # Re: Loi du bobsleigh

Posté par **Valentin** le 13 novembre 2026 à 09:30. Évalué à 3 (+2/-0). Dernière modification le 13 novembre 2026 à 09:30.

> Perso je serais bien incapable aujourd'hui de lire un code machine comme je le faisais il y a 40 ans

C'est bien pour cela qu'il faut commenter correctement son code.

Car sinon il faut beaucoup moins de 40 ans pour ne plus comprendre de quoi il s'agit…

Répondre

[^] # Re: Loi du bobsleigh

Posté par **devnewton** (**site web personnel**) le 13 novembre 2026 à 11:04. Évalué à 8 (+7/-2).

Un vrai programmeur ne commente pas: son code est évident.

--
Ce post est offensant ? Prévenez moi sur https://linuxfr.org/board

Répondre

[^] # Re: Loi du bobsleigh

Posté par **Thomas** (**site web personnel**) le 13 novembre 2026 à 11:24. Évalué à 8 (+7/-0).

un vrai programmeur ne commente pas son code, il laisse les autres le commenter.

Répondre

[^] # Re: Loi du bobsleigh

Posté par **Colin Pitrat** (**site web personnel**) le 13 novembre 2026 à 13:55. Évalué à 10 (+12/-0).

Un vrai programmeur ne commente pas, sauf sur Linuxfr!

Répondre

[^] # Re: Loi du bobsleigh

Posté par **piratebab2** le 13 novembre 2026 à 11:39. Évalué à 5 (+4/-0).

A l'époque, on codait directement en code machine! Pas d'IDE et autre editeur de texte évolué pour mettre des commentaires. Un terminal VT100 avec 80 caractères verts.
En ASM c'était un peu mieux, mais on n'avait pas beaucoup de place pour les commentaires.Et je ne parle pas des JUMP pour rajouter un patch en fin de fichier.
Si il fallait le relire de nos jours, je ferai un petit script pour afficher tout ça de façon plus lisible.
Encore faut il trouver un lecteur pour les disquettes qui faisait la taille d'un plat a tarte …
Il est vrai qu'un se souciait peu de le licence dans les années 80. C'était avant l'arrivée de la GPL (1989).

Répondre

[^] # Re: Loi du bobsleigh

Posté par **Serge Julien** le 13 novembre 2026 à 14:09. Évalué à 5 (+3/-0).

(Un peu hors propos mais je ne résiste pas à l'envie)

> Pas d'IDE et autre editeur de texte évolué pour mettre des commentaires

En effet, c'est bien connu, les vrais programmeurs ne les utilisent pas : https://xkcd.com/378/

Tiens à propos, est-ce que quelqu'un a déjà formulé la loi suivante : "En informatique, quel que soit le sujet, il y a un XKCD qui l'illustre" ?

Répondre

[^] # Re: Loi du bobsleigh

Posté par **Faya** le 13 novembre 2026 à 14:45. Évalué à 5 (+3/-0).

> Tiens à propos, est-ce que quelqu'un a déjà formulé la loi suivante : "En informatique, quel que soit le sujet, il y a un XKCD qui l'illustre" ?

Je l'ai souvent lu et il y a carrément un subreddit qui recense toutes les fois où un xkcd a été pertinemment utilisé. Je viens de trouver un site qui dit avoir fait un MCP pour pouvoir brancher son agent IA sur xkcd et trouver automatiquement celui qui est pertinent. Bref, cette "loi" n'a pas encore reçu de numéro officiel comme la `rule 34` mais elle est assez bien établie.

Répondre

[^] # Re: Loi du bobsleigh

Posté par **BAud** (**site web personnel**) le 13 novembre 2026 à 14:58. Évalué à 5 (+3/-0). Dernière modification le 13 novembre 2026 à 14:58.

> Encore faut il trouver un lecteur pour les disquettes qui faisait la taille d'un plat a tarte …

alors que pour les mêmes dimensions et l'épaisseur d'une tarte, tu pouvais stocker sur bande de l'ordre de 33 Mo en 6250 BPI ᵂ !

Répondre

J'ai tout relu

Posté par **wilk** (**site web personnel**, **Mastodon**) le 12 novembre 2026 à 23:03. Évalué à 10 (+22/-1).

https://mamot.fr/@matiu_bidule/117602686038297685

> «J'ai relu tout ce que l'AI a produit»
> est le nouveau
> «J'ai lu et j'accepte les conditions d'utilisation»

Répondre

[^] # Re: J'ai tout relu

Posté par **abriotde** (**site web personnel**, **Mastodon**) le 14 novembre 2026 à 07:57. Évalué à 1 (+1/-2).

Mais tu peux tout relire avec un LLM 😅

Au delà de la blague, c'est vraiment pertinent, surtout si c'est un LLM différent. C'est un peu comme avoir une relecture par un pair.

Mais au fond, est-ce si grave de ne pas tout relire? On ne relit pas non plus le code asm généré par le compilo. Pire si je parle de l'ASM généré par un compilo à la volé de JavaScript ou Python. Par contre pour du débogage on peut le relire.
C'est comme les logs. Personne ne les lit si ce n' est des programmes ou des humains en cas de problème. Cela ne rends pas les logs inutiles.

--
Sous licence Creative common. Lisez, copiez, modifiez faites en ce que vous voulez.

Répondre

mon alternant et GPL

Posté par **jtremesay** (**site web personnel**) le 12 novembre 2026 à 23:13. Évalué à 10 (+11/-2).

> La FSF précise, à propos de la liberté numéro 1 :
>
> > Du code source rendu illisible [obfuscated] n'est pas du vrai code source et ne compte pas comme code source.

AJA j'ai appris que le code "très créatif" que produit mon alternant quand on ne le supervise pas assez ne peut pas être distribué sous licence GPL.

Répondre

[^] # Re: mon alternant et GPL

Posté par **Jona** le 13 novembre 2026 à 09:46. Évalué à 4 (+2/-0).

AJA Que la première fois que j'ai lu "AJA", il était utilisé "AJA j'ai appris" ! 🙃

Répondre

[^] # Re: mon alternant et GPL

Posté par **Benoît Sibaud** (**site web personnel**) le 13 novembre 2026 à 09:59. Évalué à 5 (+2/-0).

Pour *Au Jour d'Aujourd'hui j'ai appris de savoir présentement ce jour de maintenant.*

Répondre

[^] # Re: mon alternant et GPL

Posté par **Jona** le 13 novembre 2026 à 10:15. Évalué à 4 (+2/-0).

Il faut mettre à jour https://fr.wikipedia.org/wiki/Aja ᵂ alors ! :-P

Répondre

[^] # Re: mon alternant et GPL

Posté par **Liorel** le 13 novembre 2026 à 10:53. Évalué à 4 (+2/-0). Dernière modification le 13 novembre 2026 à 10:53.

Tiens, un mot de 3 lettres, prononçable, écrit sur DLFP, et qui n'est pas le nom d'un langage de programmation. Créons vite le langage de programmation !

--
Ça, ce sont les sources. Le mouton que tu veux est dedans.

Répondre

spec.md

Posté par **Nicolas Boulay** (**site web personnel**) le 13 novembre 2026 à 07:04. Évalué à -7 (+1/-11).

[^] # Re: spec.md

Posté par **wilk** (**site web personnel**, **Mastodon**) le 13 novembre 2026 à 08:24. Évalué à 7 (+6/-1).

https://www.gnu.org/philosophy/free-sw.fr.html#four-freedoms

> Prenez par exemple un programme A qui lance automatiquement un programme B afin de gérer certaines situations. Si nous souhaitons distribuer A tel quel, cela implique que les utilisateurs auront besoin de B ; il nous appartient donc de juger si A et B sont tous deux libres.

C'est comme de diffuser un programme dans un langage où on n'aurait pas le compilateur.
Inversement on pourrait dire qu'un programme binaire est libre et que le code c'est le langage machine.

Répondre

[^] # Re: spec.md

Posté par **Nicolas Boulay** (**site web personnel**) le 13 novembre 2026 à 09:39. Évalué à 2 (+2/-3).

Ton extrait correspond a un script qui lancerait un binaire.
C'est la fameuse vaccination de la GPL qui rend compliqué les drivers proprio dans Linux par exemple. En gpl, tu ne peux pas dépendre d un logiciel proprio à l'execution.

Pour la compilation, c'est different.

Faq sur les outils non libre
https://www.gnu.org/licenses/gpl-faq.en.html#NonFreeTools

--
"La première sécurité est la liberté"

Répondre

[^] # Re: spec.md

Posté par **wilk** (**site web personnel**, **Mastodon**) le 13 novembre 2026 à 10:00. Évalué à 3 (+2/-1).

Techniquement c'est différent mais par rapport à ce qui nous intéresse c'est exactement le même problème, on perd cette liberté d'étudier de fait de dépendre d'outils qui ne sont pas libres.
Un des signes c'est que la plupart des vibecodeurs vomissent en Rust alors que ça n'est pas le langage le plus propice à se laisser étudier.
Jusqu'à présent la plupart des projets libres s'interdisent de rendre le code trop compliqué pour pouvoir le maintenir le plus longtemps et par le plus de monde possible. De même pour des projets privés d'ailleurs.
On pourrait s'imaginer utiliser l'IA justement pour s'assurer qu'un code soit exempt de bugs et bien documenté mais ça n'assurerait qu'un bénéfice sur le long terme et ça donnerait plutôt plus de boulot à court terme (cf les rapports de bugs plus fréquents). De fait on voit avec les dégraissages que c'est tout l'inverse.

Répondre

[^] # Re: spec.md

Posté par **Nicolas Boulay** (**site web personnel**) le 13 novembre 2026 à 10:09. Évalué à 2 (+3/-4).

On parlait de droit et de licence, le problème est traité depuis 25 ans par la fsf avec les compilo non libre.

D'un point de vu technique, soit tu garde un code nickel et tu relis et corrige le code llm, ce qui est chiant et lourd ou tu te contente de ses revues. Pour un code ancien, c'est souvent le choix (cf gimp).

Ou bien tu passes par les .md comme clevercloud, et tu ne touches plus le code. Par contre, il faut bosser le processe pour que le llm te fournisse des preuves (d ou l usage de rust, des revue, d outils formel, de test enorme bout en bout,…)

--
"La première sécurité est la liberté"

Répondre

[^] # Re: spec.md

Posté par **wilk** (**site web personnel**, **Mastodon**) le 13 novembre 2026 à 11:00. Évalué à 1 (+1/-2).

Non, on parle de la liberté d'étudier.
" l'accès au code source est une condition nécessaire ; ". Mais pas suffisante.

Répondre

[^] # Re: spec.md

Posté par **Faya** le 13 novembre 2026 à 14:57. Évalué à 10 (+8/-0).

Mais est-ce que le code du LLM est *impossible* à étudier ?
Il est juste chiant à étudier mais ça n'est vraiment pas l'apanage des LLM. Dernièrement j'ai participé à un projet dans lequel un dev (senior !) a écrit un module Wordpress entièrement à la main. Il y a 4 fonctions de plus de 4000 lignes chacune avec des effets de bord dans tous les sens. Ce truc tourne en prod et en plus il fonctionne ! Par contre maintenant qu'ils veulent le migrer pour le faire sortir de Wordpress, tout le monde s'arrache les cheveux (plot-twist : c'est l'IA qui a permis de comprendre ce que faisait le code rédigé par l'humain). Donc je trouve l'argument spécieux… le code est peut-être alambiqué mais pas obfusqué. Si tu as le temps et la motivation, rien ne t'empêche d'étudier le code produit par le LLM.

Répondre

[^] # Re: spec.md

Posté par **Gof** (**site web personnel**) le 14 novembre 2026 à 08:09. Évalué à 2 (+0/-0).

Je laisse un LLM générer les .md aussi

Répondre

[^] # Re: spec.md

Posté par **Nicolas Boulay** (**site web personnel**) le 14 novembre 2026 à 10:45. Évalué à 3 (+0/-0). Dernière modification le 14 novembre 2026 à 10:46.

Je déconseille de tout laisser au llm, c'est verbeux. D'un coté, il pose plein de questions pour etre complet, de l autre le texte est peu lisible.

J ai pas encore testé mais je pense qu'un répertoire human_docs_only/ serait utile. C'est incomplet mais cela contient ce qui est vraiment important.

--
"La première sécurité est la liberté"

Répondre

Différences entre un LLM et un compilateur ?

Posté par **Vlobulle** le 13 novembre 2026 à 11:56. Évalué à 6 (+6/-2).

Question un peu naïve.

La différence entre un compilateur et un LLM, c'est quoi formellement ? Au final, ça ne fait que transcrire un code vers un autre. D'une spec rédigée en français vers un code en Rust ou d'un code en Rust vers un code binaire, on reste dans la traduction au sens large.

Alors oui, un LLM a une part stochastique dans sa transcription, et n'est pas particulièrement conçu pour optimiser du code, mais mis à part ça ? On a bien des décompilation et du transcodage de binaire de plateforme en plateforme réalisée à l'aide de LLM aujourd'hui. J'imagine qu'un jour on verra des compilateurs mainstream utiliser des algos issus de LLM pour réaliser des optimisations. Où se situera la frontière à ce moment ?

Dans ma jeunesse, j'ai fait de la génération de code semi-automatique pour faire évoluer des codes massifs (dérivation analytique avant d'avoir des outils dédiés). Aujourd'hui, la question ne se poserait pas, on utiliserait un LLM. Dans notre GIT de l'époque (CVS de mémoire), les règles de traduction étaient versionnées - elle faisaient partie intégrante du code qu'on développait.

Aujourd'hui je croise dans beaucoup de structures la pratique de versionner (avec changelog et tout) les documents de specs ou même les procédures métiers. Alors en un .docx dégueulasse qui essaye d'inclure versionning et des specs en markdown (ou je ne sais quoi de moderne), où se situe la différence ?

Bon, mon point de vue est lointain. Le seul code que je fais aujourd'hui, c'est de la recherche V sous Excel.

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **Colin Pitrat** (**site web personnel**) le 13 novembre 2026 à 14:09. Évalué à 8 (+8/-2).

La réponse est dans la question : le déterminisme.

Et aussi le côté exhaustif. Pour la majeure partie des logiciels vibe-codés, réutiliser le même prompt avec le même agent (ou un autre) produit un résultat radicalement différent. Pas juste dans le code, mais aussi dans le fonctionnement. Le code n'étant pas relu, impossible de savoir si il y a des backdoors, des bugs évidents, …

Les rares logiciels pour lesquels ça ne sera pas le cas sont ceux où l'on a donné soit une implémentation de référence à traduire (ou à copier), soit des specs très détaillées accompagnées de tests très complets. Ce dernier scénario est très rare.

Ce que j'ai vu faire aussi, c'est faire écrire les specs détaillées par le LLM. En théorie, ça devrait permettre de reproduire plus fidèlement le même comportement. Mais les specs ne sont pas vraiment connues de l'humain qui ne fait que les approuver sans trop lire. Avec les corrections de bugs, on se retrouve avec des incohérences qui peuvent facilement conduire à des absurdités dans le code qui persisteront puisqu'elles viennent de la spec.

Bref, le deuxième problème c'est la perte de contrôle. Avec les compilateurs, on perd le contrôle sur le détail de la traduction en assembleur, pas grave dans 99,9% des cas. Avec les LLMs, on perd le contrôle de la traduction de l'idée en logiciel, beaucoup plus gênant AMHA.

Le troisième problème, directement lié au deuxième : la responsabilité. Qui est responsable du résultat ? On voit de plus en plus de boîtes justifiant leurs erreurs par des "c'était un agent". C'est inacceptable. Mais si les utilisateurs doivent assumer les conséquences de ce que produisent leurs modèles, pas sûr qu'il y aura autant de preneurs…

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **gUI** (**Mastodon**) le 13 novembre 2026 à 14:31. Évalué à 10 (+12/-0).

Moi je chercherais plutôt la distinction en "un LLM" et "n'importe quel autre programmeur que moi". Je trouve qu'on fait vraiment le mauvais procès aux LLMs. Il y a plein de choses à critiquer, mais pour la qualité du code, je trouve que c'est vouloir taper à côté.

N'importe qui qui me file 10k lignes de code à comprendre c'est un enfer. Par contre, si je vous file 200 lignes de python, même générées par LLM, nous n'aurez aucun mal à les comprendre (et vous ne saurez même pas qui les a écrites tellement c'est du code "normal").

On a dénoncé pendant des décennies "les sous-traints" ou "le offshore" pour dénoncer du code écrit avec le cul, des tests insuffisants, ou qui étaient artificiellement OK… on se foutait aussi de la gueule "du stagiaire" qui a certainement codé le logiciel de réservation de la SNCF… et là tout à coup on se dit que tout ce qui est fait par un humain c'est bien, c'est propre, c'est intelligent et pertinent.

Non, ce qui est mal fait est mal fait, et s'attendre à 10k de code propre, bien construit, avec des commentaires pertinents et des variables bien nommées du premier coup à partir de quelques lignes de spec, ça n'existe pas et ça n'a jamais existé, LLM ou pas.

--
En théorie, la théorie et la pratique c'est pareil. En pratique c'est pas vrai.

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **wilk** (**site web personnel**, **Mastodon**) le 13 novembre 2026 à 19:00. Évalué à 1 (+2/-3).

> tout à coup on se dit que tout ce qui est fait par un humain c'est bien

C'est justement parce qu'on sait que c'est un problème qu'il ne faut pas l'aggraver.

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **abriotde** (**site web personnel**, **Mastodon**) le 14 novembre 2026 à 07:23. Évalué à 0 (+0/-2).

Je rajouterai une couche, les premiers compilateurs n'étaient pas si fiable, pas exempte de bugs. Et les compilateurs moderne ne sont pas si déterministe que ça. Déjà parce que cela dépends des options de compilation et du compilateur, mais aussi parce qu'il fait parfois des choix de manière indeterministe (dans les optimisations). Cela ne rend pas vraiment le comportement aléatoire mais il y a des différences dans le binaire génèré. Pire un même code génère des comportements différents suivant les architectures processeur (sur les manipulation binaires) et donc dans le code C on met des macros.
Il en est de même avec les LLM, certes le résultat peut différer en code, mais le comportement reste celui décrit. Et comme il existe des décompilateurs, le LLM peut lire un code et le décrire…

--
Sous licence Creative common. Lisez, copiez, modifiez faites en ce que vous voulez.

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **palain** le 14 novembre 2026 à 08:36. Évalué à 2 (+2/-0).

Le projet reproducible builds de debian (https://reproduce.debian.net/) montre au contraire que la compilation est déterministe. Il me semble que ce qui n'est pas encore reproductible c'est des paquets avec des données où la date est stockée. Le même compilateur dans le même environnement, avec les mêmes options donnera le même binaire.

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **Nicolas Boulay** (**site web personnel**) le 14 novembre 2026 à 10:48. Évalué à 3 (+0/-0).

Le projet de Debian montre surtout que c'est très compliqué à faire.

--
"La première sécurité est la liberté"

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **Sébastien B.** le 13 novembre 2026 à 14:34. Évalué à 10 (+9/-0).

Si un LLM c'est comme un compilateur, pourquoi tous les projets écrits avec un LLM publient le code source généré et non le matériel original (un prompt et des fichier .md ?) ?
Personne dans le logiciel libre ne commite que les exécutables des programmes générés par le compilo …

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **Nicolas Boulay** (**site web personnel**) le 14 novembre 2026 à 10:49. Évalué à 2 (+0/-1).

C est une très bonne question, et je pense que cela pourrais changer quand les llm coderont en moins de 10 min et non des heures.

--
"La première sécurité est la liberté"

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **Lionel** le 13 novembre 2026 à 14:57. Évalué à 2 (+3/-3).

Si ton LLM est un compilateur, alors pour que ton code soit libre, le compilateur doit être libre. Juridiquement (licence libre) et techniquement (4 libertés). Or, l'argument [illegible] en 1000 fois plus, pour le LLM lui-même (j'avais fait une dépêche sur le sujet à l'époque). Donc non: ton code généré par LLM ne sera jamais libre si tu considères qu'il est vraiment que l'output du compilo, parce que ton compilo lui-même ne sera jamais libre.

--
Ça, ce sont les sources. La solution que tu peux est préféré.

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **gUI** (**Mastodon**) le 13 novembre 2026 à 15:54. Évalué à 6 (+6/-1).

> parce que ton compilo lui-même ne sera jamais libre.

Attention, ça existe des LLM libres. Bon, c'est pas les meilleurs ni les plus récents, mais ça existe. En général c'est publié par des universités ou des organismes d'état. Ça inclue les données d'entraînement, le code d'entraînement, et bien évidemment les poids sous licence type MIT.

J'ai jamais trop joué avec, mais ça fait un moment que je veux faire l'expérience, et ça pourrait faire l'objet d'un journal sympa d'ailleurs.

On peut citer :

- https://eleuther.ai/
- https://allenai.org/olmo
- https://www.llm360.ai/

--
En théorie, la théorie et la pratique c'est pareil. En pratique c'est pas vrai.

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **Sébastien B.** le 13 novembre 2026 à 17:41. Évalué à 8 (+7/-1).

Si tu prends en compte la liberté de modification, il n'existe aucun LLM (j'entends, suffisamment solide pour être utile) libre puisque rares sont les personnes ayant les moyens de se payer un entraînement de LLM.

Le noyau Linux, je peux le télécharger, modifier une ligne, le compiler, l'utiliser.

Allenai, si je veux modifier un peu le set d'entraînement et refaire l'étape de pre-training, ils parlent de 5000 heure de H100 pour un modèle de 1milliards de params (vraiment peu), soit environ 10000$ (et probablement plus vu que là c'est pour faire l'entraînement sur 1 trions de tokens).

C'est inenvisageable pour à peu prêt tout le monde de payer plus de 10000$ chaque compilation.

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **abriotde** (**site web personnel**, **Mastodon**) le 14 novembre 2026 à 07:30. Évalué à 2 (+1/-1).

Le coût ne rends pas le modèle non libre. Il le rends juste non accessible au particulier mais ça c'est une autre question. Si cela coûte 10 000 euros à l'université de le faire, pourquoi ce serait pas libre ? Ça devrait le coûter moins ?
Si une voiture est 100% open-source, cela ne la rendra pas gratuite. La refaire coûtera au minimum le double du prix du marché…

--
Sous licence Creative common. Lisez, copiez, modifiez faites en ce que vous voulez.

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **ted** (**site web personnel**) le 13 novembre 2026 à 21:34. Évalué à 4 (+2/-0). Dernière modification le 13 novembre 2026 à 21:36.

Je crois que tu te fourvoies: tu peux avoir un logiciel libre même si le compilateur est propriétaire. Je suis d'accord pour dire que ça pue, touche pas à ça. Pour moi c'est du fauxpensource.

Je ne connais pas exactement ce genre de cas, mais des exemples de trucs qui s'en rapprochent:
- Le logiciel Scratch "libre" mais codé avec des techno d'Adobe
- Des logiciels utilisant Matlab, qui est propriétaire, et se targuent d'être Open Source

--
Un LUG en Lorraine : https://enunclic-cappel.fr

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **abriotde** (**site web personnel**, **Mastodon**) le 14 novembre 2026 à 07:34. Évalué à 2 (+1/-1).

Sans aller jusqu'au compilateurs: tous les logiciels libres sont compilé et tournent avec des processeurs non libre généralement avec des firmware non libre.
Bien rare sont ceux qui tourne avec RiscV avec tous les drivers et bios libre. Et même là le processeur n'est généralement pas open-source mais seulement son ISA…

--
Sous licence Creative common. Lisez, copiez, modifiez faites en ce que vous voulez.

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **abriotde** (**site web personnel**, **Mastodon**) le 14 novembre 2026 à 07:50. Évalué à 1 (+0/-1).

Et j'ajouterai des domaines moins clivants mais tout de même problématiques :
- les logiciels Apple dont le code est libre mais où il faut généralement un compte développeur Apple payant pour faire un logiciel Apple…
La seule chose c'est que le compilateur Swift (et objective-C) existe sous Linux ce qui permet, si le code a été adapté de le faire tourner sous Linux.
- les soft Android, Apple et Microsoft utilisent généralement des API propriétaires dont ils dépendent entièrement pour leur fonctionnement. Sur le principe on a un logiciel dont seul une petite parti de ce qui tourne est libre (puisque les lib ne le sont pas) et l'os non plus. Alors sous Androïd microG tente de les reproduire en open-source et on a e/OS…

--
Sous licence Creative common. Lisez, copiez, modifiez faites en ce que vous voulez.

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **Nicolas Boulay** (**site web personnel**) le 14 novembre 2026 à 10:52. Évalué à 3 (+0/-0).

C est faux.

Le compilo libre n'a jamais été une condition.

https://www.gnu.org/licenses/gpl-faq.en.html#NonFreeTools

--
"La première sécurité est la liberté"

Répondre

[^] # Re: Différences entre un LLM et un compilateur ?

Posté par **PhRæD** le 13 novembre 2026 à 22:07. Évalué à 7 (+5/-0).

> Dans ma jeunesse, j'ai fait de la génération de code semi-automatique pour faire évoluer des codes massifs (dérivation analytique avant d'avoir des outils dédiés). Aujourd'hui, la question ne se poserait pas, on utiliserait un LLM.

Spoiler : ça ne marche pas !

Tout simplement parce qu'un LLM n'est pas capable de répétabilité. Une équipe informatique dans le groupe,bancaire dans lequel je travaille voulais-tu migrer ses contrats d'interface d'API d'une norme à une autre. Vouloir le faire faire par le LLM n'a été qu'un très gros gâchis de tokens. La bonne approche a été de construire avec le LLM l'application de migration.

Plus ça va plus je pense que loin de ce que nos chefs avaient rêvé, le LLM ne pourra pas remplacer les programmeurs : un LLM n'a aucune « idée » de ce qu'est « l'esprit » d'une application, un programme étant bien plus qu'une suite de lignes de code.

Par contre, c'est un « super-assistant » corvéable à merci.

--
« Y a même des gens qui ont l'air vivant, mais ils sont morts depuis longtemps ! »

Répondre

Un peu n'imp

Posté par **Misc** (**site web personnel**) le 13 novembre 2026 à 12:17. Évalué à 10 (+11/-2).

C'est franchement n'importe quoi.

Les tentatives constantes de redéfinir ce que libre veut dire, c'est assez fatiguant. C'était fatiguant quand des boites ont tentés de le faire via des trucs comme la BUSL, comme les badgeswares, etc, c'est aussi fatiguant quand c'est pas des boites. Et c'est fatiguant parce que c'est hypocrite.

Venir définir le logiciel libre sur la base des compétences de la personne qui va lire le code, c'est faire preuve d'un aveuglement historique. Le logiciel libre, comme je l'ai plusieurs dit en commentaires, c'est avant tout des libertés donné aux personnes capable de coder, vu que la moitié des 4 libertés sont à destination de ce groupe. La liberté 1, ça demande des compétences de dev, la liberté 3 aussi. J'ai pas besoin de faire des stats pour expliquer que la grande majorité de la population ne sait pas coder, donc bénéficie de moins de liberté que le groupe des personnes qui savent. On m'a toujours dit "oui, mais les gens peuvent payer quelqu'un pour ça", transformant ça en un problème de pognon, pognon qui est égalitairement réparti dans le monde comme on le sait tous (et trahissant quand même un certain privilège des ingés à ce niveau, surtout quand on regarde la réalité des financements du libre en pratique).

Venir rajouter la question de la lisibilité dans l'équation tout d'un coup, c'est donc un oubli historique car ça a toujours été la. La différence, c'est qu'avant, on était plus facilement du bon coté. Il y a quand même un coté "peur du déclassement" qui n'est pas exprimé ouvertement mais assez visible quand on y pense, et on sait tous comment ça se termine quand la dite peur est instrumentalisé en politique en ce moment.

Et un autre point qui me mets en rogne, c'est la question de l'outil tiers et la distinction qui est fait à ce niveau. On a toujours eu besoin d'outils tiers pour s'occuper du code. Parfois, c'est des outils tiers libres, mais pendant longtemps, ça a été quasiment impossible de faire ça avec 0% de code proprio. Est ce qu'un logiciel qui ne tourne que sous Windows ne peut pas être libre ? Est ce qu'un logiciel qui dépend des drivers nvidia, des firmwares proprios de sa carte wifi ou d'un bios uefi ne peut pas être libre ? La réponse est clairement "non", car il serait ridicule de dire qu'un logiciel n'a été libre qu'aprés la création de gcc, de Linux ou de Coreboot. Les 4 libertés que j'ai ne dépendent pas du fait que j'ai besoin de logiciel proprio pour les exercer, et il n'y a jamais eu de ligne franche, donc il me semble ridicule d'en mettre une maintenant quand la majorité des gens continuent à utiliser des archis proprios (x86_64) avec des bios/uefis proprios, communiquant avec des services proprios (github) pour produire du code qu'on voit comme libre.

Il y a plein de sujet de discussion avec les LLMs (genre sur l'impact en consommation, la centralisation du pouvoir, etc) mais le fait de produire du code qui n'est pas maintenu n'en ai pas un, surtout quand on voit la quantité de code qui est de facto produit et non maintenu.

> Avez-vous observé des situations où, après avoir commencé à utiliser des LLMs dans la gestion du code jusqu'à un certain point, les développeurs sont revenus sur leur décision et ont réduit ou banni l'utilisation de ces outils ?

De ce que j'ai vu, c'est surtout des exemples anonymes sur Reddit, de temps en temps. Il y a pas eu d'études ou de vérification à grande échelle. Mais bon, ça sert à rien d'avoir des données quand on peut avoir des anecdotes.

Par exemple, je vois sur les réseaux sociaux des gens qui sous entendent que la panne d'Office365, c'est parce que 1) Microsoft a viré du monde 2) y a eu des trucs vibecodés. Bien sur, personne bossant chez Microsoft ne dit ça, rien ne montre que ça soit le cas, et on va pas avoir de post mortem avant quelques jours. La pile technique de Azure était déjà affreusement complexe et sans doute mauvaise bien avant 2026 comme on peut le voir des dans des discussions comme ici (ça parle de 173 agents par noeud azure, et ça remonte jusqu'en 2009). Il est plus probable que ça soit la cause de la panne, bien plus que les 2 autres causes du moment.

Si on veut des données, il y a ce papier de recherche qui semblent dire qu'il y a pas trop d'évolution. Ensuite, je comprends bien que comme c'est pas une anecdote mal sourcé qui va dans le sens qu'on veut, ça vaut sans doute pas grand chose.

Et pourtant, des codes bases fait par 100% d'IA, j'en ai regardé, et ouais, c'est pas joli (mais bon, j'ai aussi vu des horreurs de script en bash pour des initscripts de logiciel proprios datant d'il y a 35 ans). Et regarder des scripts/programmes fait 100% par IA, mais pas en mode "singe sous crack", j'ai aussi fait, j'ai été appelé quand le modèle a eu du mal à gérer (genre du code python qui mélange sync et async, et suffisamment gros pour que ça rentre pas dans le contexte, ça coince). Mais je pense pas que la solution soit de bannir, car clairement, il y a des gens qui sont productifs avec. Et en fait, il y a même des gens qui n'auraient rien fait sans ça. Mon chef, ex journaliste, a écrit un script pour poster des news de baseballs sur bluesky (le truc que j'ai regardé avec sync/async dont je parle). Jamais il aurait pu le faire de lui même sans un gros investissement de temps, et jamais il ne pourrait passer du temps pour corriger un bug sans ça. Et c'est un point articulé par Luis Villa dans un post de blog récent, à savoir l'ouverture à des gens qu'on ne touchait pas avant.

Répondre

[^] # Re: Un peu n'imp

Posté par **Ysabeau** (**site web personnel**, **Mastodon**) le 13 novembre 2026 à 17:06. Évalué à 7 (+6/-2). Dernière modification le 13 novembre 2026 à 17:23.

> Mais je pense pas que la solution soit de bannir, car clairement, il y a des gens qui sont productifs avec. Et en fait, il y a même des gens qui n'auraient rien fait sans ça.

Certes, le problème c'est que ces gens inondent ensuite les dépôts et les sites de leur iasaloperies, ici, souvent, en faisant écrire l'article par un LLM. Et c'est ça qui lasse à peu près tout le monde. Enfin, tous les gens qui doivent gérer, affronter ces déluges. Dans le lot, il y a probablement du bon, mais aussi du mauvais et surtout, des trucs qui réinventent la route et pas forcément en mieux.

La productivité pour la productivité, ça n'a aucun sens.

--
Je n'ai aucun avis sur systemd

Répondre

Usage des LLMs

Posté par **black_magic** le 13 novembre 2026 à 12:53. Évalué à 4 (+9/-5).

Opposer un code humain prétendument parfait à un code généré par LLM jugé illisible et immaintenable pour conclure qu'il faut rejeter l'outil en bloc est une posture très manichéenne.

L'abstraction est l'essence même de l'informatique. Un langage de haut niveau masque déjà la réalité du processeur et de son jeu d'instructions. En début de carrière dans les assurances, j'ai assisté aux mêmes querelles entre les puristes de l'assembleur sur fichiers indexés et les développeurs COBOL (DL/I, puis DB2) dont je faisais partie : on nous accusait déjà de perdre le contrôle de la mémoire et du stockage disque. Quant à la maintenabilité, l'humain n'a pas attendu l'IA pour produire des monstres intouchables vieux de dix ans, y compris au coeur de projets Open Source majeurs (glibc, zlib, Autotools, sendmail…).

Un LLM n'est qu'un outil, et générer du code brut n'est qu'une facette de son utilité :

- Revue de code : Lui faire relire son propre travail est une excellente leçon d'humilité.
- Architecture et traçabilité : Via des méthodologies dédiées comme superpowers, il aide à rédiger des spécifications et des plans d'implémentation documentant chaque décision dans des fichiers Markdown versionnés. Le codage proprement dit devient alors optionnel.
- Tests unitaires : Idéal pour déléguer le côté rébarbatif de l'écriture tout en gardant le contrôle : je conçois la liste de scénarios à la main et je trie.

Ne jetons pas le bébé avec l'eau du bain : tout est question de pratique. Restent bien sûr les vrais sujets de fond : souveraineté face aux géants américains ou chinois et coût écologique ; mais ils méritent un débat à part entière.

Répondre

Parlez de ce que vous connaissez ?

Posté par **Axioplase** (**site web personnel**) le 13 novembre 2026 à 13:50. Évalué à 5 (+7/-4).

> Les LLMs ont un effet, involontaire, de rendre le code illisible. Le code ainsi généré n'est pas libre, à terme.

Bon, ça va faire deux ans que je code avec des LLMs professionnellement. Codex, Claude, Llama, ou autre modèles parfois pas encore publiques. Quand je lis DLFP, j'ai l'impression de vivre dans un autre monde.

Le code généré par mes LLMs est tout à fait lisible. Il est clair, et bien organisé. Pourquoi ? Parce que je passe 2-3 jours à faire mon design. Parce que le LLM *code*, mais je suis toujours l'ingénieur qui planifie, organise, gère, etc. En gros, je fais mon travail.

Tu veux voir mon architecture ? Regarde les diagrammes de data, de flow, de sequence que j'ai définis.
Tu veux voir mon planning ? Regarde le document avec les milestones, les taches, l'arbre de dépendences.

La manière de laquelle on travaille est "considère le LLM comme un employé junior; maintenant, livre ce projet avec autant de juniors qu'il t'est nécessaire". Et je peux te dire que ce junior, en plus, il s'améliore incroyablement de jour en jour…

Pour paraphraser un meme/tweet que j'ai lu l'autre jour "si tu as peur que ton métier d'ingénieur logiciel soit en danger, sache que j'ai rencontré quelqu'un qui a dépensé plus d'un million de tokens pour créer une application web de jeu d'échecs."

Répondre

Envoyer un commentaire

Suivre le flux des commentaires

Note : les commentaires appartiennent à celles et ceux qui les ont postés. Nous n'en sommes pas responsables.

Revenir en haut de page

Derniers commentaires

- Re: Ca sert à quoi ?
- Re: Les ravages de l'IA
- Re: Ca sert à quoi ?
- Re: Ca sert à quoi ?
- Re: Remplissage
- Re: Différences entr…
- Re: Différences entr…
- Re: Différences entr…
- Re: spec.md
- Re: Sans la source?
- Re: Sans la source?
- Re: Sans la source?

Étiquettes (tags) populaires

- intelligence_artificielle
- slop
- merdification
- sécurité
- cybersécurité
- linux
- php
- vibe_coding
- microsoft
- administration_fran…
- disquette
- grands_modèles_de…

Sites amis

- Agenda du Libre
- April
- Éditions D-BookeR
- Éditions Diamond
- Éditions Eyrolles
- Éditions ENI
- En Vente Libre
- Framasoft
- La Quadrature du Net
- Lea-Linux
- Open Source Initiative
- Imprimerie Grafik Plus

À propos de LinuxFr.org

- Mentions légales
- Faire un don
- L'équipe de LinuxFr…
- Informations sur le s…
- Aide / Foire aux que…
- Suivi des suggestion…
- Wiki du site
- Règles de modération
- Statistiques
- API pour le dévelop…
- Code source du site
- Plan du site