La prochaine frontière : IA et code

Christian Jauvin

Daniel Lemire, professeur

Université du Québec (TÉLUQ)

Montréal 🇨🇦

blog: https://lemire.me

X: @lemire

GitHub: https://github.com/lemire/

Plan de la présentation

- L'état actuel de l'IA générative pour le code
- Comment fonctionnent les grands modèles de langage (LLM)
- Outils d'assistance au codage
- Qualité et performance du code généré
- Impacts sur la profession
- Ce qui vient ensuite

La révolution en cours

- Les LLM génèrent du code depuis ~2021 (Codex, Copilot)
- En 2026, les assistants IA sont omniprésents
- Plus de 70% des développeurs utilisent un assistant IA
- La question n'est plus « si » mais « comment »

Qu'est-ce qu'un grand modèle de langage?

- Réseau de neurones entraîné sur d'immenses corpus de texte
- Prédit le prochain jeton (token) dans une séquence
- Le code est du texte : les LLM s'y appliquent naturellement
- Pas de « compréhension » au sens humain, mais une capacité de synthèse remarquable

L'entraînement en bref

- Phase 1 : pré-entraînement sur des milliards de lignes de code
- Phase 2 : fine-tuning avec des exemples de haute qualité
- Phase 3 : RLHF — apprentissage par rétroaction humaine
- Résultat : un modèle qui produit du code syntaxiquement correct la majorité du temps

Les outils d'aujourd'hui

Outil	Approche
GitHub Copilot	Complétion en ligne dans l'éditeur
Claude Code	Agent en ligne de commande
Cursor / Windsurf	IDE augmenté par IA
ChatGPT / Claude	Conversation interactive
Codex CLI	Exécution autonome de tâches

Démonstration typique

```
# Demande : trier une liste de dictionnaires par clé "score"
données = [{"nom": "Alice", "score": 88},
           {"nom": "Bob", "score": 95},
           {"nom": "Charlie", "score": 72}]

résultat = sorted(données, key=lambda x: x["score"],
                  reverse=True)
```

L'IA génère ce code en moins d'une seconde.

Mais est-ce **performant**?

Le piège de la facilité

- Le code généré compile et passe les tests de base
- Mais il peut être :
  - Sous-optimal en performance
  - Fragile face aux cas limites
  - Difficile à maintenir
- « Ça marche » ≠ « C'est bon »

Mesurer la performance : pourquoi c'est crucial

- Un programme 10× plus lent, c'est 10× plus de serveurs
- L'énergie consommée est proportionnelle au temps CPU
- La performance, c'est aussi de la durabilité
- Daniel Lemire : « La performance n'est pas un luxe, c'est une responsabilité »

Exemple : parsing JSON

- **simdjson** : ~3 Go/s sur un seul cœur
- Code naïf généré par IA : ~200 Mo/s
- Facteur **15×** de différence
- L'IA ne connaît pas les instructions SIMD spécialisées
- L'expertise humaine reste irremplaçable pour l'optimisation bas niveau

L'IA et les algorithmes

- Les LLM reproduisent les algorithmes courants (tri, recherche)
- Mais peinent avec :
  - L'optimisation vectorielle (SIMD)
  - Les structures de données non standard
  - Les compromis espace/temps subtils
- Les algorithmes nouveaux exigent encore une réflexion humaine

Bitmaps compressés : un cas d'étude

- Roaring Bitmaps : utilisé par Google, Apache, Uber
- Structure hybride : tableaux, bitmaps, runs
- L'IA peut *utiliser* la bibliothèque
- Mais elle ne pourrait pas *inventer* cette structure
- L'innovation algorithmique reste humaine

Ce que l'IA fait bien

- Écrire du code « boilerplate »
- Traduire entre langages
- Générer des tests unitaires
- Documenter du code existant
- Prototyper rapidement
- Expliquer du code inconnu

Ce que l'IA fait mal

- Raisonner sur la complexité algorithmique
- Optimiser pour le matériel spécifique
- Comprendre le contexte métier profond
- Gérer la concurrence et les conditions de course
- Garantir la sécurité (injections, failles)
- Maintenir la cohérence dans un grand projet

Le problème de l'hallucination

- Les LLM inventent des API qui n'existent pas
- Ils citent des bibliothèques fictives
- Le code peut sembler plausible mais être incorrect
- Vérification humaine **obligatoire**
- « Faire confiance, mais vérifier » — surtout vérifier

Transparence et honnêteté

- Christian Jauvin propose le concept de « Proof of Prompt »
- Idée : déclarer ouvertement quand l'IA a contribué
- Pas de honte à utiliser un outil
- Mais l'honnêteté intellectuelle exige la transparence
- Le code généré par IA devrait être étiqueté

L'impact sur les développeurs juniors

- Risque : apprendre à « prompter » au lieu de « programmer »
- L'IA masque la complexité sous-jacente
- Un développeur qui ne comprend pas le code qu'il utilise est fragile
- L'apprentissage des fondamentaux reste essentiel
- L'IA est un accélérateur, pas un substitut à la compétence

L'impact sur les développeurs seniors

- Productivité accrue pour les tâches répétitives
- Plus de temps pour l'architecture et la réflexion
- Nouveau rôle : « réviseur de code IA »
- La valeur se déplace vers le jugement, pas l'exécution
- Les experts en performance sont plus précieux que jamais

Benchmarks : code humain vs code IA

Tâche	Humain expert	IA (2026)
Code correct	95%	85%
Code performant	80%	40%
Code sécuritaire	75%	50%
Code maintenable	85%	60%

Estimations basées sur la littérature récente

Le coût énergétique de l'IA

- Entraîner un LLM : des milliers de MWh
- Chaque requête de complétion consomme de l'énergie
- Paradoxe : l'IA peut générer du code inefficace
  qui consomme *plus* de ressources à l'exécution
- Il faut mesurer le coût total : génération + exécution

Les langages et l'IA

- L'IA est meilleure en Python et JavaScript (plus de données)
- Performance variable en C, C++, Rust
- Les langages moins courants (Zig, Nim) posent problème
- Biais vers les solutions « populaires », pas les « optimales »
- Le choix du langage influence la qualité de la sortie IA

Tests et validation

- L'IA peut générer des tests, mais :
  - Elle teste ce qu'elle « pense » être correct
  - Les cas limites sont souvent négligés
  - La couverture est superficielle
- Le test reste une discipline humaine
- Fuzzing et tests de propriété : compléments essentiels

Sécurité du code généré

- Les LLM reproduisent des patrons vulnérables (injection SQL, XSS)
- Ils ne comprennent pas les modèles de menace
- Le code généré doit passer par les mêmes revues de sécurité
- Ne jamais déployer du code IA sans audit
- La sécurité est un processus, pas une fonctionnalité

L'IA comme partenaire de revue de code

- Excellente pour détecter :
  - Le code mort
  - Les incohérences de style
  - Les bogues évidents
- Moins fiable pour :
  - Les problèmes d'architecture
  - Les failles de sécurité subtiles
  - La logique métier

L'avenir proche (2026–2028)

- Agents autonomes capables de résoudre des bogues complets
- Meilleure intégration avec les outils de build et CI/CD
- Modèles spécialisés par domaine (embarqué, web, données)
- Compréhension de projets entiers, pas juste de fichiers isolés
- Boucle de rétroaction : l'IA apprend du code en production

L'avenir lointain : spéculations

- Programmation en langage naturel comme interface primaire?
- Les langages de programmation deviennent-ils des « bytecode »?
- Le développeur comme « architecte d'intention »?
- Ou bien : les fondamentaux restent, l'IA accélère
- L'histoire de l'informatique suggère : les abstractions montent, mais le bas niveau ne disparaît jamais

Conseils pratiques

1. Utilisez l'IA pour le code jetable et le prototypage
2. Vérifiez **toujours** le code généré
3. Mesurez la performance avant de faire confiance
4. Investissez dans votre compréhension des fondamentaux
5. Soyez transparent sur l'utilisation de l'IA
6. Restez curieux : l'outil évolue rapidement

Ce qu'on ne vous dit pas assez

- L'IA ne remplace pas 10 ans d'expérience en une requête
- Les benchmarks marketing ≠ la réalité du terrain
- Le code le plus rapide est celui qu'on n'exécute pas
- Un bon algorithme bat toujours un mauvais algorithme optimisé par IA
- La pensée critique est votre compétence la plus précieuse

Conclusion

- L'IA transforme le développement logiciel — c'est indéniable
- Mais elle ne remplace pas l'expertise, elle l'amplifie
- La performance, la sécurité, le jugement restent humains
- Adoptons ces outils avec lucidité et rigueur
- La prochaine frontière, c'est la collaboration humain-IA

Questions?

Christian Jauvin — cjauvin.github.io

Daniel Lemire — lemire.me

Merci! 🇨🇦