Répondre aux questions complexes : limites des LLM et solutions hybrides

Xavier Daull[1,3], Patrice Bellot[2], Emmanuel Bruno[1], Vincent Martin[3] and Elisabeth Murisasco[1]

[1]*LIS UMR 7020 CNRS / AMU / UTLN, Université de Toulon Campus de La Garde, 83041 Toulon Cedex 9, France*
[2]*LIS UMR 7020 CNRS / AMU / UTLN, Aix Marseille Université Campus de Saint Jérôme, 13 3997 Marseille Cedex 20, France*
[3]*Naval Group, 199 Av. Pierre-Gilles de Gennes, 83190 Ollioules*

Abstract

Nous étudions les besoins et limites des grands modèles de langage (LLM) pour répondre à des questions complexes ("quelle est la meilleure solution et les alternatives possibles pour un problème écologique, économique ou industriel ?"), et présentons les solutions d'hybridation pouvant y remédier. Nous proposons une définition des questions complexes dans le cadre des LLM, identifions les principaux besoins du CQA (Complex Question Answering) pour répondre et les principales limites associées des LLM, telles que les besoins en raisonnement avancé, la compréhension du contexte et des concepts, l'alignement aux attentes, la prévention des hallucinations. Face à ces limites, nous proposons des stratégies d'hybridation des LLM qui intègrent des approches complémentaires pour y pallier et améliorer leur capacité de réponse à des questions complexes.

Keywords

Modèles de langue, LLM, Questions complexes, Hybridation, Augmentation

1. Introduction

Les grands modèles de langage (LLM) ont démontré des capacités remarquables dans diverses tâches de compréhension et de génération du langage naturel. Leur potentiel pour répondre aux questions, même complexes, est déjà expérimenté par le grand public à travers ChatGPT. Cependant, il reste encore des limites à surmonter pour exploiter pleinement leur potentiel dans la réponse aux questions complexes (Complex Question Answering - CQA). Dans cet article, nous passons en revue leurs principales limitations et proposons des modèles de conception architecturale hybrides pour relever ces défis. Nous soutenons que l'état actuel des LLM est limité pour traiter seul le CQA en raison de facteurs tels que la nécessité de décomposition pour réduire la complexité, l'acquisition nécessaire de connaissances complémentaires, les capacités de raisonnement spécifiques nécessitant parfois de nombreuses itérations, l'alignement sur les attentes humaines, la protection des données sensibles, la nécessaire capitalisation de l'expérience. Pour remédier à ces limitations, nous proposons la composition de différents modèles architecturaux hybrides LLM augmentant les capacités des LLM sur différentes capacités.

Ainsi, dans cet article, nous débuterons par une présentation des architectures typiques de réponse aux questions, l'arrivée des *transformers*, nous proposerons une définition des questions complexes au sein du contexte des LLM ainsi qu'une liste des limitations principales pour y répondre. Nous avançons ensuite

COnférence en Recherche d'Informations et Applications-CORIA 2024, 19th French Information Retrieval Conference, La Rochelle, France
xavier.daull@naval-group.com (X. Daull); patrice.bellot@univ-amu.fr (P. Bellot); emmanuel.bruno@lis-lab.fr (E. Bruno); vincent.martin@naval-group.com (V. Martin); elisabeth.murisasco@lis-lab.fr (E. Murisasco)


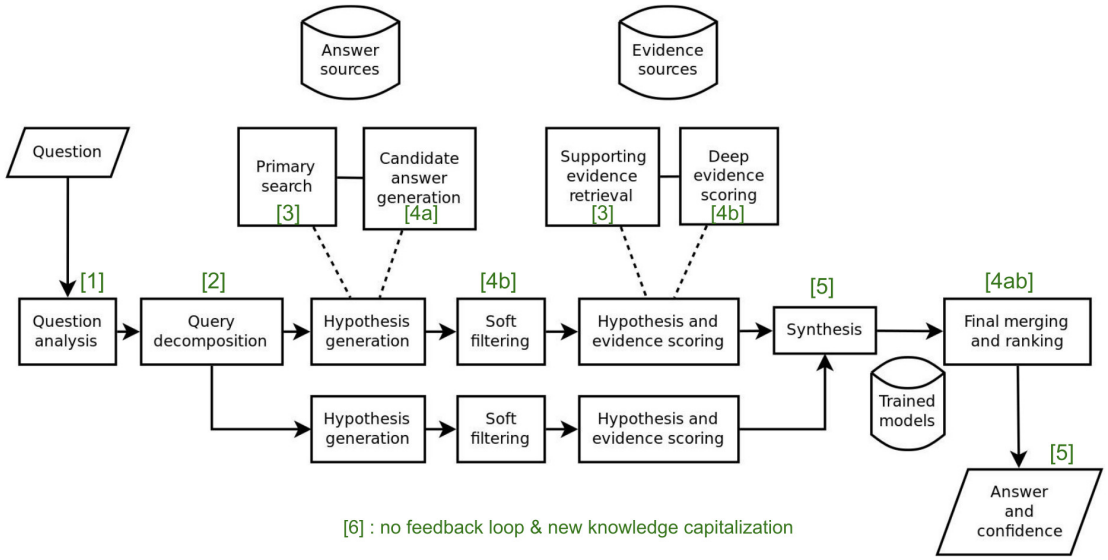

Figure 1: Architecture IBM DeepQA (2010) [2] et étapes du pipeline CQA.

vers la proposition de composition de modèles architecturaux hybrides comme solutions potentielles, et terminons par différentes pistes de recherche.

2. Question answering (QA), nouveau paradigme et limites

Les architectures des systèmes de QA ont évolué récemment avec l'arrivée des architectures *transformers* [1] et la disponibilité de grands modèles de langue LLM pré-entraînés et mis à disposition de la communauté scientifique. Nous allons d'abord rapidement passer en revue les architectures historiques, puis les *transformers* avec les LLM, les nouvelles tendances, et les architectures hybrides. Allons-nous vers des modèles de connaissances gigantesques ou des architectures plus complexes et composées, peut-être en réseau de modèles spécialisés plus petits et d'autres composants ? Nous n'avons pas la réponse mais proposons un aperçu des principales approches pour augmenter les LLM et répondre à des QA complexes.

2.1. Pipeline typique de réponse aux questions

De la capture avec possible affinement de la question, à la génération de la réponse et la capitalisation des connaissances, le pipeline de réponse aux questions (QA) ou de réponse aux questions complexes (CQA) peut suivre un nombre variable d'étapes en fonction de l'architecture et des fonctionnalités. Certaines étapes sont explicites et bien séparées, certaines peuvent être implicites et fusionnées avec d'autres dans la même opération d'un modèle. Cependant, nous pouvons identifier les étapes et options les plus fréquentes. *L'architecture IBM DeepQA* (voir la figure 1 et les références ci-dessous) semble une architecture datée par rapport à certains modèles de langage neuronaux actuels mais elle illustre bien les étapes majeures : **(1) Compréhension et analyse de la question**, cela peut inclure l'affinement de la question et du contexte, l'analyse et la compréhension de la question, du contexte et de l'intention (pour l'identification de la tâche). Cela peut également intégrer une gestion de dialogue pour interagir avec l'utilisateur et comprendre le contexte et l'état de la conversation ; **(2) Construction de la requête** avec une éventuelle **décomposition** des questions complexes et des requêtes en plusieurs étapes ; **(3) Récupération de l'information (RI)** avec une possibiltié d'extension de la base de connaissances du système ; **(4a) Extraction de l'information**, et **(4b) évaluation, notation, classement, filtrage** ; **(5) Génération de réponse**, en langage naturel ou format défini (par exemple, code, tableau, formalisme métier...) ; **(6) Boucle de retour d'information & capitalisation des nouvelles connaissances** : apprentissage et amélioration à partir des retours des

utilisateurs et des modèles, stockage des connaissances liées et générées pour améliorer les compétences de réponse. Ce processus n'est qu'une base car la réponse aux questions complexes peut être un processus dynamique et progressif, et peut également être collaboratif.

2.2. Architectures typiques avant les *transformers*, une approche modulaire

Les architectures typiques des systèmes de QA avant les *transformers* pouvaient être regroupées dans les approches ou méthodes complémentaires suivantes : les **systèmes basés sur des règles** qui utilisent un ensemble de règles prédéfinies pour répondre aux questions ; les **systèmes basés sur la recherche d'information (RI)** qui utilisent un moteur de recherche ou une base de données pour trouver des réponses aux questions ; les **systèmes d'extraction d'informations** qui utilisent le traitement automatique du langage naturel (TALN) pour extraire des informations pertinentes de documents textuels et s'appuient souvent sur des systèmes de recherche d'informations (SRI) ; les **systèmes basés sur des connaissances** qui stockent et récupèrent des informations d'une base de connaissances; les **systèmes de raisonnement à base de cas** qui utilisent une base de données de problèmes précédemment résolus pour trouver des solutions à de nouvelles questions ; les **architectures hybrides** qui assemblent certaines des approches orientées tâches ci-dessus (par exemple, "IBM DeepQA Architecture, 2010" [2]) pour offrir un système de QA plus avancé et pourraient être intégrées avec des modèles de langage naturel pour comprendre par exemple la question initiale.

2.3. La percée des *transformers*

L'émergence de réseaux de neurones avec beaucoup plus de couches, de profondeur, a permis d'apprendre et d'inférer des relations plus riches avec les entrées ; puis l'apparition du mécanisme d'attention [1] a permis à un modèle de se concentrer sélectivement sur certaines parties de l'entrée pour une meilleure compréhension et contextualisation. Cela a conduit à des modèles de langue surpassant les humains dans certaines tâches. Nous pouvons regrouper les modèles de langue basés sur des *transformers* actuels en trois types [3] :

Encodeurs uniquement (BERT [4], RoBERTa [5]) : ils codent une séquence de texte (entrée) en une représentation riche (embedding vectoriel) qui peut être utilisée par un modèle ou une fonction spécifique à une tâche pour la classification, la reconnaissance d'entités nommées (NER), la mesure de similarité sémantique utilisée en RI et QA ou la modélisation thématique (topic modeling). Cela est souvent appelé attention bidirectionnelle, signifiant qu'ils prennent en compte le contexte des mots avant et après le mot cible, ce qui leur permet d'améliorer le succès de certaines tâches. BERT est l'un des modèles de ce type les plus connus, RoBERTa une version optimisée.

Décodeurs uniquement (ex: GPT-3 [6], Mistral 7B [7]) : complètent une séquence d'entrée (principalement une invite, un prompt, du texte) par les mots, ou séquence de mots, suivants les plus probables (génération). Cette génération de texte de gauche à droite peut être très riche, comme écrire une histoire ou répondre à une question (modèle utilisé par ChatGPT). L'invite d'entrée peut être formatée avec des instructions pour réaliser une tâche spécifique (par exemple, classification, résumé, décomposition...) lors de sa génération. Cela est souvent appelé attention causale ou autorégressive.

Encodeurs-Décodeurs (ex: T5 [8], BART [9]) : encodent d'abord le texte d'entrée en un vecteur de taille fixe (couche intermédiaire), puis le décodent vers un texte de sortie sous une forme différente (mapping text-to-text). Ils sont utilisés pour la traduction, le résumé, ou la génération d'une réponse à une question. Ils se composent à la fois d'un modèle encodeur et d'un modèle décodeur, où l'encodeur fournit au décodeur une représentation de taille fixe du texte d'entrée, et le décodeur génère le texte de sortie depuis cette représentation. T5 est connu pour sa capacité multitâches, BART est principalement utilisé pour la génération de texte et le résumé.

3. Répondre aux questions complexes (CQA) dans le contexte des LLM ?

Une question simple comme "Quelle est la capitale de la France ?" nécessite seulement le rappel d'une connaissance factuelle, tandis qu'une question plus complexe comme "Quelles sont les principales causes du changement climatique et leurs solutions potentielles dans une région ?" attend une réponse élaborée et exige ainsi l'intégration de nombreux faits, concepts, procédures et la capacité d'évaluer différentes solutions via un processus potentiellement long, distribué et itératif. Les questions complexes peuvent être définies comme celles qui nécessitent d'adresser simultanément ou séquentiellement plusieurs facteurs élevés de complexité dans les processus cognitifs à impliquer et la variété de connaissances spécifiques au domaine à intégrer pour répondre avec précision. [10] suggère d'utiliser la taxonomie de Bloom pour évaluer les facteurs de complexité des questions, en intégrant les connaissances requises (allant du plus facile au plus difficile : factuelles, conceptuelles, procédurales, métacognitives) et les processus cognitifs (allant du plus facile au plus difficile : se souvenir, comprendre, appliquer, analyser, évaluer). L'acquisition des connaissances requises comme le déroulement du processus cognitif peuvent être longs, répartis entre de nombreux systèmes, et se faire en de nombreuses étapes. Dans le contexte des LLM, nous proposons également d'évaluer la complexité des questions en considérant les principales difficultés et efforts requis pour qu'un LLM puisse les résoudre :

- Les compétences et connaissances requises : une simple recherche d'information dans une mémoire ou engager des raisonnements complexes (résolution de contraintes, déduction, induction, abduction); l'intégration d'un ou plusieurs types de logique; connaissances spécifiques au domaine requises; récupération et traitement d'informations facilement accessibles ou rares; raisonnement sur de nombreuses informations distantes entre elles et à combiner; le format de réponse attendu et l'explicabilité attendue; le gestion de l'ambiguïté et des nuances dans les questions non factuelles; la nécessité d'une décomposition spécifique et d'une résolution en plusieurs étapes.
- Les difficultés à concevoir des métriques ou des ensembles de données appropriés pour développer et mesurer les compétences des LLM dans la résolution des questions cibles.
- Les principales limitations des LLM à surmonter dans ce contexte (voir section 4).
- La quantité d'effort d'entraînement requis pour développer les compétences et les connaissances nécessaires pour résoudre efficacement ces questions.
- Les efforts complémentaires au modèle pour résoudre les questions cibles en contexte après l'entraînement, tels que fournir un contexte et des instructions supplémentaires, des exemples et un renforcement préalable, suivre un processus, et capitaliser progressivement de la connaissance.

4. CQA : besoins et limitations des LLM

L'augmentation de la taille des modèles de langage a montré une amélioration prévisible de la performance [11] dans une large gamme de tâches en aval, bien que des modèles optimisés plus compactes et légèrement moins performants voient le jour [7]. Les études HELM [12] et BIG [13] montrent que l'état de l'art dans la plupart des scénarios est dominé par ces modèles très larges, mais ces modèles présentent encore des lacunes sur différents aspects (par exemple, la robustesse à travers les tâches et les domaines, le raisonnement, l'équité). Les besoins cognitifs et de connaissance de réponse à des questions complexes exigent de repousser ces limites. De nombreux composants supplémentaires sont utilisés ou étudiés pour faire face aux limitations de ces modèles par hybridation avec des LLM. Par exemple, ChatGPT [14] a ajouté l'apprentissage par renforcement avec feedback humain (RLHF) au modèle de langage GPT-3/4 de base pour améliorer considérablement [15] la performance de réponse dans son alignement aux attentes humaines, l'accès à des sources tierces avant génération pour augmenter sa connaissance contextuelle, la capacité de générer et exécuter du code pour réaliser des analyses plus fiables et hors de sa limite de taille d'entrée (contexte). Nous présentons ci-dessous les principaux besoins et limites à adresser des LLM pour

répondre aux questions complexes. Nous les relions également aux solutions de modèles architecturaux hybrides présentées dans la section suivante.

A **Compréhension des questions et du contexte** - amélioration de la compréhension du contexte par des questions de clarification, expansion de la question, définition de sous-objectifs, et dialogue (voir modèles architecturaux : 12, 2).

B **Stratégie de décomposition des questions** - décomposer les questions complexes en sous-questions plus simples, permettant un raisonnement multi-étapes (voir modèles architecturaux : 3, 9).

C **Raisonnement** [16] - intégration de nouveaux raisonnement logiques ou supérieurs, de causalité, et d'apprentissage à partir de code pour améliorer les capacités de résolution de problèmes. Ces capacités de raisonnement pourraient être spécifiques et ajoutées au moment de l'inférence (voir modèles architecturaux: 2, 6, 7, 13, 14).

D **Alignement avec les attentes et valeurs humaines** [14] - garantir que les modèles s'alignent sur les attentes et valeurs humaines tout en gérant les compromis et les différences culturelles (voir modèles architecturaux : 8, 16).

E **Prévention de l'hallucination, véracité, explicabilité et sécurité** [17] - réduire l'hallucination, garantir l'exactitude des réponses, fournir confiance et explications, et maintenir la sécurité dans les domaines critiques (voir modèles architecturaux : 8, 10).

F **Gestion des questions longues** - gérer les entrées de contexte longues, aborder les dépendances à long terme dans le raisonnement, et résumer les documents/sources multiples. La plupart des LLM sont conçus avec une limitation dans la longueur des entrées et des sorties, affectant la taille de la connaissance entrante, les dépendances de longueur de raisonnement, la taille de la réponse (voir modèles architecturaux : 3, 4, 5, 9, 13).

G **Recherche et raisonnement multi-modaux** - de nombreuses connaissances et modèles du monde ne peuvent être capturés uniquement par le texte et peuvent nécessiter l'analyse d'images par exemple pour comprendre et répondre (voir modèles architecturaux : 15).

H **Dimension temporelle** - gérer le raisonnement basé sur le temps, la mise à jour des connaissances, et comprendre les séquences ou processus (voir modèles architecturaux: 8, 18).

I **Protection de la sensibilité des données** - utiliser et protéger les données sensibles, telles que les données privées, la propriété intellectuelle, organisationnelles ou gouvernementales (voir modèles architecturaux : 4, 5).

J **Capitalisation de l'expérience / des connaissances et des compétences** [18] - comme un humain, il devrait être capable de s'améliorer continuellement par l'expérience sur les compétences et les connaissances en utilisant des retours d'information implicites et explicites (voir modèles architecturaux : 2, 8, 11, 12, 17).

K **Adaptation et mise à jour** - adapter les modèles à des domaines et tâches spécifiques, et garantir qu'ils restent à jour avec les nouvelles connaissances (voir modèles architecturaux : 1, 2, 4, 5, 17).

L **Biais** [12] - atténuer les biais, en particulier ceux liés à la race, au genre et à la religion, dans les sorties des modèles (voir modèles architecturaux : 8, 17).

M **Scalabilité** [12] - que ce soit pour l'entraînement ou l'inférence, les ressources de calculs ou mémoire disponibles limitent la capacité maximale d'information pouvant être intégrée pour répondre, comment augmenter cette capacité (voir modèles architecturaux: 4, 11, 13, 16).

D'autres défis existent, comme la résistance aux attaques adverses [19].

5. Modèles architecturaux hybrides augmentant les LLM

Pour répondre aux différentes limites des LLM et aux capacités identifiées pour répondre aux questions complexes, nous avons référencé des composants architecturaux pouvant être ajoutés à un LLM de base, comme un LLM spécialisé, un moteur de recherche, un logiciel, un interpréteur de code... Ceci afin d'aider

à concevoir les différentes manières d'augmenter un LLM à la fois lors de l'inférence ou de l'entraînement, même si certains peuvent se chevaucher. Nous proposons de les classer dans la liste suivante de modèles architecturaux hybrides clés, chacun avec sa description, ses forces et les limites identifiées dans la section précédente que cela peut adresser (**+**), ses faiblesses (**-**), et des illustrations (**e.g.**).

1. LLM + Tête ou Adaptateur: couches ou modules ajoutés à la sortie du modèle pour le spécialiser sur une tâche, ou adaptateur pour ré-entraîner une fraction du modèle dans son comportement principalement..

+: solution pour K (Adaptation et mise à jour) - atteint les performances de référence (SOTA) pour une tâche et/ou un domaine ciblé avec moins de ressources informatiques que le ré-entraînement d'un LLM.
-: nécessite un ensemble de données structuré pour l'entraînement, limité à la tâche spécifique pour laquelle il est conçu pour la spécialisation de tâche, l'ajout de connaissance peut entraîner un effacement d'autres.
e.g. BART avec une tête pour la réponse aux questions [9], les adaptateurs type LoRA [20].

2. LLM + Module d'optimisation des entrées ou sorties: découvrir le meilleur prompt en entrée (contexte ou instructions) pour interroger un LLM ou mieux contrôler la génération en sortie.

+: solution pour A (Compréhension des questions et du contexte), C (Raisonnement), J (Capitalisation de l'expérience / des connaissances et des compétences), K (Adaptation et mise à jour) - améliore la performance des LLM sur une seule ou plusieurs tâches sans ré-entraînement, et peut s'adapter dynamiquement à la tâche.
-: très sensible à de légères variations de prompt, peut nécessiter un contexte substantiel, trouver un prompt robuste peut être complexe, le contrôle de la sortie peut générer de fortes incohérences si mal aligné avec le prompt.
e.g. optimisation du prompt en entrée [21], contrôle programmatique de la génération en sortie [22], génération d'instructions [23].

3. LLM + décomposeur de question/tâche, plan, action: décompose efficacement les tâches complexes en sous-tâches adressables suivant un plan de résolution.

+: solution pour B (Stratégie de décomposition des questions), F (Gestion des questions longues)
- efficace pour résoudre des tâches plus complexes nécessitant plusieurs étapes ou sources en les convertissant en plusieurs sous-tâches gérables et un plan de résolution efficace (a priori, itératif ou récursif). Il peut bénéficier de l'incorporation de sources de connaissances externes ou de capacités de raisonnement.
-: plus de temps et de ressources à mettre en œuvre, difficultés avec contexte long.
e.g. décomposition itérative avec supervision du processus de raisonnement [24], lie raisonnement et décomposition d'actions [25], décomposition de QA non supervisée [26], Talk2Data pour la décomposition de QA de haut niveau [27], DeepQA décompose des QA basés sur des faits [28], apprend à décomposer des QA composées par renforcement [29], décomposition d'invites successives pour CQA [30].

4. LLM + Recherche sémantique d'informations: intègre des sources externes plutôt que de les stocker dans le LLM. Peut être amélioré avec un mécanisme d'apprentissage par renforcement (RL: reinforcement learning).

+: solution pour F (Gestion des questions longues), I (Protection de la sensibilité des données), K (Adaptation et mise à jour), M (Scalabilité) - intègre toute source externe à jour sans augmenter la taille du LLM, permet des modèles beaucoup plus petits (RETRO est 1/25 de la taille de GPT-3 pour une performance équivalente), contrôle sur les sources (sensibilité, explicabilité, mise à jour des connaissances).
-: moins performant avec des tâches nécessitant un raisonnement abstrait ou créatif qu'un LLM ayant appris sur ces données, limité par la qualité et la couverture des sources externes.
e.g. RETRO de Deepmind [31], DrQA de Facebook [32], FiDO [33], Atlas [34], formation d'agents RL pour interroger des connaissances externes [35], Toolformer [36], minimisation de l'effort de recherche [37].

---

5. LLM + Récupération d'informations symboliques/structurées: exploite les informations symboliques et structurées (ex. graphe de connaissance, ontologies, SQL).

+: solution pour F (Gestion des questions longues), I (Protection de la sensibilité des données) - permet une adaptation rapide à une tâche, un domaine, avec peu de données, de suivre plus efficacement les règles et les concepts structurés (ontologies, SGBDR, graphe, taxonomie, métadonnées).
-: l'intégration neuro-symbolique est complexe, la création de données symboliques requiert du un effort/temps important.
e.g. modèle UNIQORN [38], Heterformer [39], UnifiedSKG [40].

---

6. LLM + Programme (logiciel ou service via API): exploite des capacités logicielles externes spécialisées (ex. solveur mathématique, simulation, ...) pour effectuer des tâches difficiles ou impossibles à réaliser par un LLM.

+ : solution pour C (Raisonnement) - tire parti des performances et de la robustesse éprouvées des logiciels/services externes en matière de capture ou traitement d'informations, de modélisation du monde.
- : apprentissage de bout en bout difficile et intégration potentiellement complexe (ex. étapes supplémentaires de prétraitement ou de post-traitement)
ex. utilisation d'un modèle physique [41], WebGPT [42], SeeKeR [43], Toolformer [36].

---

7. LLM + Interpréteur de code : génère du code pour déléguer des tâches difficiles ou non réalisables par un LLM, permet des traitements neuro-symboliques.

+ : solution pour C (Raisonnement) - exploite des capacités robustes de raisonnement et algorithmiques, ainsi que l'écosystème du langage.
- : difficile avec des tâches nécessitant une compréhension plus profonde du contexte ou des concepts, dépendance envers des interpréteurs de code externes, risque cyber plus élevé.
ex. PAL [44], résolution de problèmes mathématiques par raisonnement coopératif [45] ou une synthèse de programme [46], auto-amélioration de ses capacités de programmation [47], Codex [48], AlphaCode [49].

---

8. LLM + Boucle de feedback Homme/AI : apprendre la politique optimale pour des objectifs (qualité de réponse, sécurité, sources de données...)

+ : solution pour D (Alignement avec les attentes et valeurs humaines), E (Prévention de l'hallucination, véracité, explicabilité et sécurité), H (Dimension temporelle), J (Capitalisation de l'expérience, des connaissances, des compétences), L (Biais) - brique majeure pour aligner les LLM aux attentes humaines, la personnalisation, le contrôle de la sécurité et de la qualité des réponses.
- : le feedback humain est extrêmement chronophage. Cela peut être réduit en incorporant l'apprentissage actif et l'apprentissage de feedback AI, mais la conception est plus complexe.
ex. apprentissage par renforcement avec feedback humain [14, 50, 51], avec feedback par IA [52], algorithmes de recherche Monte Carlo [53, 54] ou DiL-piKL [55] ou préférence PPO dans Diplomacy [55], imitation d'experts [56], recherche Web RL dans WebGpt [42], citation RL dans GopherCite [57].

---

9. LLM en cascade/chaînés : résoudre des problèmes complexes en les résolvant par étapes comme une chaîne avec plusieurs requêtes séquentielles au LLM, pouvant être itérative ou récursive..

+ : solution pour B (Stratégie de décomposition des questions), F (Gestion des questions longues) - facilite le contrôle humain sur le processus de conception et d'exécution; peut résoudre des problèmes de complexité supérieure aux capacités des mêmes LLM en décomposant les tâches et la conception ; chaîne causale utile pour l'explicabilité ; permet l'optimisation de la chaîne et tire parti de la spécialisation pour palier certaines limitations des LLM.
- : moins efficace pour les tâches nécessitant un contexte et des dépendances de raisonnement étendus.
ex. résolution en cascade de LLM [58], chaînes AI [59], chaîne visuelle collaborative de prompts [60], raisonnement robuste par sélection-inférence [61], déduction logique multi-étapes lisible par l'homme en QA scientifique améliorant exactitude et fidélité [62], prompter itérativement un LLM [63, 64, 30].

---

10. LLM + Vérificateur : fournit une évaluation de la véracité et des sources.

+ : solution pour E (Prévention de l'hallucination, véracité, explicabilité et sécurité) - garantit la crédibilité et la fiabilité de l'information tout en atténuant les hallucinations en fournissant des sources vérifiables et une évaluation des preuves.
- : limité par la qualité et la couverture des sources externes, n'élimine pas le risque de désinformation.
ex. GopherCite fournit des citations vérifiées [57], raisonnement logique avec vérification factuelle interprétable [65], survey sur la vérification automatique des faits [66], détection de contenu halluciné [67], approche RL pour l'explicabilité [68].

---

11. LLM + Routeur ou Discriminateur : orienter une tâche/un domaine vers le modèle le plus approprié avec instructions et contexte.

+ : solution pour J (Capitalisation de l'expérience / des connaissances et des compétences), M (Scalabilité) - accélére l'entraînement du LLM et surtout sa vitesse d'inférence en orientant vers le modèle le plus adapté (performance vs ressource) avec les instructions les plus appropriées.
- : complexe à mettre en œuvre et maintenir ; risque sur raisonnement partagé et dépendances sémantiques distantes.
ex. tâches zéro-shot sur LLM par discriminateur [69], Mixtral améliore considérablement un LLM par l'évolution vers une architecture mixture of experts MoE [70], Branch-Train-Merge un entraînement rapide de LM experts [71], couches DEMIX [72].

---

12. LLM + Module de Dialogue : dialogue pour affiner une question ou mieux capturer un contexte long, peut inclure une ontologie pour aider à structurer la définition des tâches.

+ : solution pour A (Compréhension des questions et du contexte), J (Capitalisation de l'expérience / des connaissances et des compétences) - améliore la compréhension des contextes, des problèmes et des concepts complexes grâce à l'interaction humaine, à l'orientation, à l'affinement progressif et à la résolution de problèmes.
- : implémentation plus longue et plus coûteuse, doit correspondre au cadre d'utilisation ciblé.
ex. GODEL [73], OPAL [74], CommaQA [75], ChatGPT

---

13. LM + Mémoire en Lecture-Écriture : ajouter une mémoire externe au LLM permettant de stocker et de transmettre des informations entre plusieurs inférences ou tierces.

+ : solution pour C (Raisonnement), F (Gestion des questions longues), M (Scalabilité) - permet s'affranchir de la limite de contexte (taille de l'entrée), renforcer la contrôlabilité et la robustesse, peut simuler tout processus et les dépendances à long terme (pour le raisonnement, les dialogues, la synthèse, la récupération, l'algorithmique...).
- : problèmes de scalabilité avec l'augmentation de la taille du modèle.
ex. mémoire universelle pour LLM [76], ajout d'une mémoire récurrente [77], conversations à long terme [78], mémoire de travail pour le raisonnement scientifique [79].

---

14. LLM + Générateur/Vérificateur : résolution par la génération de multiples solutions potentielles, vérifie la cohérence, regroupe/classe pour identifier les meilleures réponses.

+ : solution pour C (Raisonnement) - peut résoudre des questions/tâches complexes nouvelles par génération et hybridation.
- : intensif en ressources et coûteux ; peut ne pas toujours conduire à des améliorations de performance proportionnelles à l'effort.
ex. modèle AlphaCode [49], raisonnement coopératif [45], auto-cohérence de la chaîne de pensée[80], vérificateurs de résolution mathématique [81].

---

15. LLM + Multimodalité : rechercher et raisonner sur des sources complémentaires non textuelles (image, audio, vidéo, capteurs...)..

+ : solution pour G (Recherche et raisonnement multi-modaux) - tire parti des connaissances non disponibles ou moins performantes sous format texte, et combine chaque modalité dans le raisonnement.
- : complexité d'intégration (représentation, alignement, raisonnement, génération...) et coût, difficulté accrue pour aborder les problèmes d'explicabilité et d'hallucination.
ex. fondations et tendances récentes en apprentissage multimodal [82, 83].

---

16. Composition de LLM ou systèmes Multi-Agents : combinaison de LLM en collaboration ou compétition pour répondre..

+ : solution pour D (Alignement avec les attentes et valeurs humaines), M (Scalabilité) - améliore la précision de l'inférence, la généralisation et la stabilité en combinant la diversité, le multi-agent peut permettre une distribution dynamique de rôles spécialisés adapté au problème.
- : coût et complexité, la coordination du multi-agent peut s'avérer complexe et la convergence plus lente.
ex. collaboration d'agents spécialisés pour répondre [84, 85], apprentissage en ensemble pour validation et explication [86], formation parallèle d'experts LLM [71], ensembles de LLM via un consensus itératif [87], composition automatique de modules neuronaux [88].

---

17. LLM + (multi) Teacher : améliore efficacement les connaissances/compétences des LLM grâce à 1 ou plusieurs enseignants experts dans des domaines/tâches donnés..

+ : solution pour J (Capitalisation de l'expérience / des connaissances et des compétences), K (Adaptation et mise à jour), L (Biais) - accélère et améliore l'apprentissage des connaissances, l'adaptation, le multitâche, renforce les capacités de raisonnement (ex. temporel).
- : coût, complexité & complétude du transfert de connaissances (ex. portée du domaine, biais).
ex. Architecture teacher-student pour l'apprentissage de connaissances [89], mieux apprendre de plusieurs enseignants [90].

18. LLM + Raisonnement temporel (peut être étendu à la dimension spatiale) : améliore les performances sur les tâches liées au temps (compréhension, récupération, raisonnement temporel).

+ : solution pour H (Dimension temporelle) - permet l'estimation temporelle, le classement & le regroupement, le raisonnement, la détection d'incohérence, la gestion de l'oubli et la mise à jour des connaissances.
- : l'intégration efficace peut être un défi, encore peu de recherche.
ex. TimeBERT : Extension des représentations avec des informations temporelles [91], améliore le raisonnement temporel grâce à une modalité audio ajoutée [92].

6. Conclusion et travaux futurs

Dans cet article, nous avons examiné les besoins et limites actuelles des LLM pour répondre à des questions complexes et proposé diverses stratégies de modèles d'hybridation pour augmenter leurs capacités. Nous avons identifié plusieurs domaines clés dans lesquels les LLM rencontrent des défis, tels que l'adaptabilité, l'atténuation des biais, la scalabilité, l'amélioration du contexte, la stratégie de décomposition, le raisonnement, l'alignement avec les valeurs humaines, la prévention des hallucinations et le raisonnement multimodal, entre autres. Pour chacun, nous avons présenté et lié différentes solutions architecturales hybrides pour augmenter les LLM et pallier ses limites. Cela offre des solutions prometteuses pour surmonter les limites des LLM dans des scénarii de réponse à des questions complexes. Leur exploration continue ouvrira la voie à des systèmes d'IA plus puissants et adaptables, capables de résoudre des problèmes réels complexes en collaboration avec des experts humains.

Les travaux futurs pourraient affiner automatiquement ou avec une boucle humaine/LLM (human-in-the-loop) les rôles, les processus de résolution, les performances et la sélection des patterns d'architecture, en mettant l'accent sur leur conception et leur intégration pour permettre de répondre à des questions toujours plus complexes par capitalisation automatique tout en réduisant le coût de réponse et en élargissant son adaptation à de multiples domaines. Ces améliorations pourraient être inspirées des nombreuses approches par composition de LLM comme présenté dans cet article, mais aussi des architectures informatiques existantes, l'intégration fonctionnelle et l'orchestration cérébrale des fonctions cognitives, des écosystèmes biologiques, des systèmes sociaux. Un autre domaine de recherche important pourrait être l'amélioration continue des capacités de résolution de problèmes des LLM grâce à un apprentissage renforcé polyvalent par les humains et des systèmes tiers, des stratégies évolutives, une gestion plus large de l'expérience. Nous devrions également explorer des moyens d'orchestrer et d'optimiser la boucle de collaboration humain-IA, le transfert de connaissances croisées humain-IA, permettant aux LLM et aux experts humains de résoudre ensemble des problèmes de plus en plus complexes tout en maintenant des niveaux élevés d'acceptabilité et de confiance.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention Is All You Need, 2017. arXiv:1706.03762.

[2] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, C. Welty, Building Watson: An Overview of the DeepQA Project, AI Magazine 31 (2010) 59–79. doi:10.1609/aimag.v31i3.2303.

[3] T. Lewis, v. W. Leandro, W. Thomas, Natural Language Processing with Transformers_ Building Language Applications with Hugging Face, 2022.
[4] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019. arXiv:1810.04805.
[5] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa: A Robustly Optimized BERT Pretraining Approach, 2019. doi:10.48550/arXiv.1907.11692.
[6] M. Zong, B. Krishnamachari, A survey on GPT-3, 2022. arXiv:2212.00857.
[7] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, W. E. Sayed, Mistral 7B, 2023. doi:10.48550/arXiv.2310.06825.
[8] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, 2020. doi:10.48550/arXiv.1910.10683. arXiv:1910.10683.
[9] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, 2019. doi:10.48550/arXiv.1910.13461. arXiv:1910.13461.
[10] S. Ullrich, M. Geierhos, Using Bloom's Taxonomy to Classify Question Complexity, in: Proceedings of the Fourth International Conference on Natural Language and Speech Processing (ICNLSP 2021), Association for Computational Linguistics, Trento, Italy, 2021, pp. 285–289.
[11] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, W. Fedus, Emergent Abilities of Large Language Models, 2022. doi:10.48550/arXiv.2206.07682. arXiv:2206.07682.
[12] Liang et al., Holistic Evaluation of Language Models, 2022. doi:10.48550/arXiv.2211.09110. arXiv:2211.09110.
[13] BIG et al., Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models, 2022. doi:10.48550/arXiv.2206.04615. arXiv:2206.04615.
[14] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, N. Joseph, S. Kadavath, J. Kernion, T. Conerly, S. El-Showk, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, T. Hume, S. Johnston, S. Kravec, L. Lovitt, N. Nanda, C. Olsson, D. Amodei, T. Brown, J. Clark, S. McCandlish, C. Olah, B. Mann, J. Kaplan, Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback, 2022. doi:10.48550/arXiv.2204.05862. arXiv:2204.05862.
[15] K. Mahowald, A. A. Ivanova, I. A. Blank, N. Kanwisher, J. B. Tenenbaum, E. Fedorenko, Dissociating language and thought in large language models: A cognitive perspective, 2023. arXiv:2301.06627.
[16] A. Rogers, M. Gardner, I. Augenstein, QA Dataset Explosion: A Taxonomy of NLP Resources for Question Answering and Reading Comprehension, ACM Computing Surveys (2022) 3560260. arXiv:2107.12708.
[17] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, P. Fung, Survey of hallucination in NL Generation, ACM Computing Surveys 55 (2023) 248:1–248:38. doi:10.1145/3571730.
[18] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, A. Anandkumar, Voyager: An Open-Ended Embodied Agent with Large Language Models, 2023. arXiv:2305.16291.
[19] E. Shayegani, M. A. A. Mamun, Y. Fu, P. Zaree, Y. Dong, N. Abu-Ghazaleh, Survey of Vulnerabilities in Large Language Models Revealed by Adversarial Attacks, 2023. doi:10.48550/arXiv.2310.10844. arXiv:2310.10844.
[20] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-Rank Adaptation of Large Language Models, 2021. doi:10.48550/arXiv.2106.09685.

arXiv:2106.09685.
[21] B. Lester, R. Al-Rfou, N. Constant, The Power of Scale for Parameter-Efficient Prompt Tuning, 2021. doi:10.48550/arXiv.2104.08691. arXiv:2104.08691.
[22] L. Beurer-Kellner, M. Fischer, M. Vechev, Prompting Is programming: Query language for large language models, 2022. doi:10.48550/arXiv.2212.06094. arXiv:2212.06094.
[23] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, H. Hajishirzi, Self-Instruct: Aligning Language Model with Self Generated Instructions, 2022. doi:10.48550/arXiv.2212.10560. arXiv:2212.10560.
[24] J. Reppert, B. Rachbach, C. George, L. Stebbing, J. Byun, M. Appleton, A. Stuhlmüller, Iterated Decomposition: Improving Science Q&A by Supervising Reasoning Processes, 2023. arXiv:2301.01751.
[25] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, ReAct: Synergizing Reasoning and Acting in Language Models, 2022. doi:10.48550/arXiv.2210.03629. arXiv:2210.03629.
[26] E. Perez, P. Lewis, W.-t. Yih, K. Cho, D. Kiela, Unsupervised Question Decomposition for Question Answering (EMNLP 2020), 2020. doi:10.48550/arXiv.2002.09758. arXiv:2002.09758.
[27] D. Shi, Y. Guo, M. Guo, Y. Wu, Q. Chen, N. Cao, Talk2Data: High-Level Question Decomposition for Data-Oriented Question and Answering, arXiv:2107.14420 [cs] (2021). arXiv:2107.14420.
[28] A. Kalyanpur, S. Patwardhan, B. K. Boguraev, A. Lally, J. Chu-Carroll, Fact-based question decomposition in DeepQA, IBM Journal of Research and Development 56 (2012) 13:1–13:11. doi:10.1147/JRD.2012.2188934.
[29] H. Yang, H. Wang, S. Guo, W. Zhang, H. Chen, Learning to Decompose Compound Questions with Reinforcement Learning (2022).
[30] D. Dua, S. Gupta, S. Singh, M. Gardner, Successive Prompting for Decomposing Complex Questions, 2022. doi:10.48550/arXiv.2212.04092. arXiv:2212.04092.
[31] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. V. D. Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. D. L. Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. Rae, E. Elsen, L. Sifre, Improving Language Models by Retrieving from Trillions of Tokens, in: Proceedings of the 39th International Conference on Machine Learning, PMLR, 2022, pp. 2206–2240.
[32] D. Chen, A. Fisch, J. Weston, A. Bordes, Reading Wikipedia to Answer Open-Domain Questions, 2017. doi:10.48550/arXiv.1704.00051. arXiv:1704.00051.
[33] M. de Jong, Y. Zemlyanskiy, J. Ainslie, N. FitzGerald, S. Sanghai, F. Sha, W. Cohen, FiDO: Fusion-in-Decoder optimized for stronger performance and faster inference, 2022. arXiv:2212.08153.
[34] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, E. Grave, Atlas: Few-shot Learning with Retrieval Augmented Language Models, 2022. arXiv:2208.03299.
[35] I.-J. Liu, X. Yuan, M.-A. Côté, P.-Y. Oudeyer, A. G. Schwing, Asking for Knowledge: Training RL Agents to Query External Knowledge Using Language, 2022. doi:10.48550/arXiv.2205.06111. arXiv:2205.06111.
[36] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, T. Scialom, Toolformer: Language Models Can Teach Themselves to Use Tools, 2023. doi:10.48550/arXiv.2302.04761. arXiv:2302.04761.
[37] N. Varshney, M. Luo, C. Baral, Can Open-Domain QA Reader Utilize External Knowledge Efficiently like Humans?, 2022. doi:10.48550/arXiv.2211.12707. arXiv:2211.12707.
[38] S. Pramanik, J. Alabi, R. S. Roy, G. Weikum, UNIQORN: Unified Question Answering over RDF Knowledge Graphs and Natural Language Text, arXiv:2108.08614 [cs] (2022).

arXiv:2108.08614.

[39] B. Jin, Y. Zhang, Q. Zhu, J. Han, Heterformer: A Transformer Architecture for Node Representation Learning on Heterogeneous Text-Rich Networks, 2022. doi:10.48550/arXiv.2205.10282. arXiv:2205.10282.

[40] T. Xie, C. H. Wu, P. Shi, R. Zhong, T. Scholak, M. Yasunaga, C.-S. Wu, M. Zhong, P. Yin, S. I. Wang, V. Zhong, B. Wang, C. Li, C. Boyle, A. Ni, Z. Yao, D. Radev, C. Xiong, L. Kong, R. Zhang, N. A. Smith, L. Zettlemoyer, T. Yu, UnifiedSKG: Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models, 2022. doi:10.48550/arXiv.2201.05966. arXiv:2201.05966.

[41] R. Liu, J. Wei, S. S. Gu, T.-Y. Wu, S. Vosoughi, C. Cui, D. Zhou, A. M. Dai, Mind's Eye: Grounded Language Model Reasoning through Simulation, 2022. doi:10.48550/arXiv.2210.05359. arXiv:2210.05359.

[42] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, X. Jiang, K. Cobbe, T. Eloundou, G. Krueger, K. Button, M. Knight, B. Chess, J. Schulman, WebGPT: Browser-assisted question-answering with human feedback, 2022. doi:10.48550/arXiv.2112.09332. arXiv:2112.09332.

[43] K. Shuster, M. Komeili, L. Adolphs, S. Roller, A. Szlam, J. Weston, Language Models that Seek for Knowledge: Modular Search & Generation for Dialogue and Prompt Completion, arXiv:2203.13224 [cs] (2022). arXiv:2203.13224.

[44] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, G. Neubig, PAL: Program-aided Language Models, 2023. doi:10.48550/arXiv.2211.10435. arXiv:2211.10435.

[45] X. Zhu, J. Wang, L. Zhang, Y. Zhang, R. Gan, J. Zhang, Y. Yang, Solving Math Word Problem via Cooperative Reasoning induced Language Models, 2022. doi:10.48550/arXiv.2210.16257. arXiv:2210.16257.

[46] I. Drori, S. Zhang, R. Shuttleworth, L. Tang, A. Lu, E. Ke, K. Liu, L. Chen, S. Tran, N. Cheng, R. Wang, N. Singh, T. L. Patti, J. Lynch, A. Shporer, N. Verma, E. Wu, G. Strang, A Neural Network Solves, Explains, and Generates University Math Problems by Program Synthesis and Few-Shot Learning at Human Level, 2022. arXiv:2112.15594.

[47] P. Haluptzok, M. Bowers, A. T. Kalai, Language Models Can Teach Themselves to Program Better, 2022. arXiv:2207.14502.

[48] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, W. Zaremba, Evaluating Large Language Models Trained on Code, 2021. doi:10.48550/arXiv.2107.03374. arXiv:2107.03374.

[49] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. D. Lago, T. Hubert, P. Choy, C. d. M. d'Autume, I. Babuschkin, X. Chen, P.-S. Huang, J. Welbl, S. Gowal, A. Cherepanov, J. Molloy, D. J. Mankowitz, E. S. Robson, P. Kohli, N. de Freitas, K. Kavukcuoglu, O. Vinyals, Competition-Level Code Generation with AlphaCode, 2022. doi:10.48550/arXiv.2203.07814. arXiv:2203.07814.

[50] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe, Training language models to follow instructions with human feedback, 2022. arXiv:2203.02155.

[51] O. Daniels-Koch, R. Freedman, The Expertise Problem: Learning from Specialized Feedback, 2022.

doi:10.48550/arXiv.2211.06519. arXiv:2211.06519.

[52] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, C. Chen, C. Olsson, C. Olah, D. Hernandez, D. Drain, D. Ganguli, D. Li, E. Tran-Johnson, E. Perez, J. Kerr, J. Mueller, J. Ladish, J. Landau, K. Ndousse, K. Lukosuite, L. Lovitt, M. Sellitto, N. Elhage, N. Schiefer, N. Mercado, N. DasSarma, R. Lasenby, R. Larson, S. Ringer, S. Johnston, S. Kravec, S. E. Showk, S. Fort, T. Lanham, T. Telleen-Lawton, T. Conerly, T. Henighan, T. Hume, S. R. Bowman, Z. Hatfield-Dodds, B. Mann, D. Amodei, N. Joseph, S. McCandlish, T. Brown, J. Kaplan, Constitutional AI: Harmlessness from AI Feedback, 2022. arXiv:2212.08073.

[53] W. Ye, P. Abbeel, Y. Gao, Spending Thinking Time Wisely: Accelerating MCTS with Virtual Expansions (2021).

[54] J. Laurent, A. Platzer, Learning to Find Proofs and Theorems by Learning to Refine Search Strategies, undefined (2022).

[55] A. Bakhtin, D. J. Wu, A. Lerer, J. Gray, A. P. Jacob, G. Farina, A. H. Miller, N. Brown, Mastering the Game of No-Press Diplomacy via Human-Regularized Reinforcement Learning and Planning, 2022. arXiv:2210.05492.

[56] M. Yang, D. Schuurmans, P. Abbeel, O. Nachum, Chain of Thought Imitation with Procedure Cloning, 2022. doi:10.48550/arXiv.2205.10816. arXiv:2205.10816.

[57] J. Menick, M. Trebacz, V. Mikulik, J. Aslanides, F. Song, M. Chadwick, M. Glaese, S. Young, L. Campbell-Gillingham, G. Irving, N. McAleese, Teaching language models to support answers with verified quotes, 2022. doi:10.48550/arXiv.2203.11147. arXiv:2203.11147.

[58] D. Dohan, W. Xu, A. Lewkowycz, J. Austin, D. Bieber, R. G. Lopes, Y. Wu, H. Michalewski, R. A. Saurous, J. Sohl-dickstein, K. Murphy, C. Sutton, Language Model Cascades, 2022. doi:10.48550/arXiv.2207.10342. arXiv:2207.10342.

[59] T. Wu, M. Terry, C. J. Cai, AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts, in: CHI Conference on Human Factors in Computing Systems, CHI '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 1–22. doi:10.1145/3491102.3517582.

[60] T. Wu, E. Jiang, A. Donsbach, J. Gray, A. Molina, M. Terry, C. Cai, PromptChainer: Chaining Large Language Model Prompts through Visual Programming, 2022.

[61] A. Creswell, M. Shanahan, I. Higgins, Selection-Inference: Exploiting Large Language Models for Interpretable Logical Reasoning, 2022. arXiv:2205.09712.

[62] A. Creswell, M. Shanahan, Faithful Reasoning Using Large Language Models, 2022. doi:10.48550/arXiv.2208.14271. arXiv:2208.14271.

[63] B. Wang, X. Deng, H. Sun, Iteratively Prompt Pre-trained Language Models for Chain of Thought, 2022. doi:10.48550/arXiv.2203.08383. arXiv:2203.08383.

[64] K. Yang, Y. Tian, N. Peng, D. Klein, Re3: Generating Longer Stories With Recursive Reprompting and Revision, 2022. doi:10.48550/arXiv.2210.06774. arXiv:2210.06774.

[65] J. Chen, Q. Bao, C. Sun, X. Zhang, J. Chen, H. Zhou, Y. Xiao, L. Li, LOREN: Logic-Regularized Reasoning for Interpretable Fact Verification, Proceedings of the AAAI Conference on Artificial Intelligence 36 (2022) 10482–10491. arXiv:2012.13577.

[66] Z. Guo, M. Schlichtkrull, A. Vlachos, A Survey on Automated Fact-Checking, 2022. doi:10.48550/arXiv.2108.11896. arXiv:2108.11896.

[67] C. Zhou, G. Neubig, J. Gu, M. Diab, P. Guzman, L. Zettlemoyer, M. Ghazvininejad, Detecting Hallucinated Content in Conditional Neural Sequence Generation, 2021. doi:10.48550/arXiv.2011.02593. arXiv:2011.02593.

[68] T. Liu, Q. Guo, X. Hu, Y. Zhang, X. Qiu, Z. Zhang, RLET: A Reinforcement Learning Based Approach for Explainable QA w/ Entailment Trees, 2022. doi:10.48550/arXiv.2210.17095. arXiv:2210.17095.

[69] H. Xu, Z. Lin, J. Zhou, Y. Zheng, Z. Yang, A Universal Discriminator for Zero-Shot Generalization, 2022. doi:10.48550/arXiv.2211.08099. arXiv:2211.08099.
[70] Mistral.ai Team et al, Mixtral of Experts, 2024. doi:10.48550/arXiv.2401.04088. arXiv:2401.04088.
[71] M. Li, S. Gururangan, T. Dettmers, M. Lewis, T. Althoff, N. A. Smith, L. Zettlemoyer, Branch-Train-Merge: Embarrassingly parallel training of expert language models, 2022. doi:10.48550/arXiv.2208.03306. arXiv:2208.03306.
[72] S. Gururangan, M. Lewis, A. Holtzman, N. A. Smith, L. Zettlemoyer, DEMix Layers: Disentangling Domains for Modular Language Modeling, 2021. doi:10.48550/arXiv.2108.05036. arXiv:2108.05036.
[73] B. Peng, M. Galley, P. He, C. Brockett, L. Liden, E. Nouri, Z. Yu, B. Dolan, J. Gao, GODEL: Large-Scale Pre-Training for Goal-Directed Dialog, 2022. arXiv:2206.11309.
[74] Z. Chen, Y. Liu, L. Chen, S. Zhu, M. Wu, K. Yu, OPAL: Ontology-Aware Pretrained Language Model for End-to-End Task-Oriented Dialogue, 2022. doi:10.48550/arXiv.2209.04595. arXiv:2209.04595.
[75] T. Khot, K. Richardson, D. Khashabi, A. Sabharwal, Learning to Solve Complex Tasks by Talking to Agents, arXiv:2110.08542 [cs] (2021). arXiv:2110.08542.
[76] D. Schuurmans, Memory Augmented Large Language Models are Computationally Universal, 2023. arXiv:2301.04589.
[77] A. Bulatov, Y. Kuratov, M. Burtsev, Recurrent Memory Transformer, Advances in Neural Information Processing Systems 35 (2022) 11079–11091.
[78] J. Xu, A. Szlam, J. Weston, Beyond Goldfish Memory: Long-Term Open-Domain Conversation, 2021. doi:10.48550/arXiv.2107.07567. arXiv:2107.07567.
[79] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, R. Stojnic, Galactica: A Large Language Model for Science, 2022. doi:10.48550/arXiv.2211.09085. arXiv:2211.09085.
[80] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, D. Zhou, Self-Consistency Improves Chain of Thought Reasoning in Language Models, 2022. arXiv:2203.11171.
[81] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, J. Schulman, Training Verifiers to Solve Math Word Problems, 2021. doi:10.48550/arXiv.2110.14168. arXiv:2110.14168.
[82] P. P. Liang, A. Zadeh, L.-P. Morency, Foundations and Recent Trends in Multimodal Machine Learning: Principles, Challenges, and Open Questions, 2022. doi:10.48550/arXiv.2209.03430. arXiv:2209.03430.
[83] OpenAI, GPT-4 Technical Report, 2023. doi:10.48550/arXiv.2303.08774. arXiv:2303.08774.
[84] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, C. Wang, AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation, 2023. doi:10.48550/arXiv.2308.08155. arXiv:2308.08155.
[85] M. Zhuge, H. Liu, F. Faccio, D. R. Ashley, R. Csordás, A. Gopalakrishnan, A. Hamdi, H. A. A. K. Hammoud, V. Herrmann, K. Irie, L. Kirsch, B. Li, G. Li, S. Liu, J. Mai, P. Piękos, A. Ramesh, I. Schlag, W. Shi, A. Stanić, W. Wang, Y. Wang, M. Xu, D.-P. Fan, B. Ghanem, J. Schmidhuber, Mindstorms in Natural Language-Based Societies of Mind, 2023. arXiv:2305.17066.
[86] N. Q. Huy, T. M. Phuong, N. X. Bach, Autoencoding Language Model Based Ensemble Learning for Commonsense Validation and Explanation, 2022. doi:10.48550/arXiv.2204.03324. arXiv:2204.03324.
[87] S. Li, Y. Du, J. B. Tenenbaum, A. Torralba, I. Mordatch, Composing Ensembles of Pre-trained Models via Iterative Consensus, 2022. arXiv:2210.11522.
[88] J. Andreas, M. Rohrbach, T. Darrell, D. Klein, Neural Module Networks, in: Proceedings of the

IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 39–48.
[89] C. Hu, X. Li, D. Liu, X. Chen, J. Wang, X. Liu, Teacher-Student Architecture for Knowledge Learning: A Survey, 2022. doi:10.48550/arXiv.2210.17332. arXiv:2210.17332.
[90] C. Wu, F. Wu, Y. Huang, One Teacher is Enough? Pre-trained Language Model Distillation from Multiple Teachers, 2021. doi:10.48550/arXiv.2106.01023. arXiv:2106.01023.
[91] J. Wang, A. Jatowt, M. Yoshikawa, TimeBERT: Extending Pre-Trained Language Representations with Temporal Information, 2022. arXiv:2204.13032.
[92] H. M. Fayek, J. Johnson, Temporal Reasoning via Audio Question Answering, IEEE/ACM Transactions on Audio, Speech, and Language Processing 28 (2020) 2283–2294. doi:10.1109/TASLP.2020.3010650.