La Recherche

Dans trimestriel 584 daté janvier-mars 2026 - 2169 mots

Quand les agents LLM passent à l'action

Popularisés par ChatGPT, les grands modèles de langage (LLM) ont ouvert une ère d'interactions naturelles entre humains et machines. Une nouvelle génération, les « agents LLM », va plus loin : dotés de mémoire, capables de planifier et d'agir, ils transforment les modèles de langage en moteurs d'objectifs concrets. De la réservation d'un billet de train au devis vocal pour un artisan, les usages se multiplient. Reste à surmonter les défis techniques de la planification et de l'évaluation pour franchir un nouveau cap dans l'autonomie des IA. Panorama de la situation, qui évolue très rapidement. Avec Nathanaël Fijalkow (spécialiste de théorie des jeux et de synthèse de programmes, il mène ses travaux au Laboratoire bordelais de recherche en informatique (Labri - CNRS - université de Bordeaux - ENSEIRB) et Xavier Blanc (Professeur à l'université de Bordeaux, ce spécialiste de génie logiciel est directeur du Labri.)

En lançant ChatGPT fin 2022, la société OpenAI a popularisé une idée puissante : on peut désormais « parler » à une intelligence artificielle - un *chatbot* (agent conversationnel en français) - et obtenir une réponse textuelle crédible, souvent utile, parfois bluffante. Mais dans l'ombre de cette révolution conversationnelle, une autre évolution plus discrète est en marche. Elle ne consiste plus seulement à répondre à des requêtes - les fameux *prompts* -, mais à agir. Bienvenue dans l'univers des agents LLM, ces systèmes capables d'orchestrer le dialogue, de retenir les informations, de planifier des étapes et... d'exécuter des actions dans le monde numérique.

L'idée de doter les LLM de la capacité d'interagir avec des outils, de planifier et d'exécuter des tâches de manière plus autonome, a gagné en importance en 2024. Si le concept d'IA agentique existe depuis quelques décennies, la démocratisation et la discussion autour des « agents LLM » sont devenues particulièrement prégnantes avec l'avènement de LLM plus puissants comme GPT-3.5 et GPT-4 et des interfaces conversationnelles comme ChatGPT, apparues fin 2022 (1).

Prenons un exemple : « Peux-tu ajouter un rendez-vous à mon agenda lundi prochain à 9 heures avec Sophie, et lui envoyer une invitation ? » L'agent, contrairement au simple *chatbot*, est conçu pour exécuter cette consigne de bout en bout. Pour ce faire, il repose sur quatre composantes clés qui transforment le modèle de langage en rouage d'un système plus large : un profil, qui définit l'identité de l'agent, la manière dont il interagit avec l'utilisateur et la façon dont il exploite le modèle de langage ; une mémoire, qui conserve les informations utiles à la mission au-delà de la capacité contextuelle limitée du LLM ; un plan, qui formalise l'objectif poursuivi et les étapes nécessaires pour l'atteindre ; enfin, des actions qui permettent à l'agent d'interagir avec son environnement, par exemple consulter un agenda, remplir un formulaire, interroger un moteur de recherche... ou invoquer un LLM.

Détaillons ces composantes à travers quelques exemples pour apprécier leur importance. Dès les premières expérimentations avec les LLM - ChatGPT et Gemini - en 2023, on s'est rendu compte de l'importance d'expliquer au modèle qui il est. Ainsi, si vous souhaitez obtenir des informations sur la gestation des bonobos, il est conseillé de commencer la requête par : « Vous êtes un scientifique spécialisé dans l'étude des primates. » Cette phrase, qu'on appelle un *prompt* d'initialisation, permet au modèle de mobiliser des connaissances spécifiques et d'adopter le ton approprié. C'est l'idée du profil, la première composante.

La limitation de la mémoire est inhérente aux modèles de langage. Lorsque l'échange entre un utilisateur et le LLM s'allonge, on s'aperçoit que les messages les plus anciens sont oubliés. La raison est que le LLM dispose d'une taille de contexte maximale. Bien que cette taille ait énormément augmenté depuis les premiers modèles (par exemple, GPT-3 utilise 2 048 jetons (*token*), soit environ 1 500 mots, alors que Gemini 1.5 Pro dépasse un million de jetons, pas loin d'un million de mots), elle reste bornée. D'où l'importance de la deuxième composante d'un agent, la mémoire, qui consiste à externaliser les informations importantes, et donc de ne pas la limiter *a posteriori*. Ce n'est pas seulement une question de longueur des messages : le LLM peut également lire des documents extérieurs (par exemple des fichiers au format PDF). Si ces documents sont trop longs, le modèle ne saura pas fidèlement restituer les informations pertinentes. La mémoire externe de l'agent permet de conserver le document et d'y accéder de manière plus robuste, sans perte d'informations. Autrement dit, l'agent ne se contente plus de répondre : il se « souvient ».

La troisième composante, cruciale, est la planification. Là où un LLM pur agit au coup par coup, sans vision d'ensemble, l'agent est capable de décomposer une tâche complexe en sous-tâches. Surtout, il s'agit d'évaluer les conséquences de la réalisation de ces sous-tâches et d'adapter la planification de manière adéquate. Dans un agent LLM, cette troisième composante peut être gouvernée par un algorithme de planification ou directement fournie par un modèle de langage.

PAS UNE ENTITÉ AUTONOME, MAIS UN CADRE CONCEPTUEL

En dernier lieu, un agent n'est pas tout-puissant : il est limité par l'ensemble des actions qu'il peut effectuer, et qui sont précisément décrites dans sa dernière composante. Cette capacité d'action repose sur un répertoire d'actions. Par exemple, l'agent peut demander à une calculatrice de faire un calcul, d'effectuer une recherche sur un moteur de recherche, ou même... d'écrire une requête pour un LLM ! En effet, le LLM est un outil parmi d'autres, à la disposition de l'agent pour la réalisation de la tâche décrite par l'utilisateur.

La description de ces quatre composantes met en lumière un point essentiel : un agent LLM n'est pas une entité autonome, mais un cadre conceptuel. Il formalise l'interaction entre un utilisateur et un algorithme en la structurant autour d'un objectif à atteindre, de moyens limités et de contraintes explicites. Ce n'est donc pas une intelligence libre d'improviser, mais un système délimité par les briques qui le composent.

L'agent est ainsi défini autant par ce qu'il peut faire que par ce qu'il ne peut pas faire. Sans mémoire robuste, il perd la trace des échanges passés et devient incapable de reconnaître des situations déjà rencontrées. De même, si ses actions sont limitées à de simples clics ou à la navigation passive sur une page web, il sera dans l'incapacité de remplir un formulaire, de transmettre des informations ou d'interagir de manière fine avec un système extérieur.

Qu'est-ce qui fait la force des agents LLM ? C'est leur capacité à exploiter pleinement les aptitudes des modèles de langage, en particulier leur maîtrise du langage naturel. Cette compétence ouvre la voie à des interactions plus intuitives, plus souples, entre l'humain et la machine : formuler une intention en langage courant devient suffisant pour déclencher des actions complexes. Au-delà de cette dimension technique, l'agent LLM propose surtout un nouveau cadre de pensée. Il invite à concevoir l'intelligence artificielle non plus comme une boîte noire qui répond mais comme un système structuré, capable de dialoguer, de planifier, de mémoriser et d'agir. Une manière de formaliser - et peut-être de maîtriser - la complexité croissante des rapports entre les utilisateurs et les machines.

En matière d'applications concrètes des agents LLM, la start-up Boby, située à Bordeaux, illustre bien le potentiel de ces outils. Spécialisée dans les solutions de facturation pour les artisans du bâtiment, l'entreprise propose une interface simple afin de créer devis et factures, dans un secteur où la diversité des matériaux et des références rend l'exercice souvent fastidieux. Pour alléger cette charge, Boby a intégré un agent LLM, capable d'interpréter des consignes dictées en langage naturel. L'utilisateur peut ainsi prononcer une requête comme : « Boby, peux-tu me créer un devis pour Mme Floyd avec un évier blanc ? » Immédiatement, l'agent déclenche les actions nécessaires : ouverture du devis, insertion du bon produit, génération du document. Le tout de manière visible et en temps réel, sur l'écran de l'utilisateur. L'assistant vocal devient alors un véritable opérateur numérique, traduisant une instruction verbale en une suite d'interactions complexes.

L'automatisation des tâches sur le Web est un des champs d'applications les plus dynamiques pour les agents LLM. L'idée est simple en apparence, ambitieuse en réalité : permettre à un agent d'exécuter une action décrite en langage naturel, sans passer par une interface spécifique, comme réserver un billet de train, remplir un panier d'achat ou interagir avec un formulaire - le tout à partir d'une phrase du type : « Peux-tu me réserver un train de Bordeaux à Paris le 11 octobre au matin ? » ou « Mets dans mon panier le même tee-shirt que sur cette photo. » Afin de mesurer les avancées dans ce champ encore balbutiant, plusieurs plateformes de test ont vu le jour, dont Visual Web Arena, qui propose des centaines de scénarios standardisés. Les meilleurs agents parviennent désormais à réaliser plus de 60 % des tâches définies, signe d'un potentiel réel mais aussi d'une grande marge de progression.

Parmi les projets prometteurs d'automatisation de tâches sur le Web, l'agent Mind2Web, proposé par des chercheurs en informatique de l'université d'État de l'Ohio (États-Unis) en 2023, se distingue par son approche modulaire et sa transparence : il est en accès libre sur la plateforme de partage GitHub, permettant aux chercheurs de l'analyser, le tester et l'améliorer. De multiples équipes dans le monde entier se sont emparées du sujet, publiant à un rythme soutenu des résultats sur l'interprétation d'intentions, la navigation visuelle ou encore la détection d'éléments sur des interfaces complexes. Par exemple, l'entreprise Skyvern commercialise un agent LLM pour le Web dont les performances sur la base de tests *Web Bench* les placent parmi les meilleurs.

VOYAGER ET SON LANGAGE INTERNE REMARQUABLE

À l'opposé des environnements balisés du Web ou des logiciels métiers, le jeu populaire Minecraft offre un terrain ouvert où l'utilisateur est libre de choisir ses objectifs : construire un abri, explorer des grottes ou fabriquer des outils. C'est dans cet univers que Voyager, un agent LLM expérimental, mis au point en 2023 par un consortium d'universitaires et d'industriels (2), montre toute l'ampleur de ce que peut devenir une intelligence artificielle

capable d'acquérir seule de nouvelles compétences. Concrètement, Voyager ne se contente pas de suivre des instructions : il explore, expérimente, progresse. Il repère de nouvelles opportunités, s'exerce à de nouvelles compétences, évalue ses résultats... et adapte sa stratégie. Cette boucle d'apprentissage implique une planification autonome : décomposer une mission, tester des actions, corriger ses erreurs. Mais ce qui rend cet agent particulièrement remarquable, c'est son langage interne. Plutôt que d'échanger en langage naturel, les différentes composantes de l'agent communiquent en s'envoyant du code informatique. Plus précisément, il s'agit de programmes Python, que le LLM génère, modifie, exécute et corrige. Cette approche présente deux avantages majeurs : elle permet de produire des instructions non ambiguës, et elle introduit des boucles de rétroaction rigoureuses, indispensables à l'apprentissage progressif de nouvelles tâches. Avec Voyager, l'agent LLM devient un système de raisonnement actif, capable d'élargir son champ d'action au fil de l'expérience. Un pas de plus vers des intelligences artificielles capables d'adaptation - dans un monde certes virtuel, pour l'instant.

UNE EFFICACITÉ FREINÉE PAR TROIS LIMITES MAJEURES

Si les agents LLM ouvrent des perspectives prometteuses, leur efficacité reste freinée par trois limites majeures : la planification, la capacité à agir sur l'environnement et l'évaluation. Autant de domaines où les progrès sont nécessaires afin de franchir un véritable cap en autonomie.

En termes de planification, la plupart des agents fonctionnent encore sur un mode très réactif, structuré autour d'une boucle en trois étapes : choisir l'action la plus probable, l'exécuter, en évaluer l'effet... et recommencer. Une stratégie certes fonctionnelle, mais myope : l'agent agit au coup par coup, sans anticiper les étapes suivantes, ni bâtir de stratégie globale. Certaines avancées récentes, notamment dans les nouveaux modèles comme GPT-5, intègrent des mécanismes de planification plus explicites (3).

Deuxième limite : la capacité à agir sur l'environnement. Les agents disposent généralement d'une liste d'actions prédéfinies - cliquer, écrire, copier, soumettre -, décrites sous forme de fonctions avec des paramètres. Le défi est alors d'aligner l'intention exprimée en langage naturel avec l'action disponible : nous avons tous été confrontés à la frustration de ne pas trouver une information sur un site web parce qu'elle était cachée derrière un mot-clé ou une catégorie inattendue.

Enfin, l'évaluation de l'impact des actions demeure rudimentaire. L'agent doit s'en remettre à son LLM pour analyser son environnement - une capture d'écran, un extrait de code HTML (le langage natif de description des pages web) - et en déduire si la tâche a bien été accomplie. Dans la réservation d'un billet de train, il faut que l'agent comprenne qu'il ne suffit pas de cliquer sur le bouton de réservation mais qu'il faut répondre à toutes les questions suivantes (choisir les sièges, accepter ou renoncer à l'assurance, etc.) afin de finaliser la tâche, et cela ne peut se faire que dynamiquement, en observant les écrans qui s'enchaînent. Des travaux récents cherchent à y remédier en introduisant des approches formelles plus rigoureuses, inspirées des méthodes de vérification logicielle. Par exemple, les travaux de chercheurs en informatique de l'université du Wisconsin à Madison (États-Unis) introduisent des techniques de vérification de modèles (*model checking*) pour évaluer la qualité des planifications suggérées par le LLM (4). La vérification de modèle apporte ici une rigueur mathématique nécessaire : plutôt que de se contenter d'intuitions textuelles, on vérifie que la planification

couvre bien toutes les étapes obligatoires d'un processus. Cela permet d'éviter des erreurs logiques dans des tâches critiques (réservations, formulaires, etc.).

La promesse des agents LLM est immense : passer de l'assistance passive à l'action autonome, dans des interfaces toujours plus naturelles. Mais cette ambition se heurte à de nombreux obstacles, comme nous l'avons décrit. L'avenir des agents LLM ne dépend donc pas seulement des progrès des modèles eux-mêmes, mais aussi de l'intégration fine d'architectures logicielles capables de structurer leur intelligence. En somme, il ne s'agit plus de faire parler les machines, mais de leur apprendre à décider. Et si leurs promesses sont grandes, qu'en est-il de leur éthique ? Les agents LLM ne répondent absolument pas à ces questions soulevées par les LLM, voire ils en soulèvent de nouvelles, essentiellement parce que l'utilisateur délègue à l'agent une capacité d'action, laquelle n'est, à ce jour, malheureusement pas contrôlée (lire p. 107)

(1) L. Wang *et al.*, *Front. Comput. Sci., 18*, 186345, 2024.

(2) voyager.minedojo.org (en anglais)

(3) openai.com/fr-FR/gpt-5

(4) Ch. P. Lee *et al., Proceedings of the 2025 Conference on Human Factors in Computing Systems, 247*, 1, 2025.

RETOUR AU SOMMAIRE