Comment intégrer de l'intelligence dans les processus de développement logiciels ?
Porteurs : Anne Etien, Nicolas Anquetil, Christelle Urtado, Jean-Rémy Fallieri

La place du logiciel dans nos sociétés est toujours croissante. La France, par exemple a beau former chaque année plus d'ingénieurs en informatique, il en manquerait 80 000 selon la Direction de l'animation de la recherche des études et des statistiques (Dares). Et le reste du monde est confronté au même problème.

Pour faire face à cette pénurie qui existe depuis de nombreuse années, le génie logiciel a intégré des techniques venues d'autres domaines de l'informatique pour automatiser un certain nombre de tâches répétitives. On notera par l'exemple l'introduction au début des années 2000 de l'ingénierie dirigées par les modèles pour automatiser certaines parties du développement du logiciel et pour permettre une plus grande adaptation aux nouvelles plateformes.

Par ailleurs, l'intelligence artificielle est de plus en plus présente dans notre quotidien dans des domaines variés. Par exemple:

- les banques utilisent des systèmes experts pour évaluer les risques liés à l'octroi d'un crédit ;
- l'armée américaine a demandé l'aide d'entreprise pour développer une intelligence avancée de collecte d'information et avec la capacité de décision rapide pour l'aider à attaquer ses ennemis rapidement à leurs points les plus vulnérables ;
- la médecine, s'appuie sur des systèmes experts pour d'aider au diagnostic de maladie comme le cancer, ou des maladies oculaires.
- Certaines polices - par exemple, la police britannique - font actuellement développer une IA pour prévenir des crimes et des délits.
- dans le domaine de la logistique, des approches heuristiques permettent de résoudre des problèmes de satisfaction de contraintes ou dans le domaine des transports en commun, pour faciliter la régulation et la gestion du trafic au sein des réseau de plus en plus complexes, comme le système UrbanLoop actuellement en cours de développement dans la ville de Nancy.

L'idée de ce défi est de voir *comment des techniques d'apprentissage peuvent aider dans le processus de développement logiciels*.

L'intérêt de la fouille de données et de l'apprentissage automatique pour l'évolution logicielle ont été très tôt reconnus. Dans les conférences telles que ASE, ou ICSE, les articles utilisant des techniques d'IA sont de plus en plus fréquents. L'apprentissage automatique a été appliqué à diverses problématiques telles que :

- Découverte automatique de règles architecturales [MVT+16] ;
- Découverte automatique de règles d'évolution d'API [HEA+14] ;
- Automatisation des messages de commit [LXH+18], [JAM17] ;
- Génération automatique de tests ;
- Prédiction de défaut juste à temps (*just in time defects*) [YLX+15] ;
- Vérification de conventions de nommage [BRV15]
- Amélioration de systèmes de complétion de code à partir d'exemples [BMM09]
- Explication automatique de code par la génération de commentaires

Le génie logiciel produit de plus en plus de données qui sont conservées sous différentes formes (commits, traces d'exécution, changements, bugs,...). L'idée de ce défi est donc

d’explorer les techniques d’apprentissage très variées (sequential learning, deep learning, statistical learning sont quelques grande familles) pour tirer parties des bases de codes massives (on parle aussi de Big Code) et résoudre des problèmes difficiles.

Les analyses génériques de l'IA se heurtent à trois caractéristiques fondamentales en génie logiciel :

Premièrement, en programmation, chaque "détail" est potentiellement critique. Les erreurs de programmes C où le test d'égalité ('==') est remplacé par une affectation ('=') sont bien connues des programmeurs. L'IA repose surtout sur la statistique et les probabilités. Son principe est que les décisions qui, en général, sont bonnes, peuvent s'avérer fausses dans un (petit) nombre de cas particuliers. Le génie logiciel est donc un domaine qui, intrinsèquement, n'est pas bien adapté à l'utilisation de l'IA, et il faut prévoir des mécanismes de gestion de ce hiatus. Ces mécanismes peuvent-être soit automatiques, comme par exemple dans la correction automatique de bugs, où des tests automatisés permettent de vérifier la qualité de la correction ; soit manuels en soumettant la décision à un programmeur humain.

Deuxièmement, l'IA n'invente rien, mais reproduit des comportements déjà connu et familiers. Elle se "construit" à partir de grandes masses de données dans un domaine précis déjà bien sédimenté. De telles masses de données existent en génie logiciel (dépots de projets tels GitHub), mais ils rendent compte du passé. Dans un domaine où la puissance des ordinateurs augmente très rapidement (ex: loi de Murphy), les évolutions technologiques sont tout aussi rapides et ce qui était impossible quelques années auparavant devient courant. Cela a une influence sur les projets informatiques où chaque projet peut se baser sur des innovations profondes qu'aucun projet précédent ne prend en compte.

Troisièmement, les programmes informatiques étant (comme nous l'avons déjà dit) des pures constructions de l'esprit, se pose le problème classique de décrire la solution que l'on souhaite en termes suffisamment précis pour permettre une implémentation informatique. Si cette description est très détaillée, elle devient le programme en elle-même, si elle ne l'est pas assez, on court le risque de ne pas pouvoir générer de programme exécutable à partir d'elle.

L'utilisation de l'IA pour résoudre des problèmes typiques de génie logiciel ne va donc pas de soi et demande une analyse détaillée des solutions qu'elle peut offrir en regard des nombreux problèmes qui se posent :

- Quelles techniques d’IA pour quelles classes de problèmes du génie logiciel ? Comme listé ci-dessus, des propositions de solutions existent déjà. Mais elles sont encore très embryonnaires et on ne peut présager de leur bienfondé pratique à ce stade. Plus de recherche est nécessaire.
- Le génie logiciel est un domaine où les aspects sociaux ont aussi un très fort impact. Comment intégrer les techniques d’IA dans le développement logiciel pour que les deux intelligences (humaine et artificielle) collaborent efficacement ?
- Les défis techniques sont nombreux et variés et comme déjà discuté, ne sont pas que des détails. Certaines technologies (langages de programmation, piles techniques) sont-elles mieux adaptées que d’autres à l'utilisation de l'IA ? Les solutions proposées seront-elles portables d'une technologie à une autre ?

[BRV15] Pavol Bielik, Veselin Raychev and Martin Vechev. Programming with "Big Code": Lessons, Techniques and Applications. *1st Summit on Advances in Programming Languages (SNAPL 2015)*, 2015

[BMM09] Marcel Bruch, Martin Monperrus, and Mira Mezini. Learning from examples to improve code completion systems. In *Proc. of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering (ESEC/FSE'09)*, pages 213–222, 2009.

[HEA+14] Andre Hora, Anne Etien, Nicolas Anquetil, Stéphane Ducasse, and Marco Túlio Valente. Apievolutionminer : Keeping api evolution under control. In *Proceedings of the Software Evolution Week (CSMR- WCRE'14)*, 2014.

[MVT+16] Cristiano Maffort, Marco Túlio Valente, Ricardo Terra, Mariza Bigonha, Nicolas Anquetil, and Andre Hora. Mining architectural violations from version history. *Empirical Software Engineering*, 21(3) :854–895, jan 2016.

[LXH+18] Zhongxin Liu, Xin Xia, Ahmed E. Hassan, David Lo, Zhenchang Xing, and Xinyu Wang. 2018. Neural-machine-translation-based commit message generation: how far are we?. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE 2018). ACM, New York, NY, USA, 373-384. DOI: https://doi.org/10.1145/3238147.3238190

[JAM17] Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. Automatically generating commit messages from diffs using neural machine translation. In Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2017). IEEE Press, Piscataway, NJ, USA, 135-146

[YLX+15] X. Yang, D. Lo, X. Xia, Y. Zhang and J. Sun, "Deep Learning for Just-in-Time Defect Prediction," 2015 IEEE International Conference on Software Quality, Reliability and Security, Vancouver, BC, 2015.