Intégration du raisonnement numérique dans les modèles de langue : État de l'art et direction de recherche

Sarah Abchiche[1, 2] Lynda Said Lhadj[1] Laure Soulier[2, 3] Vincent Guigue[2, 4]
(1) Ecola Nationale Supérieure d'Informatique (ESI), LCSI, Alger, Algérie
(2) Sorbonne Université, CNRS, ISIR, F-75005 Paris, France.
(3) Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique, 91400, Orsay, France.
(4) AgroParisTech, UMR MIA-PS, France.
is_abchiche@esi.dz, l_said_lhadj@esi.dz, laure.soulier@isir.upmc.fr, vincent.guigue@isir.upmc.fr

RÉSUMÉ

Ces dernières années, les modèles de langue ont connu une évolution galopante grâce à l'augmentation de la puissance de calcul qui a rendu possible l'utilisation des réseaux de neurones. Parallèlement, l'intégration du raisonnement numérique dans les modèles de langue a suscité un intérêt grandissant. Bien que l'entraînement des modèles de langue sur des données numériques soit devenu un paradigme courant, les modèles actuels ne parviennent pas à effectuer des calculs de manière satisfaisante. Pour y remédier, une solution est d'entraîner les modèles de langue à utiliser des outils externes tels qu'une calculatrice ou un "runtime" de code python pour effectuer le raisonnement numérique. L'objectif de ce papier est double, dans un premier temps, nous passons en revue les travaux de l'état de l'art sur le raisonnement numérique dans les modèles de langue et dans un second temps nous discutons des différentes perspectives de recherche pour augmenter les compétences numériques des modèles.

ABSTRACT

Integrating numerical reasoning into language models : State of the art.

In recent years, language models have undergone a rapid evolution thanks to the increase in computing power that has made the use of neural networks possible. At the same time, the integration of numerical reasoning in language models has attracted growing interest. Thus, training language models on numerical data has become a common paradigm, although current models have not been able to perform well in terms of calculations. To overcome this limitation, one solution is to train language models to use external tools such as a calculator or a python code runtime to perform numerical reasoning. This paper has two objectives, first we review state of the art work that has incorporated numerical reasoning and second we discuss different research perspectives to increase the numerical performance of language models.

MOTS-CLÉS : Modèles de langue, raisonnement numérique, question réponses.

KEYWORDS: Language models, numerical reasoning, question answering.

1 Introduction

Le traitement automatique du langage naturel (TAL) a été boulversé ces dernières années grâce aux Transformers (Vaswani *et al.*, 2017). Cette architecture permet d'encoder efficacement le sens des

mots et des paragraphes avec leur contexte dans un espace vectoriel de grande dimension à travers un mécanisme d'attention qui focalise le modèle sur certains groupes de mots. Ce mécanisme a démontré des performances importantes dans de nombreuses tâches de TAL comme les tâches de séquence à séquence (e.g., traduction automatique (Luong *et al.*, 2015), résumé de documents (Rush *et al.*, 2015)) et les tâches de classification (e.g., l'analyse de sentiments (Basiri *et al.*, 2021), l'étiquetage des parties du discours ou la catégorisation thématique (Cheng *et al.*, 2019)). Cette avancée a permis aux modèles de langue pré-entraînes (*Pre Trained Language Models*, PTLM) de comprendre les nuances et les subtilités du langage naturel devenant ainsi des outils universels (Howard & Ruder, 2018) utilisés dans de nombreux domaines (Araci, 2019; Steinberg *et al.*, 2021; Chen *et al.*, 2022; Borsos *et al.*, 2022). Un exemple marquant est le système conversationnel ChatGPT qui est conçu pour répondre à des questions très diverses et résoudre toute une série de problèmes en communiquant aux utilisateurs la réponse d'une manière semblable à celle d'un être humain (Jiao *et al.*, 2023; Lund & Wang, 2023). BLOOM, avec 176 milliards de paramètres, est capable de générer du texte dans 46 langues et 13 langages de programmation (Scao *et al.*, 2022) tandis que LaMDA saisit des nuances fines distinguant le dialogue des autres formes de langage (Thoppilan *et al.*, 2022). Aujourd'hui, grâce aux PTLM, souvent il suffit d'étiqueter une petite quantité de données et d'y ajuster un modèle de langue pré-entraîné pour résoudre un problème (Houlsby *et al.*, 2019), en effet, celui-ci ayant déjà acquis une quantité importante des connaissances en TAL au préalable. Le succès des PTLM est également lié à l'explosion des données textuelles disponibles sur internet, qui ont permis de pré-entraîner ces modèles sur des corpus de plus en plus vastes et variés incluant l'intégralité de Wikipédia, Reddit et de nombreuses sources d'informations. Par exemple, le corpus C4 (Raffel *et al.*, 2020) atteint aujourd'hui la taille record de 800 Go de textes et de méta données. Cependant, les compétences de haut niveau, telles que la capacité à effectuer un raisonnement numérique sur du texte, restent un verrou lorsque ces modèles sont appris seulement avec un objectif de masquage des mots ou des générations de phrases contextuelles (Dua *et al.*, 2019; Andor *et al.*, 2019; Chen *et al.*, 2020).

Le raisonnement numérique est une tâche critique qui intervient dans divers scénarios allant du shopping à la modélisation du climat (Lithner, 2000). Il s'agit de traiter des opérations numériques telles que l'addition ou la multiplication et d'interpréter les tendances et les relations numériques (Cobbe *et al.*, 2021). En effet, des données numériques sont présentes dans la grande majorité des documents (finance, santé, journalisme, etc) (Gelman & Butterworth, 2005), d'où le besoin d'avoir des modèles de langue compétents en raisonnement numérique (Dua *et al.*, 2019). Néanmoins, le raisonnement numérique sur du texte est une tâche particulièrement difficile (Al-Negheimish *et al.*, 2021; Mialon *et al.*, 2023) car le modèle doit 1) comprendre les scénarios complexes décrits dans les textes des problèmes, 2) trouver l'enchaînement des opérations nécessaires, 3) identifier les variables mathématiques et associer le texte à la logique des équations mathématiques, 4) projeter les données vers un espace sémantique (ou un système) différent pour effectuer les calculs et enfin 5) produire le résultat attendu et/ou générer la réponse en langage naturel.

La figure 1 montre un exemple de questions à partir de DROP (Dua *et al.*, 2019). Pour répondre à la question "How many years after founding of Hughes/ Donahue was Art Euphoric founded ?", le modèle doit d'abord comprendre le contexte décrit et identifier les années de création des deux entreprises (Hughes/Donahue et Art Euphoric). Ensuite, il doit trouver l'opération à effectuer, dans ce cas c'est une soustraction pour enfin générer la réponse qui est un.

Dans cet article nous nous intéressons à la problématique d'intégration du raisonnement numérique dans les modèles de langue. En d'autres terme, l'enjeu est de résoudre des problèmes mathématiques décrit en langage textuel en utilisant les modèles de langues.

FIGURE 1 – Exemple de question de DROP (Dua *et al.*, 2019) nécessitant un raisonnement numérique

Passage : Taunton has four art galleries... **Hughes/Donahue** Gallery founded in **2007**, a local community gallery serving local Taunton artists... **Art Euphoric** founded in **2008** has both visual and craft exhibits...
Question : How many years after founding of Hughes/ Donahue was Art Euphoric founded?
Réponse : **2008 - 2007 = 1**

1. Nous proposons un état de l'art des différentes approches d'incorporation des compétences numériques dans les modèles de langues.
2. Nous discutons des perspectives de recherche pour construire des modèles capables d'identifier un enchaînement d'opérations numériques et d'extraire toutes les informations nécessaires pour l'effectuer.

2 État de l'art

Nous présentons les principales approches permettant aux modèles d'effectuer des raisonnements numériques sur du texte. Nous distinguerons d'une part les architectures spécifiques basées sur des modules de raisonnement ou de prédiction et, d'autre part, les larges modèles de langues pré-entraînés pour effectuer des calculs ou pour générer des programmes de raisonnement permettant d'arriver à la réponse. Pour évaluer la capacité de ces modèles à résoudre des questions nécessitant un calcul arithmétique, les approches actuelles reposent sur la métrique "Exact Match". Ainsi, il est question de générer une réponse pour la comparer à la réponse correcte.

2.1 Architectures spécifiques au raisonnement numérique

Les premiers modèles développés pour résoudre des problèmes de raisonnement numérique à partir de texte utilisent des architectures spécialisées dotées de modules de raisonnement.

Le modèle NAQANet, basé sur le modèle QANet (Yu *et al.*, 2018), a été proposé pour intégrer le raisonnement à travers un module de prédiction permettant la génération de quatre types de réponses : l'extraction de texte, les comptages, les additions ou les soustractions de nombres sous forme d'expression arithmétique. De nombreux travaux ont suivi la même approche pour la génération des réponses tels que (Ran *et al.*, 2019; Geva *et al.*, 2020; Chen *et al.*, 2020; Zhou *et al.*, 2022). Parmi ces approches, plusieurs sont basées sur des graphes pour modéliser les dépendances entre valeurs numériques. Par exemple, NumNet de (Ran *et al.*, 2019), construit un graphe à partir des nombres mentionnés dans la question et le passage textuel. Ces derniers représentent les noeuds du graphe et leurs relations encodent les comparaisons entre eux. L'architecture de NumNet atteint de meilleures performances que le modèle NAQANet en développant un cadre qui permet une comparaison numérique des nombres. Néanmoins, de nombreuses questions impliquent la prise en compte d'un nombre intermédiaire n'apparaissant pas littéralement dans le document ou la question, ce qui limite la performance de NumNet. La table 1 montre un exemple de ce type de questions à partir de DROP (Dua *et al.*, 2019), le modèle doit d'abord effectuer des additions pour trouver le pourcentage de personnes ayant un âge supérieur à 40 et le pourcentage de personne ayant un âge inférieur à 19 pour ensuite les comparer afin de retourner la catégorie ayant le plus grand pourcentage. A droite, la réponse générée par NumNet est une preuve de l'inefficacité de ce modèle.

Dans une approche plus élaborée, (Chen *et al.*, 2020) utilisent un graphe orienté qui encode les

TABLE 1 – Exemple de question de DROP (Dua *et al.*, 2019) nécessitant un calcul intermédiaire

Question	Passage	Réponse	NumNet
Were more people 40 and older or 19 and younger?	Of Saratoga Countys population in 2010, 6.3% were between ages of 5 and 9 years, 6.7% between 10 and 14 years, 6.5% between 15 and 19 years, ... , 7.9% between 40 and 44 years, 8.5% between 45 and 49 years, 8.0% between 50 and 54 years, 7.0% between 55 and 59 years, 6.4% between 60 and 64 years, and 13.7% of age 65 years and over ...	40 and older	19 and younger

relations entre les nombres et les entités mentionnés dans le contexte et la question. Dans l'exemple de la table1, les nombres 5 et 9 sont du même type (age) et ils sont liés à l'entité *year*. Le modèle construit un réseau d'attention du graphe qui incorpore l'encodage contextuel de la question dans le processus de raisonnement.

Le développement de structures spécialisées destinées au raisonnement numérique est un premier pas important dans l'amélioration des compétences numériques des modèles. Toutefois, ces modèles présentent des limitations en raison de leur incapacité à effectuer des calculs complexes. Par exemple, certains modèles ont été conçus pour compter jusqu'à neuf ou pour effectuer des opérations telles que l'addition et la soustraction seulement à partir des nombres vus en apprentissage : si le résultat implique plus de valeurs neuves ou que les nombres n'apparaissent pas dans les données d'entraînement le modèles est incapable d'y répondre (Dua *et al.*, 2019; Ran *et al.*, 2019; Chen *et al.*, 2020; Zhou *et al.*, 2022). L'étude expérimentale réalisée par (Al-Negheimish *et al.*, 2021) pour tester les compétences numériques des modèles de langue a montré que les modèles précédents ne parviennent pas nécessairement à la bonne réponse en exploitant les bonnes informations. Ces expérimentations ont mis en lumière des lacunes de ces modèles : ces approches sont souvent capables de répondre aux questions sans avoir accès aux contextes ! Ils captent des motifs récurrents dans le jeu de données pour extrapoler la réponse : ils sont donc biaisés par rapport au format des passages et des questions ou à la distribution des réponses. Ainsi, il existe une grande disparité de performance entre les questions dont les réponses sont les plus fréquentes par rapport aux autres.

2.2 Modèles de langues pré-entraînés augmentés de compétences numériques

Dans cette section, nous nous intéresserons aux travaux explorant le pré-entraînement des modèles de langue afin de générer les résultats numériques souhaités directement, à travers des techniques comme l'augmentation de données textuelles ou numériques (Geva *et al.*, 2020; Yang *et al.*, 2021), le pré-entraînement séquentiel sur plusieurs jeux de données ou l'utilisation de programmes et de leurs résultats comme données d'entraînement (codes, requêtes, etc.) comme dans (Pi *et al.*, 2022).

GENBERT de (Geva *et al.*, 2020) est une amélioration directe du modèle BERT (Devlin *et al.*, 2018). Le module de prédiction est similaire à celui de NaQANet, où chaque type de réponse attendue est traité séparément. Le modèle est d'abord pré-entraîné sur un objectif classique de modélisation de langue puis sur des données numériques –sous forme d'expressions arithmétiques associées à leurs résultats– pour apprendre au modèle à effectuer des calculs. Enfin, le modèle est spécialisé sur des questions-réponses pour apprendre à générer les sorties attendues. Le modèle pré-entraîné se veut générique : il peut être adapté à différents jeux de données tel que DROP (Dua *et al.*, 2019). GENBERT atteint des performances similaires à l'état de l'art sur DROP (à taille de modèle comparable) et il s'adapte aux jeux de données de problèmes mathématiques (MWP) (Koncel-Kedziorski *et al.*, 2016; Hosseini *et al.*, 2014; Roy *et al.*, 2015; Koncel-Kedziorski *et al.*, 2015), tout en maintenant des

performances élevées sur SQuAD (Rajpurkar *et al.*, 2016).

Dans le même contexte, (Yang *et al.*, 2021) comparent cinq pipelines d'entraînement séquentiel qui adaptent un modèle T5 pré-entraîné pour le raisonnement numérique sur du texte. Chaque pipeline comprend deux étapes : pré-entraînement sur des données de raisonnement numérique et de compréhension générale de texte, suivi d'un réglage fin sur DROP (Dua *et al.*, 2019) et d'une tâche de classification dérivée de DROP où il s'agit de classifier le type de la réponse. L'entraînement multi-tâches est adapté à chaque étape en utilisant différents ensembles de données (données numériques et textuelles synthétiques (Geva *et al.*, 2020), DROP et SQuAD).Le pipeline d'apprentissage séquentiel nécessite peu de ressources, il permet de tester différentes hypothèses et donne de bonnes performances même en utilisant des modèles de taille limitée (T5-small). Cette approche a permis de réaliser de grandes améliorations en termes de raisonnement numérique par rapport à un modèle T5 de base.

Dans une autre optique, POET "Program Executor" (Pi *et al.*, 2022) est un nouveau paradigme de pré-entraînement qui permet aux modèles de langue d'acquérir les *capacités de raisonnement* des exécuteurs de programmes en s'entraînant sur des données composées de programmes associées à leurs résultats d'exécution. POET est conceptuellement simple et peut être instancié sur différents types de programmes : POET-Math, POET-Logic et POET-SQL. L'idée est que les modèles de langues, pour prédire le résultat d'un programme, apprennent à imiter les procédures d'exécution de programmes : ils pourraient ensuite potentiellement apprendre plus facilement les mécanismes de raisonnement logique que les humains ont adoptées pour créer l'exécuteur de programme. POET a été testé avec succès à partir de BART et RoBERTa, après une phase d'adaptation, sur DROP, HotpotQA (Yang *et al.*, 2018), TAT-QA (Zhu *et al.*, 2021) et EQUATE (Ravichander *et al.*, 2019).

Dans cette classe de modèles, la majorité des erreurs se produisent sur des exemples qui exigent des compétences de raisonnement qui ne sont pas couvertes par l'ensemble de données de pré-entraînement (Geva *et al.*, 2020; Yang *et al.*, 2021). Bien que POET ait appris des compétences de raisonnement des exécuteurs de programmes, il ne peut répondre qu'aux tâches similaires à celles de l'ensemble de données de pré-entraînement. Pour combler les lacunes de ces modèles, une option serait d'augmenter l'ensemble de données de pré-entraînement pour élargir l'éventail des raisonnements complexes assimilés et mieux répondre aux questions des jeux de données. Malgré une approche assez générique, le problème de la généralisation aux nouvelles données reste un défi.

2.3 Modèles de langues pour la génération de programmes de raisonnement

Dans cette section, nous nous intéressons à une troisième voie émergeante qui consiste à générer un ensemble d'instructions –ie un programme– qui sera interprété par un calculateur externe. La production du résultat final (e.g. manipulation des nombres, calculs) est ainsi déléguée à un outil existant, comme présenté dans la figure 2. (Mialon *et al.*, 2023) ont effectué une étude très large sur l'augmentation des modèles de langues avec des compétences basées des outils externes.

Le modèle présenté dans (Andor *et al.*, 2019) permet au modèle BERT d'effectuer un raisonnement numérique léger avec un pré-entraînement sur DROP (Dua *et al.*, 2019). BERT est augmenté avec un ensemble prédéfini de programmes exécutables qui englobent l'arithmétique simple ainsi que l'extraction de texte. Plutôt que d'avoir à apprendre à manipuler les nombres directement (Geva *et al.*, 2020; Yang *et al.*, 2021; Pi *et al.*, 2022; Xue *et al.*, 2022), le modèle peut générer un programme et l'exécuter. Les programmes sont simples de la forme *Opération(argument, ...)* parmi l'espace des dérivations défini par les auteurs. Cet espace comprend des expressions (yes, no, unkown, 0,...,9), des opérations numériques (diff, mul, div, sum et diff100), des opérations d'extraction de texte (span)

FIGURE 2 – Modèles de langues et modèles de langues augmentés d'outils, (Parisi *et al.*, 2022).

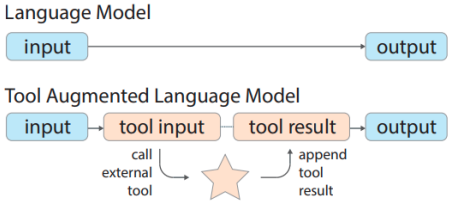
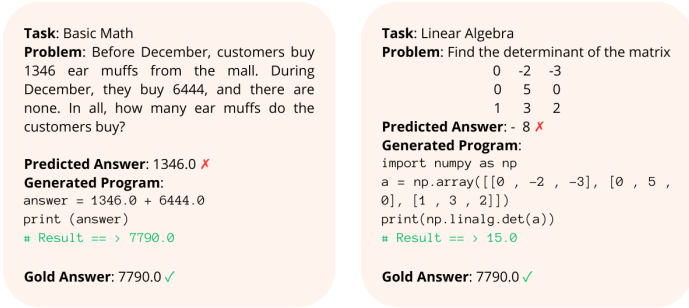


FIGURE 3 – Exemples avec Bhaskara en calcul et algèbre sur le jeu de données LILA (Mishra *et al.*, 2022). Le programme est souvent plus pertinent que l'estimation numérique.



et des compositions (merge, sum3). Une fois l'opération arithmétique générée, le calcul est exporté vers un calculateur externe. Cette approche dépasse l'état de l'art sur certaines tâches spécifiques, néanmoins, elle est très préliminaire dans le domaine et ne couvre pas les calcul numériques avancés.

Dans le même sens, (Mishra *et al.*, 2022) explore les performances de deux types de modèles nommés BHASKARA (P et A). Le premier génère des programmes python qui sont ensuite exécutés afin de prédire le résultat numérique attendu (BHASKARA-P) : le modèle n'est pas entraîné pour exécuter les calculs en interne mais plutôt pour planifier et générer les différentes étapes qui permettront d'atteindre le résultat souhaité. Le second modèle est entraîné pour produire le résultat final directement (BHASKARA-A), tous les calculs se font en interne dans le modèle. La figure 3 montre des exemples de programmes générés et de résultats estimés par BHASKARA. Les résultats de l'expérimentation ont montré que la génération de programme surpasse nettement la prédiction directe des résultats. L'étude de ce travail nous a permis de conclure que les modèles de langue manipulent mieux les opérateurs de haut niveau et leur enchaînement que les calculs numériques directs. De plus, le modèle est capable d'appeler des bibliothèques externes pour effectuer des calculs avancés de manière pertinente. Par exemple, le modèle utilise `scipy.stats.entropy` ou `np.linalg.det` respectivement en statistique et en algèbre linéaire lors de la résolution de problèmes.

Très récemment, (Schick *et al.*, 2023) ont introduit Toolformer, un modèle qui a été entraîné pour apprendre à utiliser différents outils externes tels qu'une calculatrice, un système de questions-réponses, un moteur de recherche, un système de traduction et un calendrier. Le modèle est capable de décider quel outil appeler, quand l'appeler, avec quels arguments lancer la requête et comment incorporer au mieux les résultats dans la prédiction future de texte. Toolformer ne peut pas (encore) gérer les calculs complexes en raison de la nature des exemples utilisés pour l'entraîner à exploiter la calculatrice. L'ensemble de données textuelles ne contient que des documents plutôt simples, qui remplissent l'une des conditions suivantes : (i) contiennent au moins trois nombres dans une fenêtre de 100 mots, où l'un de ces nombres est le résultat d'une opération mathématique appliquée aux deux

autres, (ii) contiennent l'une des séquences "=", "égale", "égale à", "total de", "moyenne de" suivie d'un nombre, ou (iii) contiennent au moins trois nombres. Il s'agit néanmoins d'une proposition intéressante qui ouvre beaucoup de perspectives.
Avec une philosophie très proche, (Lyu *et al.*, 2023) propose Faithful CoT *Chain of Thought*. Il s'agit d'un cadre de prompting décomposant une tâche de raisonnement en deux étapes : 1) la traduction (requête en langage naturel → chaîne de raisonnement symbolique) et 2) la résolution du problème (chaîne de raisonnement → réponse), en utilisant respectivement un PTLM pour générer la chaîne de raisonnement et un outil pour l'exécuter.

3 Discussion et perspectives

Dans cette section, nous discutons des perspectives de recherche que nous envisageons pour proposer des modèles de langues dotés de compétences numériques. L'idée de planifier et générer les différentes étapes du raisonnement nous semble très prometteuse (Andor *et al.*, 2019; Mishra *et al.*, 2022). En effet, le modèle de langue est très performant sur cette tâche tandis que ses faiblesses calculatoires peuvent être palliées en faisant appel à un calculateur externe. L'enjeu est alors d'apprendre aux modèles de langage à utiliser des outils externes (Parisi *et al.*, 2022; Schick *et al.*, 2023). La richesse des outils et bibliothèques existants pour des problèmes très divers ouvre de nombreuses perspectives. Accessoirement le comportement des modèles de langue se rapprocherait alors de celui des humains qui font appel quotidiennement à une large palette d'outils pour résoudre des problèmes divers.

3.1 Capacité des modèles de langues à générer des chaînes de raisonnement

Actuellement, les modèles de langage ont des lacunes sur le raisonnement numérique et sont incapables d'effectuer des calculs complexes. Une idée serait de se concentrer sur la capacité de génération de la chaîne de raisonnement, avec et sans transfert, en déléguant le calcul numérique à des opérateurs externes. Avec transfert en ajustant les modèles de langues de génération de code disponibles sur les datasets de raisonnement comme DROP et sans transfert en utilisant les techniques de prompting. La catégorisation des raisonnements et la supervision de ces chaînes de raisonnement sont réalisables à partir des jeux de données existants et pourraient bénéficier de l'augmentation de données. Une clé réside dans la capacité à créer des fonctions de coût intermédiaires caractérisant les enchaînements locaux d'opérations, à la manière de ce qui existe en apprentissage par renforcement. Le résultat final dépendant directement de calculateur externe serait alors utilisé dans les métriques d'évaluation plus que dans la stratégie d'apprentissage elle-même.

La génération des chaînes de raisonnement ou processus de raisonnement sous formes de codes Python entre parfaitement dans le cadre défini précédemment : 1) un code est essentiellement un enchaînement d'instructions sous forme de texte, la modalité idéale pour les modèles de langue. Il existe d'ailleurs déjà des modèles de langues entraînés sur du code qui pourraient servir de base à ce travail (Black *et al.*, 2022; Nijkamp *et al.*, 2022). 2) Il est possible de résoudre toute sorte de problèmes en utilisant les langages de programmation avec des outils de plus ou moins haut niveau selon les bibliothèques considérées. Ainsi, il s'agit d'un cadre pertinent pour étudier les capacités de généralisation en raisonnement numérique. Contrairement à Toolformer (Schick *et al.*, 2023) qui augmente les PTLMs en utilisant plusieurs outils, nous nous concentrerons sur la partie raisonnement numérique directement dans le langage de programmation : nos outils seront des fonctions.

Au bout de la chaîne, pour pouvoir produire le texte présentant les résultats des raisonnements

numériques générés, nous pensons repartir de la littérature data-to-text : l'enjeu est d'abord de placer le résultat calculé dans une phrase pertinente vis-à-vis de la question et du contexte puis d'exploiter la génération de texte pour expliquer les grandes étapes du raisonnement construit précédemment.

3.2 Application à des tâches de génération de texte à partir de données structurées ("data-to-text generation")

Le raisonnement est présent implicitement dans de nombreuses tâches de TAL (question-réponses, génération de texte, ...). En *Data-to-Text* en particulier, l'enjeu est crucial : la tâche se définit comme une sorte de traduction depuis des données complexes, parfois structurées, souvent numériques vers des descriptions textuelles plus compréhensibles par les humains. Ces données sont généralement partiellement ou complètement numériques : séries temporelles provenant de capteurs, tableaux de valeurs, résultats de requêtes SQL, bases de connaissances et graphes, etc. C'est un champ émergent très dynamique dans le domaine du traitement du langage naturel, (Wiseman *et al.*, 2018; Zhang *et al.*, 2019; Kale & Rastogi, 2020; Parikh *et al.*, 2020; Rebuffel *et al.*, 2022), possédant de très nombreuses applications, notamment dans les domaines scientifiques, du journalisme, de la santé, du marketing, de la finance, etc. La plupart des applications dans ce domaine nécessitent de la planification, de l'analyse numérique ou du raisonnement : cet enjeu est en passe de devenir central en *Data-to-Text* (Nie *et al.*, 2018; Herzig *et al.*, 2020).

Appliquer les techniques issues de la bibliographie présentée dans cet article sur des entrées tabulaires (ou autre) semble très pertinent. En effet, l'enjeu est alors de réaliser des opérations entre des entités, des valeurs ou sur des séries de valeurs (e.g., identification de la plus grande valeur sur une colonne du tableau, recherche du joueur ayant fait le plus de passes, etc...). L'idée de faire appel à des outils de calculs externes dans le domaine du *Data-to-Text* est à la fois originale et pertinente pour aider à la résolution de cette tâche difficile. Des chaînes de raisonnement pourraient être générées puis exécutées par des outils externes lors de l'inférence, nourissant un modèle de langue chargé de générer le texte final, comme dans la proposition émise en fin de section précédente. En utilisant des outils externes pour gérer les aspects numériques, les modèles de langue pourraient se concentrer davantage sur les aspects linguistiques et sémantiques, ce qui pourrait conduire à des améliorations significatives de la qualité des sorties produites.

4 Conclusion

Cet article s'intéresse à l'amélioration des modèles de langue en raisonnement numérique. Bien que le pré-entraînement sur des données numériques soit maintenant une approche courante, les résultats ne sont pas satisfaisants. Notre synthèse de l'état de l'art des approches d'intégration du raisonnement numérique dans les modèles de langue penche clairement en faveur des architectures hybrides mêlant génération de texte et calculateurs externes. Nous sommes convaincus du potentiel de ces modèles, à la fois en raisonnement automatique à partir d'énoncés textuels et dans le cadre des applications de *Data-to-text*. Nous travaillons au développement d'architectures capables de traduire ces problématiques en code Python, d'exécuter ces programmes puis d'incorporer les résultats retournés dans un texte pertinent.

Remerciements Ce travail est effectué dans le cadre du projet ANR PRCE ACDC (ANR-21-CE23-0007)

Références

AL-NEGHEIMISH H., MADHYASTHA P. & RUSSO A. (2021). Numerical reasoning in machine reading comprehension tasks : are we there yet ? *arXiv preprint arXiv :2109.08207*.

ANDOR D., HE L., LEE K. & PITLER E. (2019). Giving bert a calculator : Finding operations and arguments with reading comprehension. *arXiv preprint arXiv :1909.00109*.

ARACI D. (2019). Finbert : Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv :1908.10063*.

BASIRI M. E., NEMATI S., ABDAR M., CAMBRIA E. & ACHARYA U. R. (2021). Abcdm : An attention-based bidirectional cnn-rnn deep model for sentiment analysis. *Future Generation Computer Systems*, **115**, 279–294.

BLACK S., BIDERMAN S., HALLAHAN E., ANTHONY Q., GAO L., GOLDING L., HE H., LEAHY C., MCDONELL K., PHANG J. *et al.* (2022). Gpt-neox-20b : An open-source autoregressive language model. *arXiv preprint arXiv :2204.06745*.

BORSOS Z., MARINIER R., VINCENT D., KHARITONOV E., PIETQUIN O., SHARIFI M., TEBOUL O., GRANGIER D., TAGLIASACCHI M. & ZEGHIDOUR N. (2022). Audiolm : a language modeling approach to audio generation. *arXiv preprint arXiv :2209.03143*.

CHEN J., GUO H., YI K., LI B. & ELHOSEINY M. (2022). Visualgpt : Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 18030–18040.

CHEN K., XU W., CHENG X., XIAOCHUAN Z., ZHANG Y., SONG L., WANG T., QI Y. & CHU W. (2020). Question directed graph attention network for numerical reasoning over text. *arXiv preprint arXiv :2009.07448*.

CHENG Y., YE Z., WANG M. & ZHANG Q. (2019). Document classification based on convolutional neural network and hierarchical attention network. *Neural Network World*, **29**(2), 83–98.

COBBE K., KOSARAJU V., BAVARIAN M., CHEN M., JUN H., KAISER L., PLAPPERT M., TWOREK J., HILTON J., NAKANO R. *et al.* (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv :2110.14168*.

DEVLIN J., CHANG M.-W., LEE K. & TOUTANOVA K. (2018). Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*.

DUA D., WANG Y., DASIGI P., STANOVSKY G., SINGH S. & GARDNER M. (2019). Drop : A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv :1903.00161*.

GELMAN R. & BUTTERWORTH B. (2005). Number and language : how are they related ? *Trends in cognitive sciences*, **9**(1), 6–10.

GEVA M., GUPTA A. & BERANT J. (2020). Injecting numerical reasoning skills into language models. *arXiv preprint arXiv :2004.04487*.

HERZIG J., NOWAK P. K., MÜLLER T., PICCINNO F. & EISENSCHLOS J. (2020). TaPas : Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 4320–4333, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2020.acl-main.398.

HOSSEINI M. J., HAJISHIRZI H., ETZIONI O. & KUSHMAN N. (2014). Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical*

Methods in Natural Language Processing (EMNLP), p. 523–533, Doha, Qatar : Association for Computational Linguistics. DOI : 10.3115/v1/D14-1058.

HOULSBY N., GIURGIU A., JASTRZEBSKI S., MORRONE B., DE LAROUSSILHE Q., GESMUNDO A., ATTARIYAN M. & GELLY S. (2019). Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, p. 2790–2799 : PMLR.

HOWARD J. & RUDER S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv :1801.06146*.

JIAO W., WANG W., HUANG J.-T., WANG X. & TU Z. (2023). Is chatgpt a good translator ? a preliminary study. *arXiv preprint arXiv :2301.08745*.

KALE M. & RASTOGI A. (2020). Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, p. 97–102, Dublin, Ireland : Association for Computational Linguistics.

KONCEL-KEDZIORSKI R., HAJISHIRZI H., SABHARWAL A., ETZIONI O. & ANG S. D. (2015). Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, **3**, 585–597. DOI : 10.1162/tacl_a_00160.

KONCEL-KEDZIORSKI R., ROY S., AMINI A., KUSHMAN N. & HAJISHIRZI H. (2016). MAWPS : A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 1152–1157, San Diego, California : Association for Computational Linguistics. DOI : 10.18653/v1/N16-1136.

LITHNER J. (2000). Mathematical reasoning in task solving. *Educational studies in mathematics*, p. 165–190.

LUND B. D. & WANG T. (2023). Chatting about chatgpt : how may ai and gpt impact academia and libraries ? *Library Hi Tech News*.

LUONG M.-T., PHAM H. & MANNING C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv :1508.04025*.

LYU Q., HAVALDAR S., STEIN A., ZHANG L., RAO D., WONG E., APIDIANAKI M. & CALLISON-BURCH C. (2023). Faithful chain-of-thought reasoning. *arXiv preprint arXiv :2301.13379*.

MIALON G., DESSÌ R., LOMELI M., NALMPANTIS C., PASUNURU R., RAILEANU R., ROZIÈRE B., SCHICK T., DWIVEDI-YU J., CELIKYILMAZ A. *et al.* (2023). Augmented language models : a survey. *arXiv preprint arXiv :2302.07842*.

MISHRA S., FINLAYSON M., LU P., TANG L., WELLECK S., BARAL C., RAJPUROHIT T., TAFJORD O., SABHARWAL A., CLARK P. *et al.* (2022). Lila : A unified benchmark for mathematical reasoning. *arXiv preprint arXiv :2210.17517*.

NIE F., WANG J., YAO J.-G., PAN R. & LIN C.-Y. (2018). Operation-guided neural networks for high fidelity data-to-text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 3879–3889, Brussels, Belgium : Association for Computational Linguistics. DOI : 10.18653/v1/D18-1422.

NIJKAMP E., PANG B., HAYASHI H., TU L., WANG H., ZHOU Y., SAVARESE S. & XIONG C. (2022). Codegen : An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv :2203.13474*.

PARIKH A., WANG X., GEHRMANN S., FARUQUI M., DHINGRA B., YANG D. & DAS D. (2020). ToTTo : A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on*

Empirical Methods in Natural Language Processing (EMNLP), p. 1173–1186, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2020.emnlp-main.89.

PARISI A., ZHAO Y. & FIEDEL N. (2022). Talm : Tool augmented language models. *arXiv preprint arXiv :2205.12255*.

PI X., LIU Q., CHEN B., ZIYADI M., LIN Z., GAO Y., FU Q., LOU J.-G. & CHEN W. (2022). Reasoning like program executors. *arXiv preprint arXiv :2201.11473*.

RAFFEL C., SHAZEER N., ROBERTS A., LEE K., NARANG S., MATENA M., ZHOU Y., LI W. & LIU P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, **21**(1), 5485–5551.

RAJPURKAR P., ZHANG J., LOPYREV K. & LIANG P. (2016). Squad : 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv :1606.05250*.

RAN Q., LIN Y., LI P., ZHOU J. & LIU Z. (2019). Numnet : Machine reading comprehension with numerical reasoning. *arXiv preprint arXiv :1910.06701*.

RAVICHANDER A., NAIK A., ROSE C. & HOVY E. (2019). EQUATE : A benchmark evaluation framework for quantitative reasoning in natural language inference. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, p. 349–361, Hong Kong, China : Association for Computational Linguistics. DOI : 10.18653/v1/K19-1033.

REBUFFEL C., ROBERTI M., SOULIER L., SCOUTHEETEN G., CANCELLIERE R. & GALLINARI P. (2022). Controlling hallucinations at word level in data-to-text generation. *Data Mining and Knowledge Discovery*, p. 1–37.

ROY S., VIEIRA T. & ROTH D. (2015). Reasoning about Quantities in Natural Language. *Transactions of the Association for Computational Linguistics*, **3**, 1–13. DOI : 10.1162/tacl_a_00118.

RUSH A. M., CHOPRA S. & WESTON J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv :1509.00685*.

SCAO T. L., FAN A., AKIKI C., PAVLICK E., ILIĆ S., HESSLOW D., CASTAGNÉ R., LUCCIONI A. S., YVON F., GALLÉ M. *et al.* (2022). Bloom : A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv :2211.05100*.

SCHICK T., DWIVEDI-YU J., DESSÌ R., RAILEANU R., LOMELI M., ZETTLEMOYER L., CANCEDDA N. & SCIALOM T. (2023). Toolformer : Language models can teach themselves to use tools. DOI : 10.48550/ARXIV.2302.04761.

STEINBERG E., JUNG K., FRIES J. A., CORBIN C. K., PFOHL S. R. & SHAH N. H. (2021). Language models are an effective representation learning technique for electronic health record data. *Journal of biomedical informatics*, **113**, 103637.

THOPPILAN R., DE FREITAS D., HALL J., SHAZEER N., KULSHRESHTHA A., CHENG H.-T., JIN A., BOS T., BAKER L., DU Y. *et al.* (2022). Lamda : Language models for dialog applications. *arXiv preprint arXiv :2201.08239*.

VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER Ł. & POLOSUKHIN I. (2017). Attention is all you need. *Advances in neural information processing systems*, **30**.

WISEMAN S., SHIEBER S. & RUSH A. (2018). Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 3174–3187, Brussels, Belgium : Association for Computational Linguistics. DOI : 10.18653/v1/D18-1356.

XUE L., BARUA A., CONSTANT N., AL-RFOU R., NARANG S., KALE M., ROBERTS A. & RAFFEL C. (2022). Byt5 : Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, **10**, 291–306.

YANG P.-J., CHEN Y. T., CHEN Y. & CER D. (2021). Nt5 ?! training t5 to perform numerical reasoning. *arXiv preprint arXiv :2104.07307*.

YANG Z., QI P., ZHANG S., BENGIO Y., COHEN W. W., SALAKHUTDINOV R. & MANNING C. D. (2018). Hotpotqa : A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv :1809.09600*.

YU A. W., DOHAN D., LUONG M.-T., ZHAO R., CHEN K., NOROUZI M. & LE Q. V. (2018). Qanet : Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv :1804.09541*.

ZHANG L., ZHANG S. & BALOG K. (2019). Table2vec : Neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, p. 1029–1032.

ZHOU Y., BAO J., DUAN C., SUN H., LIANG J., WANG Y., ZHAO J., WU Y., HE X. & ZHAO T. (2022). Opera : Operation-pivoted discrete reasoning over text. *arXiv preprint arXiv :2204.14166*.

ZHU F., LEI W., HUANG Y., WANG C., ZHANG S., LV J., FENG F. & CHUA T.-S. (2021). TAT-QA : A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, p. 3277–3287, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2021.acl-long.254.