Développement collaboratif d'outils par des agents LLM et humains pour résoudre des problèmes complexes : application à la synthèse scientifique

Xavier Daull[*,**], Elisabeth Murisasco[*]
Patrice Bellot[*], Emmanuel Bruno[*], Vincent Martin[**]

[*]LIS, UMR 7020, www.lis-lab.fr
[**]Naval Group, naval-group.com

Résumé. Cette démonstration présente une architecture collaborative pour le développement itératif d'outils neuro-symboliques utilisant modèles de langage (LLM) et intervention humaine pour la synthèse scientifique. L'approche se concentre sur l'apprentissage de génération de contenus structurés, tels que des articles de recherche ou des brevets, à partir d'un titre et d'un résumé initiaux. Le processus d'apptentissage intègre quatre agents LLM principaux (Coach, Développeur, Critique, Capitaliseur) dans une boucle de feedback humain afin d'améliorer continuellement les performances des outils. Les contributions clés incluent : (1) un cadre d'apprentissage itératif combinant distances sémantiques automatisées et feedback humain ; (2) une collaboration homme-LLM pour générer et valider des fonctions Python neuro-symboliques adaptées à des tâches complexes ; (3) l'adaptation et l'extension à différents types de contenus (articles scientifiques, brevets, synthèses).

1 Introduction

Les récents progrès des modèles de langage à grande échelle (LLMs) auto-apprenants et des cadres génériques de création d'outils (Cai et al., 2023) ont démontré leur efficacité dans divers domaines, allant des tâches dans des environnements de jeu ouverts (Wang et al., 2023; Zhu, 2023) aux problèmes séquentiels (Colas et al., 2023) et mathématiques bien définis (et al, 2023). Cependant, la résolution de problèmes complexes, non structurés et nécessitant des itérations, tels que la rédaction de documents scientifiques longs, reste un défi important. Pour répondre à ces défis, les LLMs doivent intégrer des processus cognitifs multi-étapes, allant de raisonnements séquentiels internes (Chu, 2023), en passant par des actions collaboratives (Yao et al., 2022), jusqu'à des collaborations multi-agents (Chen et al., 2024). Des travaux récents comme LATM (Cai et al., 2023) ou Voyager (Wang et al., 2023) ont montré que les LLMs peuvent explorer un espace de problème de manière autonome et développer une bibliothèque d'outils pour résoudre des tâches dans des environnements ouverts comme Minecraft, sans intervention humaine. D'autres approches, comme FunSearch (et al, 2023), associent un LLM à un évaluateur dans un système itératif, permettant de découvrir des solutions optimisées pour

des problèmes complexes. AFlow (AFl, 2024) adopte une approche de recherche pour optimiser les workflows, tandis qu'Iterated Decomposition (Reppert, 2023) introduit un retour humain pour affiner les étapes intermédiaires des tâches scientifiques complexes. Plus largement, l'apprentissage par renforcement avec retour humain (RLHF) (Dong et al.) est utilisé pour affiner les modèles de récompense, avec des résultats promettant une meilleure réutilisation des feedbacks (Burns, 2024; Sun et al., 2024). Enfin, le cadre AI Scientist (Lu et al., 2024) illustre comment les LLMs peuvent générer des hypothèses scientifiques, conduire des expériences et rédiger des articles, démontrant leur potentiel dans les processus de recherche.

Ce papier propose un cadre conceptuel et un système pratique intégrant des LLMs multi-agents avec un mécanisme de feedback humain (HITL) pour créer itérativement des outils adaptés aux tâches de synthèse scientifique complexe. L'approche combine l'automatisation et l'intervention humaine pour optimiser les coûts et les courbes d'apprentissage. Nous introduisons un système d'amélioration continue utilisant des métriques automatisées pour guider et évaluer la structure des documents, la qualité du contenu et la précision des références. Nos contributions incluent : (1) l'utilisation des LLMs et des feedbacks humains pour développer des outils réutilisables adaptés à des problèmes complexes ; (2) l'introduction d'une mesure de distance sémantique (macro-guidance) associée à des feedbacks humains détaillés (micro-guidance) pour faciliter l'apprentissage ; (3) la création d'un ensemble de données et de métriques pour évaluer ces défis.

2 Principes

La figure 1 illustre notre architecture, mettant en avant la dynamique collaborative entre les agents humains et les agents LLM dans un environnement de résolution de problèmes. Ce cadre d'interaction démontre l'apprentissage itératif et continu dans une boucle de processus impliquant 4 agents LLM (Coach, Développeur, Critique, Capitaliseur) assistés, lorsque nécessaire, par des agents humains qui peuvent superviser et affiner les agents LLM.

Le Coach définit le prochain outil à apprendre, le Développeur l'implémente, le Critique évalue son succès et, en cas d'échec, renvoie le processus au Développeur tant que le nombre maximal d'essais n'est pas atteint. Le Capitaliseur documente les outils implémentés avec succès, ainsi que les échecs, pour apprentissage futur lorsque le nombre maximal d'essais est atteint (par exemple, si une tâche est jugée "trop difficile"). Cette bibliothèque d'outils informe le Coach sur l'évolution de l'apprentissage, fournissant un état des outils disponibles.

La mémoire d'évolution du problème commence avec un problème vide et vise à produire un problème résolu en sortie. Elle est utilisée pour évaluer les gains d'une implémentation d'outil et pour permettre au Coach de définir les outils dans le contexte de l'état actuel du problème. Ce cycle met en lumière les aspects d'apprentissage continu et d'amélioration du système, soulignant l'importance de la collaboration, de l'utilisation des outils et des retours dans la résolution de problèmes complexes.

Conception et agents du système : la génération de synthèses scientifiques longues et non factuelles (ex. : état de l'art, article scientifique de type Wikipédia) requiert une bibliographie étendue, des analyses approfondies et une sortie dans un format acceptable. Même les modèles de langage les plus avancés (LLM, tels que GPT4o, Llama 3.x, Mistral Large, Gemini) rencontrent des difficultés face à ces exigences, en raison des limitations de contexte, des recherches itératives et ouvertes et de l'alignement avec des intentions complexes. Nous

proposons une architecture apprenant itérativement les outils les mieux adaptés aux objectifs tout en minimisant l'espace de recherche et les interventions humaines.

Cette architecture repose sur les éléments suivants : Le **modèle de simulation et mémoire du problème** simule des actions (essais, retours en arrière, évaluation des distances à la solution) via un modèle de document instancié pour chaque problème (objectif, plan structuré, contenu, ressources). Les encodages sémantiques (sectionnels/globaux) calculent des distances sémantiques pour guider l'amélioration des contenus dans un environnement où humains et LLM testent des outils pour résoudre les problèmes.

Un **processus d'apprentissage continu** associe LLM et humains pour développer, valider, améliorer, et capitaliser les outils via une logique d'apprentissage par renforcement (Yang et al., 2023). Inspiré de Voyager (Wang et al., 2023), il repose sur 4 agents collaborant : (1) le **Coach** spécifie l'outil à développer, (2) le **Développeur** implémente le code, (3) le **Critique** évalue les résultats, (4) le **Capitaliseur** documente et centralise.

Une **boucle de feedback macro-micro** assure des itérations basées sur des retours quantitatifs (distances sémantiques) et qualitatifs (validation par le Critique ou retours humains). Les humains interviennent en phases initiales pour guider ou affiner les sorties, avec des agents LLM intégrant ces retours via l'apprentissage en contexte (Min et al., 2022).

Les **agents humains** fournissent des retours avant (pré-inférence) ou après (post-inférence), enrichissent les instructions, critiquent, valident ou ajustent directement les propositions des LLM. Ces interventions sont capitalisées sous forme d'exemples enrichissant les interactions homme-LLM.

Le **processus d'inférence** décompose les tâches complexes en sous-tâches, génère/exécute un code d'orchestration, et utilise des outils existants ou nouvellement développés. Les humains interviennent principalement en phase d'apprentissage mais peuvent optimiser les résultats en phase d'inférence.

Une **base de connaissances** centralise outils, retours humains et données pour enrichir le contexte des agents, optimisée par recherche sémantique et règles.

Les **outils développés et fonctions neuro-symboliques** sont intégrés dans une bibliothèque réutilisable, accompagnés de métadonnées sur leurs performances (ex. gain d'information, temps d'exécution) et évoluent via des itérations successives (Wang et al., 2023; Cai et al., 2023)).

3 Expérimentation

Afin de tester le système sur des problèmes itératifs complexes avec « cas résolus » et facilement évaluables par les communautés d'extraction d'information et de TAL, nous avons choisi le cas d'usage de la synthèse scientifique (1). L'article d'état de l'art (SOTA) constitue la première expérimentation. Une étape préliminaire a consisté à créer un jeu de données d'articles scientifiques et une fonction de coût mesurant la distance par rapport à divers aspects (structure, contenu, longueur, etc.).

L'agent planificateur actuel résout une tâche demandée via des fonctions disponibles ou initie une boucle d'apprentissage si elles sont insuffisantes. Notre implémentation opérationnalise l'architecture avec quatre agents principaux (Coach, Développeur, Critique, Capitaliseur) et un mécanisme commun d'intervention humaine (HITL). Chaque agent utilise des prompts

dynamiques dédiés, une logique de code, et des moteurs d'inférence sélectionnables. Les expériences peuvent être automatiques avec optimisation bayésienne (Optuna) ou interactives via une interface utilisateur configurable. L'objectif est d'améliorer la création d'outils, les flux entre agents et humains, et l'adaptation à divers cas d'usage comme la synthèse scientifique et les systèmes décisionnels basés sur l'information.

Le système inclut les agents décrits dans la section 2 et les mécanismes HITL (retours avant/après inférence). Une interface web utilisant Monaco Editor permet l'interaction. Elasticsearch est employé pour stocker embeddings, outils, activités humaines et retours.

Nous avons créé un jeu de données d'articles arxiv, Wikipédia, et brevets européens [1], incluant titres, résumés, plans, contenus et références (10 documents par catégorie). Les sections de ces documents ont été transformées en embeddings sémantiques (OpenAI AdaV2, e5-base-v2) et structurées en JSON. Ce jeu de données et le processus de création sont disponibles [2], permettant son extension à des centaines de documents pour diversifier nos évaluations.

L'expérimentation humaine a évalué la collaboration humain-IA dans le développement de fonctions Python pour générer des articles de synthèse. Pendant 60 minutes, les participants ont interagi avec le système pour créer et valider des fonctions Python définies comme « outils », visant à transformer des documents virtuels (titre + résumé) en articles structurés. Ce processus cyclique comprenait quatre étapes : identification des tâches, développement, validation/critique, capitalisation. Les participants guidaient les agents, sélectionnaient des réponses, et régénéraient des sorties.

Les données ont été utilisées pour poursuivre l'apprentissage sans intervention humaine, tout en respectant une durée totale de 60 minutes pour comparer les niveaux d'intervention.

Avec Optuna, nous avons mené une expérimentation automatique (OursAuto) explorant différentes configurations (temps, données, itérations) en recherche systématique et optimisation bayésienne pour ajuster prompts, configurations mémoire (feedback dynamique), agents et paramètres LLM.

Les performances ont été évaluées via similarité sémantique et qualité de contenu. La similarité, mesurée avec cosinus, ratio de couverture et différences de longueur, s'appliquait aux éléments structuraux (titre, table des matières) et sections de contenu comparées aux documents cibles. Les scores Rouge-L ont également été calculés comme métrique alternative.

Pour comparaison, GPT-4o (version mini pour limiter les coûts) a été testé sur les mêmes données avec des instructions optimisées pour générer des synthèses. Les résultats montrent que notre système surpasse GPT-4o dans des métriques spécifiques, soulignant les avantages de la collaboration humain-LLM.

4 Résultats

Cette section présente les résultats de notre évaluation expérimentale, en mettant l'accent sur la manière dont la collaboration entre les agents LLM et l'implication humaine a influencé les résultats, ainsi que sur les enseignements tirés du processus d'apprentissage du système. Notre évaluation analyse davantage les dynamiques collaboratives et les interactions entre humains et LLM plutôt que de se limiter à des métriques de performance isolées. Nous décrivons

1. http://data.epo.org
2. https://tinyurl.com/37tndjmd

d'abord le contexte de ces interactions, suivi des principaux résultats montrant comment l'intervention humaine a modulé la résolution des tâches et la génération d'outils.

Notre approche a permis d'apprendre de nouvelles fonctions pour générer des tables des matières, des plans, développer chaque section, rechercher et ajouter des références pour différents articles, surpassant ainsi les documents générés avec les fonctions créées uniquement par GPT-4o en termes de longueur et de qualité équivalente. Nous avons choisi de ne pas nous concentrer sur une comparaison directe avec GPT-4o seul, car ses résultats dépendent fortement du prompt utilisé et, en outre, nous l'utilisons dans notre système. Nous mettons plutôt l'accent sur la valeur ajoutée de cette approche et sur la complémentarité des différents modes de collaboration des agents.

Nous avons évalué le système selon trois configurations : (1) **Boucle humaine complète (OursHITL)** : agents humains impliqués tout au long du processus ; (2) **Sans boucle humaine (OursAuto)** : apprentissage entièrement automatique ; (3) **Boucle humaine partielle (OursHITL puis OursAuto)** : intervention humaine suivie d'un apprentissage automatique.

Les résultats des configurations (Table 1) sont comparés via un score agrégé basé sur la distance à la solution, avec des distances sémantiques maximales calculées sur des synthèses de recherche d'Arxiv (par exemple, "Macroeconomic Effects of Inflation Targeting A Survey of the Empirical Literature"). Chaque séquence automatique (OursAuto) a été exécutée trois fois pour stabiliser les moyennes, tandis que les données des expériences humaines reposent sur 6 expérimentateurs sur 10. La Table 1 montre que si OursAuto teste efficacement un grand nombre de codes candidats, les meilleurs scores sont atteints en combinant les approches (OursHITL, puis OursAuto). Ces résultats préliminaires indiquent qu'une intervention humaine stratégique améliore considérablement les performances. Lors des expériences humaines, l'étape initiale nécessitant des corrections importantes a été identifiée comme la plus difficile en raison d'un manque d'alignement sur les objectifs et des contraintes techniques. Une fois cette phase passée, les réponses des agents LLM sont mieux alignées, ce qui explique la performance de la boucle humaine partielle. Des combinaisons supplémentaires (ex. OursAuto avant OursHITL, alternance ou parallélisation des modes) méritent d'être explorées.

Durant les tests, nous avons estimé le temps moyen investi par les humains et les LLM (via le feedback des expérimentateurs) et observé comment l'humain a appris à contribuer plus efficacement au processus. Un objectif clé est de comprendre comment répartir de manière optimale l'intervention humaine dans les différentes étapes de résolution. La Table 2 détaille les contributions humaines par type de tâche, aidant à identifier les domaines où les agents humains et LLM ont le plus d'impact.

Type de Run	**Moy.**	**Max**	**Générés**	**Sim. plan**	**Sim. contenu**
60mn OursAuto	36.8	38.8	360	0.412	0.368
60mn OursHITL	33.5	34.1	-	0.368	0.412
50mn OursHITL, 10mn auto	**46.9**	**59.2**	60	**0.567**	**0.674**
30mn OursHITL, 30mn auto	42.3	54.9	180	0.487	0.496
10mn OursAuto	29.8	33.5	60	-	-
30mn OursAuto	31.7	39.6	180	0.391	0.444

TAB. 1 – *Scores moyens/max sur 3 runs, codes candidats générés, et similarité avec cibles pour différentes allocations de temps OursAuto vs OursHITL*

Agent	Alloc. tps humain	Ratio tps LLM/Humain	Choix du LLM
Coach	15%	1ère : 50% suivante : 85%	GPT-4o
Coder	80%	1ère : 40% suivante : 95%	GPT-4o
Critic	5%	1ère/suivante : 90%	GPT-3.5
Capitalizer	~0%	1ère/suivante : 100%	GPT-3.5

TAB. 2 – *Implication des LLM et des humains pendant la phase d'apprentissage.* Agent : étape dans la boucle d'apprentissage ; Alloc. tps humain : proportion de temps (somme = 100%) à chaque étape ; Ratio tps LLM/Humain : temps LLM vs humain à la 1ère itération (Coach+Coder+Critic, puis Coder ou Capitalizer pour itérations suivantes) ; Choix du LLM pour équilibrer vitesse / perf / coût

Chaque tâche implique différentes étapes d'interaction humaine : définition initiale des tâches (agent Coach), amélioration itérative du code (agent Coder), validation, et capitalisation.

Comme illustré dans la Table 2, la répartition de l'effort humain varie selon les processus en fonction des agents, avec un effort principal sur la définition des tâches et le code lors de la première itération. Nous avons également constaté qu'un LLM plus performant comme GPT-4o par rapport à GPT-3.5 est important pour la définition des tâches et la mise en œuvre du code, mais non requis pour la validation et la capitalisation, qui sont des tâches plus simples.

5 Conclusion et travaux futurs

Nous avons présenté une architecture inspirée des frameworks Voyager et LATM pour relever les défis de la génération de synthèses techniques longues. Cette méthodologie intègre des experts humains et des modèles de langage de grande taille (LLM) dans un processus collaboratif de développement de solutions. Les expériences initiales montrent que notre système peut surpasser les LLM autonomes, tels que GPT-4o, dans la génération de contenu scientifique, en utilisant des mesures automatiques de distance sémantique et une évaluation humaine. Pour valider notre approche, nous avons créé un jeu de données comprenant des articles Arxiv, des articles Wikipédia et des brevets européens. Les résultats montrent que les agents LLM, lorsqu'ils sont guidés par une intervention humaine structurée, produisent un contenu de qualité supérieure à celui généré par GPT-4o. La validation humaine reste cruciale, notamment lorsque des exemples de solution ne sont pas disponibles pour mesurer la distance à l'objectif.

Cependant, cette étude a également mis en lumière plusieurs défis dans l'optimisation de la collaboration entre les humains et les LLM pour des tâches complexes. La stabilisation des prompts s'avère difficile, car des modifications mineures entraînent des impacts significatifs sur la cohérence et la créativité. L'efficacité de la collaboration est également affectée par la charge cognitive des agents humains, souvent confrontés à de la fatigue lors de la génération de contenu itératif ou volumineux. L'équilibre entre outils spécialisés et polyvalents reste un compromis, tandis que les exigences en ressources des modèles de grande capacité et les contraintes de mémoire limitent l'étendue des expériences.

Pour surmonter ces défis, nos travaux futurs viseront à : (1) réduire la charge cognitive et améliorer l'efficacité des interactions humains-LLM via des expérimentations HITL (Humans-in-the-Loop), tout en explorant des collaborations impliquant plusieurs rôles humains et types d'interactions ; (2) renforcer la stabilité des prompts et affiner les mécanismes des agents à tra-

vers des techniques comme l'optimisation bayésienne et *TextGrad* (Yuksekgonul et al., 2024) ; (3) intégrer un agent Planner et explorer différentes configurations de collaboration humaine ; (4) améliorer les mécanismes de mémoire et de connaissances pour l'apprentissage des outils et la résolution des problèmes ; et (5) appliquer cette approche à d'autres domaines problématiques complexes au-delà de la rédaction de synthèses scientifiques. Enfin, nous continuerons à affiner le cadre collaboratif entre humains et agents LLM en développant des stratégies avancées pour une meilleure coordination et performance.

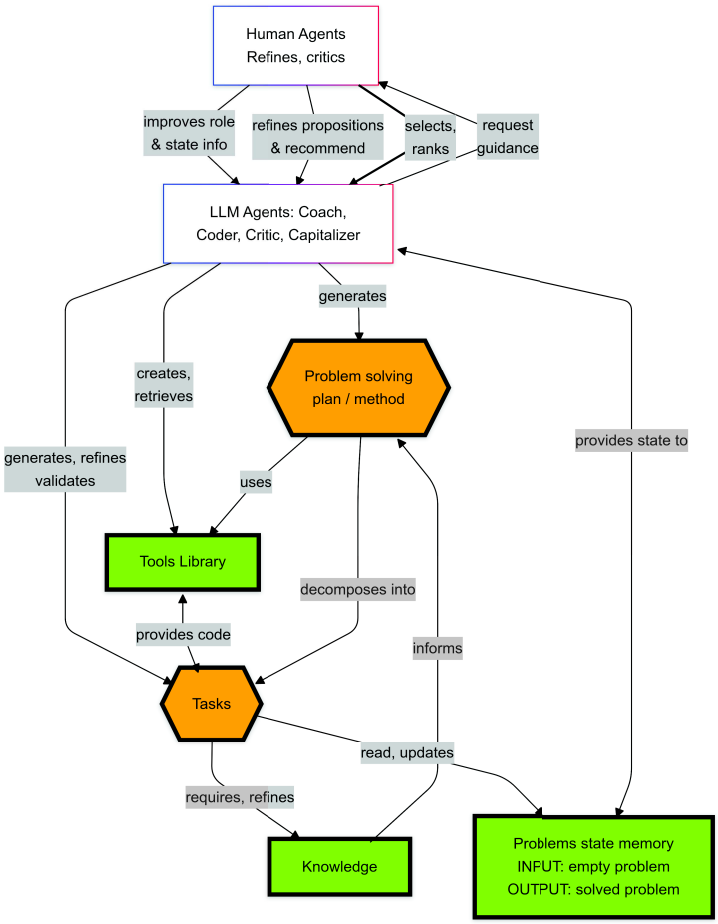

FIG. 1 – *Apprentissage de génération d'une synthèse scientifique depuis un titre et un abstract*

Références

(2024). AFlow : Automating Agentic Workflow Generation. In *The Thirteenth International Conference on Learning Representations*.

Burns (2024). Weak-to-Strong Generalization : Eliciting Strong Capabilities With Weak Supervision. In *Proceedings of the 41st International Conference on Machine Learning*.

Cai, T., X. Wang, T. Ma, X. Chen, et D. Zhou (2023). Large Language Models as Tool Makers.

Chen, P., S. Zhang, et B. Han (2024). CoMM : Collaborative Multi-Agent, Multi-Reasoning-Path Prompting for Complex Problem Solving, doi: 10.18653/v1/2024.findings-naacl.112.

Chu (2023). A Survey of Chain of Thought Reasoning : Advances, Frontiers and Future.

Colas, C., L. Teodorescu, P.-Y. Oudeyer, X. Yuan, et M.-A. Côté (2023). Augmenting Autotelic Agents with Large Language Models.

Dong, H., W. Xiong, B. Pang, H. Wang, H. Zhao, Y. Zhou, N. Jiang, D. Sahoo, C. Xiong, et T. Zhang. RLHF Workflow : From Reward Modeling to Online RLHF.

et al, R.-P. (2023). Mathematical discoveries from program search with large language models. *Nature*, 1–3, doi: 10.1038/s41586-023-06924-6.

Lu, C., C. Lu, R. T. Lange, J. Foerster, J. Clune, et D. Ha (2024). The AI Scientist : Towards Fully Automated Open-Ended Scientific Discovery.

Min, S., X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, et L. Zettlemoyer (2022). Rethinking the Role of Demonstrations : What Makes In-Context Learning Work ?

Reppert (2023). Iterated Decomposition : Improving Science Q&A by Supervising Reasoning Processes.

Sun, Z., L. Yu, Y. Shen, W. Liu, Y. Yang, S. Welleck, et C. Gan (2024). Easy-to-Hard Generalization : Scalable Alignment Beyond Human Supervision.

Wang, G., Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, et A. Anandkumar (2023). Voyager : An Open-Ended Embodied Agent with Large Language Models.

Yang, C., X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, et X. Chen (2023). Large Language Models as Optimizers.

Yao, S., J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, et Y. Cao (2022). ReAct : Synergizing Reasoning and Acting in Language Models.

Yuksekgonul, M., F. Bianchi, J. Boen, S. Liu, Z. Huang, C. Guestrin, et J. Zou (2024). TextGrad : Automatic "Differentiation" via Text.

Zhu (2023). Ghost in the Minecraft : Generally Capable Agents for Open-World Environments via LLM with Text-based Knowledge and Memory.

Summary

This demonstration presents a collaborative architecture for the iterative development of neuro-symbolic tools combining LLM agents and human intervention for scientific synthesis.