Grands Modèles de Langage (Large Language Models) et Règles Logiques pour une Automatisation Décisionnelle Avancée

P.Feillet[1]

[1] IBM France Lab

feillet@fr.ibm.com

Résumé

Les Grands Modèles de Langage (Large Language Models) impactent tous les secteurs d'activité en promettant une nouvelle expansion de l'automatisation dans tous les métiers. L'enjeu pour les organisations est de bénéficier des gains de productivité annoncés par l'IA générative tout en sécurisant le raisonnement algorithmique dans la prise de leurs décisions critiques.

Il est opportun d'évaluer les qualités des LLMs et des moteurs d'inférence de règles logiques ainsi que leur hybridation pour préparer une meilleure automatisation des décisions.

Mots-clés

IA hybride, Règles Logiques, Large Language Models, IA Générative, Extraction de connaissances

Abstract

Large Language Models (LLMs) are impacting all sectors of activity by promising a new expansion for automation. The challenge for organizations is to benefit from the productivity gains announced by generative AI while securing algorithmic reasoning in critical decision-making.

It is timely to evaluate the qualities of LLMs and rule inference engines and prepare their hybridization for improved decision-making automation.

Keywords

Hybrid AI, Rules, Logic, Large Language Models, Generative AI, Knowledge extraction

1 Introduction

L'intelligence artificielle générative disrupte le monde de l'IA en ouvrant de nouvelles perspectives d'interactions Humain Machine et repousse les limites des tests de Turing.

Étant donnés les progrès récents une question se pose : les LLMs seuls peuvent-ils suffire à automatiser les décisions critiques dans nos organisations ? Sont-ils compétents pour raisonner avec confiance afin de prendre une décision à fort impact pour les entreprises et citoyens ?

2 Les Grands Modèles de Langage

2.1 Définition générale

En résumé, un grand modèle de langage compresse les informations lues à partir d'un corpus de textes d'entraînement pour générer de nouveaux textes à partir d'une requête donnée. Basé sur une architecture de réseau de neurones, son comportement est statistique. Il prend une séquence de « tokens » (groupes de caractères) exprimés dans la requête et produit une autre séquence de « tokens » qui sont les plus probables en regard de sa base documentaire d'entraînement. Ce n'est pas dans sa construction un algorithme de raisonnement ; il ne repose pas sur des mécanismes logiques et peut être vu comme un perroquet stochastique [11].

Les modèles de langage à grande échelle (LLMs) se sont multipliés depuis 2022, disponible en source fermé ou ouvert et disponible en « Model as a Service » ou localement [8].

2.2 Apprentissage par renforcement à partir de rétroaction humaine

Au-delà de leur apprentissage auto-supervisé utilisant des corpus textuels, certains LLMs bénéficient d'un apprentissage par renforcement par rétroaction humaine (Reinforcement Learning from Human Feedback). Cette approche d'apprentissage par renforcement repose sur les commentaires et évaluations humaines pour guider l'apprentissage du modèle sur la base d'une évaluation humaine de ses résultats.

2.3 Chaine de pensée

Le "Chain-of-thought prompting" (en français, requête par chaîne de pensée) est une méthode qui vise à améliorer les capacités de raisonnement des grands modèles de langage (LLM). Cette technique fonctionne en décomposant la résolution d'un problème en modélisant les étapes successives d'un processus de pensée.

Cette approche popularisée par LangChain [7] permet au

modèle non seulement d'arriver à des réponses plus précises, mais également de développer une méthode de résolution de problèmes plus structurée et explicative, rendant les réponses générées plus compréhensibles pour les utilisateurs. Cette décomposition garde néanmoins des limites en termes de raisonnement.

2.4 « Retrieval-Augmented Generation »

Le RAG pour "Retrieval-Augmented Generation", est une technique conçue pour améliorer les réponses du modèle en intégrant des informations extraites de documents externes. Ce processus se décompose en deux étapes principales :

- Récupération (Retrieval) : Dans cette première étape, le modèle utilise une requête (généralement basée sur la question ou la tâche posée par l'utilisateur) pour chercher et récupérer des informations pertinentes à partir d'une base de données de documents. Cette base de données peut être constituée de textes provenant de diverses sources telles que des livres, des articles ou des sites internet.
- Génération (Augmented Generation) : Ensuite, le modèle utilise les informations récupérées comme contexte supplémentaire pour générer une réponse. Ce faisant, le modèle peut fournir des réponses plus précises, informatives et basées sur des données concrètes, plutôt que de se baser uniquement sur ce qu'il a appris durant son entraînement.

L'approche RAG est particulièrement utile pour les tâches qui nécessitent des réponses détaillées ou des informations spécifiques qui ne sont pas nécessairement contenues dans le corpus d'entrainement du modèle. Elle permet aux modèles de langage d'enrichir les requêtes afin de rendre la génération plus performante. Toutefois elle n'influe pas sur les qualités intrinsèques de raisonnement du LLM.

3 Les moteurs d'inférence de règles

Depuis plusieurs décennies, les entreprises s'appuient sur des solutions logicielles pour automatiser leurs décisions critiques dans le but de déterminer des éligibilités règlementaires, approuver des prêts et souscrire à des offres de services. Au-delà des applications encodant en Java, Javascript ou autre langage, la logique métier, les systèmes de gestion de règles ont été adoptés par les industriels, notamment par les services financiers. Ces systèmes permettent le développement, le test, la simulation, le déploiement et la maintenance de politiques métier en s'appuyant sur des règles logiques. Cette approche repose sur la capture de connaissances et leur formalisation. Les experts métier définissent une ontologie avec des prédicats logiques et des structures de données représentant leur domaine. Ces règles, communément formulées sous forme d'instructions Si <conditions> Alors <Actions>, d'arbres ou tables de décision, sont évalués par un moteur d'inférence causale capable d'instancier et de chaîner l'exécution dans un contexte de données structurées, automatisant ainsi un chemin de raisonnement.

Cette technologie a mûri au fil des ans, permettant de capturer des modèles de décision complexes avec des milliers de règles et de tables de décision. Des produits comme IBM Operational Decision Manager (ODM) [3] ont prouvé leur capacité à capturer des politiques complexes et exécuter plus d'un milliard de décisions quotidiennement tout en prenant quelques millisecondes pour réaliser chaque décision sur une CPU.

Les moteurs de règles comme ceux d'IBM ODM et ADS permettent de prescrire un micro-flot de raisonnement, afin par exemple de procéder à la validation d'un dossier d'emprunt, puis au calcul d'éligibilité, et enfin de calculer les modalités de remboursements et d'assurance en fonction du profil, de la politique en cours et des taux bancaires. Ces micro-flots peuvent être des DAG (Direct Acyclic Graph) ou bien autoriser des cycles dans des flots de regles (ruleflows). Ils organisent les étapes de raisonnements, et évaluent dans chacune des taches rencontrées un ensemble de règles.

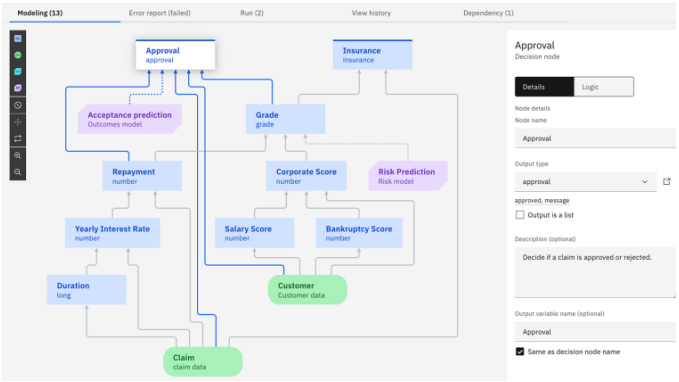

Figure 1: Un modèle de décision défini comme un graphe acyclique dirigé dans IBM ADS

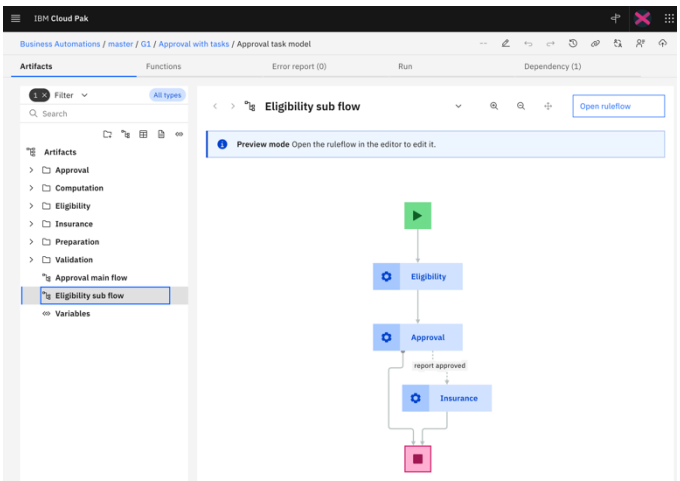

Figure 2: Un micro-flow appelé Ruleflow pouvant contenir un cycle dans IBM ADS

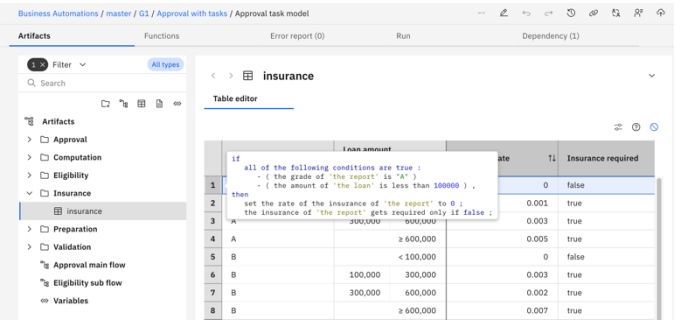

Figure 3: Une table de décision regroupant des règles dans IBM ADS

4 Les exigences d'automatisation décisionnelle

4.1 Quelles sont les exigences clés pour la prise de décision en entreprise ?

L'automatisation des décisions en entreprise implique l'utilisation de la technologie et des logiciels pour systématiser les processus de prise de décision au sein d'une organisation. Les critères clés pour une automatisation réussie des décisions en entreprise sont les suivants :

- Exactitude et fiabilité : le système doit être précis dans le traitement des données et la prise de décisions pour assurer la confiance des utilisateurs et des parties prenantes.
- Évolutivité : la solution doit être capable de gérer de grands volumes de données et de demandes de prise de décision sans dégradation significative des performances.
- Flexibilité et adaptabilité : les environnements économiques et règlementaires sont dynamiques, et les exigences en matière de prise de décision peuvent changer. Le système doit être suffisamment flexible pour s'adapter à de nouvelles règles commerciales, politiques et réglementations sans nécessiter de coûts majeurs de développement.
- Prise de décision en temps réel : dans certains scénarios, des capacités de prise de décision en temps réel ou quasi-temps réel sont cruciales. Le système doit traiter les données et fournir des décisions dans des délais acceptables.
- Transparence et auditabilité : les décisions d'entreprise impactent souvent des processus critiques, et les parties prenantes ont besoin de comprendre comment les décisions sont prises. Le système doit fournir des explications claires pour les décisions, et il doit être auditable à des fins de conformité réglementaire.
- Sécurité et confidentialité des données : puisque l'automatisation des décisions traite de données sensibles, la sécurité et la confidentialité des données sont primordiales. Le système doit employer des mesures de sécurité robustes pour protéger les données contre des accès ou manipulations non autorisés.
- Observations des performances : le système doit disposer de capacités de mesures et de rapport pour suivre les performances des processus de prise de décision et identifier les domaines à améliorer.
- Maitrise des coûts : la considération du coût du système par rapport à ses avantages est essentielle. La solution doit offrir un bon retour sur investissement et s'aligner sur les contraintes budgétaires de l'organisation.

Dans l'ensemble, une solution réussie d'automatisation des décisions en entreprise doit s'aligner sur les objectifs de l'organisation, rationaliser les processus de prise de décision et contribuer à une efficacité et une productivité accrue.

4.2 Les LLMs atteignent-ils seuls toutes ces exigences ?

Bien que certains LLMs montrent des résultats impressionnants et certaines capacités de raisonnement, ils échouent tout aussi facilement lors de la répétition de l'expérience, ou lors de légères modifications dans le requêtage. Vous pouvez expérimenter ce comportement à double face avec le « chatbot » de commande de pizza présenté dans le tutoriel OpenAI de DeepLearning.ai [1]. Selon les essais, le chatbot fournit le résultat attendu ou un résultat surprenant, même avec seulement un léger changement de requêtes.

Un autre défi avec les LLMs est leur limite maximale de « tokens », qui restreint la quantité de contexte qu'ils peuvent gérer. Une technique pour résoudre ce problème est d'appliquer une approche « Retrieve Augment Generate » [10] ou d'étendre l'apprentissage du LLM par fine-tuning sur un corpus complémentaire.

Par ailleurs, l'apprentissage par renforcement avec retour humain appliqué aux LLMs pour en atténuer leurs réponses purement statistiques peut s'avérer incomplet voir contre-productif dans certaines tâches [9].

4.3 Pouvons-nous les utiliser en combinaison avec des moteurs de décision basés sur des règles ?

Les moteurs d'inférence de règles permettent l'automatisation de prises de décisions. Ils reposent sur la capture d'un savoir-faire métier dans une ontologie et la spécification non ambiguë de la logique formalisant l'expertise sur des données structurées. Cela garantit une prise de décision déterministe, robuste et transparente en échange d'un effort de capture et de formalisation de la connaissance.

Le défi réside dans la fusion de ces technologies pour capitaliser sur leurs points forts, à l'instar de la création de matériaux composites qui surpassent les propriétés individuelles de chaque élément.

Nous proposons différentes méthodes combinant les LLMs avec des moteurs de règles.

5 Hybridations LLM et règles logiques

5.1 Compréhension du langage naturel suivie par un raisonnement basé sur des règles

Dans cette approche, nous utilisons d'abord un modèle de langage basé sur l'apprentissage automatique (LLM) pour comprendre le texte brut et en extraire des données structurées. Ensuite, nous passons ces données structurées extraites à un moteur de règles logiques pour raisonner de manière déterministe sur ces données, potentiellement combinées avec des informations supplémentaires provenant de référentiels d'entreprise.

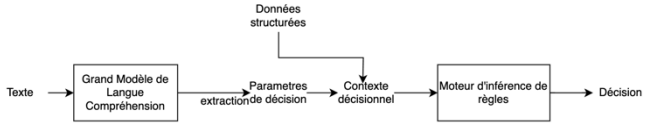

Figure 4: Compréhension du langage naturel suivie par un raisonnement basé sur des règles logiques

Avantages : L'intégration séquentielle d'un LLM dédie à la compréhension du langage naturel afin d'en extraire des entités structurées suivie par l'exécution d'un moteur de règles est simple à mettre en place. Le pipeline LLM-Moteur de règles implique de transmettre les résultats du LLM comme paramètres d'entrée pour alimenter le contexte du moteur de règles.

Inconvénients : La performance de l'ensemble dépend de l'aptitude du LLM à extraire correctement les données structurées. Pour se protéger des cas ou celles-ci s'avèrent incomplètes ou mal formatées, il est crucial de mettre en place des garde-fous avec des vérifications, des valeurs par défaut ou des heuristiques de repli pour alimenter la phase de raisonnement. L'extraction du contexte décisionnel doit se révélée robuste via une ou plusieurs appels au LLM, et éventuellement par une chaine de pensée. Elle peut s'appuyer sur des interactions utilisateur dans le cadre d'une expérience conversationnelle.

5.2 Raisonnement basé sur des règles suivi par la génération de langage naturel avec un LLM

Dans cette intégration, un moteur de règles prend d'abord une décision basée sur des données structurées. La décision structurée est ensuite transmise à un modèle de langage qui en fait déduit une transformation dans un texte de langue comme un résume ou un courriel.

Cette hybridation est particulièrement utile pour fournir des explications, justifications, résumer les décisions d'entreprise et les communiquer. Le contenu généré peut être adapté en fonction de du profil du destinataire et la communication visée.

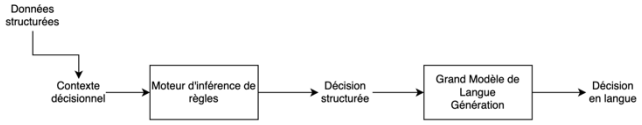

Figure 5: Raisonnement basé sur des règles suivi par une génération de texte en langage naturel avec un LLM

Avantages : Cette approche offre l'avantage de tirer parti des puissantes capacités de génération de langage naturel (NLG) fournies par le LLM pour créer des textes bien formulés pour communiquer des décisions automatisées.

Elle peut être facilement mise en œuvre facilement via une ingénierie de requêtage « prompt engineering » en passant la décision structurée au LLM.

Inconvénients : Alors que les règles se concentrent sur le raisonnement avec des données structurées, le LLM se concentre sur les tâches de traitement du langage naturel (NLP). Cette séparation nécessite une mise au point minutieuse pour gérer les variations de génération tant dans leurs formats que dans leurs contenus. Des tests automatisés du résultat de génération de langage naturel (NLG) sont nécessaire pour mesurer la performance de cette hybridation ainsi que la couverture des erreurs et variations.

5.3 Raisonnement basé sur des règles pilotant le traitement du langage naturel avec le LLM

Le moteur d'inférence de règles agit ici comme le moteur principal, invoquant le modèle de langage à la demande. Le moteur de règles pilote l'évaluation logique et appelle dynamiquement le LLM pour déléguer deux tâches :

- Traitement du texte reçu dans son contexte de décision pour la compréhension (NLU) et l'extraction de données structurées.
- Générer du texte (NLG) pour produire, par exemple, un résumé de la décision automatisée en texte brut.

Cette intégration se poursuit dans la continuité de l'appel de tout modèle d'apprentissage automatique depuis IBM ODM [3] ou IBM ADS [2] [12], pour considérer les probabilités de risque ou d'opportunité lors d'une prise de décision. De même, le LLM peut être appelé à partir d'une règle, soit à distance, soit localement, en fonction de son facteur de forme.

Il est possible d'envisager cette hybridation comme une implémentation de chaines de pensée par un moteur logique, en considérant que le LLM est appelé à chaque étape ou le traitement du langage intervient.

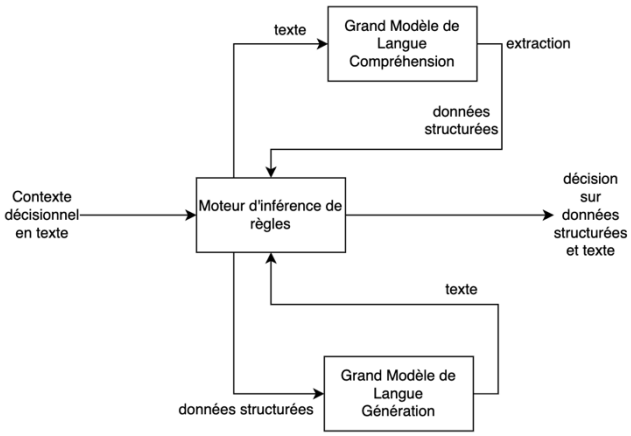

Figure 6: Raisonnement basé sur des règles pilotant le traitement du langage naturel avec le LLM

Avantages : Le contrôle est assuré par le moteur de règles permet d'orchestrer les étapes de raisonnement, et procéder par déduction. Les LLMs sont appelés à la demande en fonction du chemin de raisonnement pour réaliser les tâches de NLP.

Inconvénients : Les moteurs de raisonnement et LLMs sont étroitement couplés, nécessitant une intégration fine. Les données structurées utilisées dans la prise de décision doivent s'aligner avec les tâches de NLP. Des garde-fous appropriés sont nécessaires pour gérer la frontière entre données structurées et non structurées pour garantir la qualité et fiabilité de l'hybridation.

5.4 Extraire des règles métier depuis du texte avec un LLM pour les exécuter dans un moteur logique

Cette approche utilise un modèle de langage pour extraire des éléments d'automatisation, y compris des règles logiques, des modèles de données et des signatures fonctionnelles, à partir de politiques d'entrepris exprimées en texte brut. Ces éléments extraits sont ensuite utilisés pour générer un projet d'automatisation dans un moteur de règles comme ceux d'IBM ADS ou ODM. Cette approche a déjà été prototypée avec succès avec IBM ADS en passant par la génération d'un descripteur pivot en JSON.

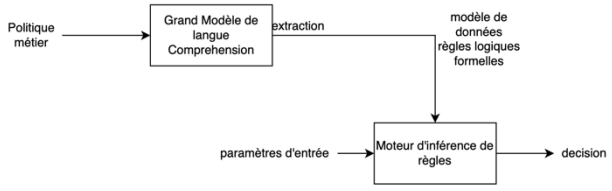

Figure 7: Extraire des règles logiques du texte avec un LLM et exécuter ces règles dans un moteur d'inférence

Avantages : Tirer parti du LLM comme outil d'extraction de connaissances permet d'extraire des règles et une ontologie de données sous-jacente. Les règles extraites, une fois révisées par un humain, permettent d'automatiser des décisions avec traçabilité et déterminisme. C'est un chemin prometteur pour assister l'extraction de connaissances expertes et capitaliser sur des moteurs d'inférence pour l'exécution des décisions.

Inconvénients : Cette approche nécessite une chaîne de requêtes efficaces, éventuellement un modèle réglé finement pour réaliser des extractions pertinentes de la logique, pour tout domaine d'expertise et formulation de la politique métier. Elle nécessite également des compétences et outils pour extraire efficacement la connaissance à automatiser, et en assurer la maintenance lorsque les documents sources évoluent ou sur la base de retours opérationnels.

5.5 Règles pour apporter un raisonnement fiable dans un agent conversationnel

Un grand modèle de langage (LLM) est ici utilisé pour piloter l'expérience conversationnelle, gérant les tâches de traitement du langage naturel (NLP). Le LLM délègue à un moteur de décision basé sur des règles pour appliquer des décisions logiques.
Cette intégration exige que le chatbot reconnaisse, pendant la conversation, quand déclencher un service de décision. Le chatbot guide le dialogue pour fournir le contexte et invoque le moteur de décision basé sur des règles lorsque tous les paramètres d'entrée sont définis. Le moteur de décision renvoie des paramètres de sortie, qui sont ensuite restitués dans la conversation par la génération de langage naturel (NLG).
IBM travaille activement à incorporer ce schéma pour apporter des capacités de prise de décision à Watson Orchestrate [6]. Par ailleurs les clients peuvent déjà développer des outils dans des solutions open-source comme LangChain pour invoquer des décisions basées sur des règles à partir d'un chatbot [4].

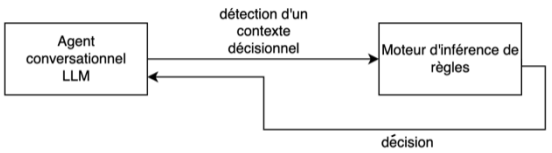

Figure 8: Décision à base de règles logiques injectée pour raisonner dans un agent conversationnel

Avantages : Cette approche permet de bénéficier d'une expérience utilisateur conversationnelle tout en déléguant le raisonnement à des moteurs déterministes spécialisés.

Inconvénients : La mise en œuvre de cette approche nécessite une détection performante des décisions à déléguer, de synchroniser un contexte entre les 2 moteurs, et de gérer les cas d'erreur à la frontière entre les domaines de données structurées et non structurées. Les défis peuvent inclure le traitement de différents formats de données et des informations de contexte incomplètes.

6 Conclusion

Nous avons exploré le pouvoir de transformation des modèles de langage (LLMs) dans l'automatisation des décisions d'entreprise. Les LLMs offrent des capacités impressionnantes dans le traitement du langage. En étant combinés à un apprentissage par renforcement à partir de rétroaction humaine, à des chaines de pensées ou du RAG ils améliorent leurs performances.
Ils manquent néanmoins de compétences pour un raisonnement fiable et répétable afin de répondre aux exigences strictes associées à la prise de décision critique. Pour combler cette lacune, nous avons introduit cinq approches d'hybridation des LLMs avec des moteurs de raisonnement basés sur des règles logiques.

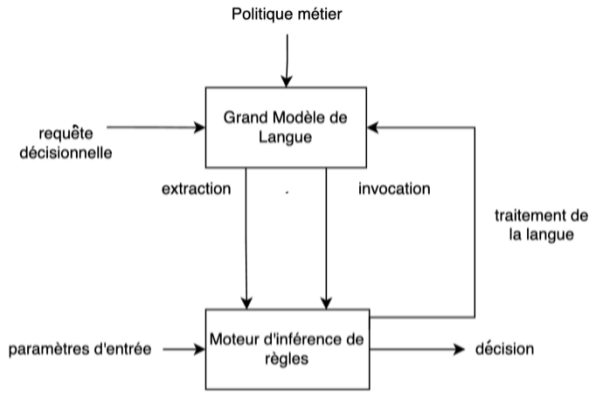

Figure 9: Récapitulatif des combinaisons LLM et règles logiques

Ces motifs d'intégration incluent l'invocation de LLM avant ou pendant l'exécution des règles, pour le traitement du texte brut en complément des données structurées, des règles pilotant le traitement du texte, l'extraction par LLM de règles logiques à partir de documents, et l'utilisation de services de décision basés sur des règles dans des agents conversationnels.

En combinant l'IA générative et symbolique, ces hybridations d'IA visent à promouvoir de nouveaux usages et démocratiser les solutions d'automatisation de prise de décision tout en satisfaisant les enjeux règlementaires et éthiques des entreprises.

7 Références

[1] D e e p L e a r n i n g OpenAI courses : https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/

[2] IBM ADS : https://www.ibm.com/products/automation-decision-services

[3] IBM Operational Decision manager : https://www.ibm.com/products/operational-decision-manager
[4] IBM ODM with LangChain : https://community.ibm.com/community/user/automation/blogs/laurent-grateau1/2023/06/09/integrating-odm-with-large-language-model

[5] IBM watsonx : https://www.ibm.com/watsonx
[6] IBM watsonx Orchestrate : https://www.ibm.com/products/watson-orchestrate
[7] LangChain : https://arxiv.org/html/2402.06196v1
[8] Large Language Models: A survey : https://arxiv.org/html/2402.06196v1
[9] Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback : https://arxiv.org/abs/2307.15217

[10] Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks : https://arxiv.org/abs/2005.11401

[11] Stochastic parrot : https://en.wikipedia.org/wiki/Stochastic_parrot

[12] Two ways of integrating Machine Learning in IBM ADS : https://www.youtube.com/watch?v=7ZJqt5bexS8