Vers des LLM moins gourmands pour interroger des graphes de connaissances en langue naturelle

Marion Schaeffer*,** Christophe Bouvard*
Jean-Philippe Kotowicz** Cecilia Zanni-Merk**

*Wikit
prenom@wikit.ai, https://www.wikit.ai/
**LITIS - INSA Rouen Normandie
prenom.nom@insa-rouen.fr, https://www.litislab.fr/

Résumé. Les graphes de connaissances sont largement utilisés pour définir et représenter un domaine. Ils permettent de réunir des données issues de diverses sources et de les stocker de façon structurée. L'accès à ces ressources peut cependant s'avérer complexe car il faut maîtriser le langage de requête compatible avec le formalisme utilisé pour représenter le graphe de connaissances. Dans cet article, nous proposons un système conversationnel de questions-réponses utilisant les grands modèles de langage pour interroger des graphes de connaissances en langue naturelle. Nous comparons différentes techniques afin de combiner au mieux frugalité et performance dans le but de maîtriser les ressources utilisées. Notre travail se base sur un graphe de connaissances décrivant les Objectifs du Développement Durables définis par les Nations Unies.

1 Introduction

Un graphe de connaissances est défini par Hogan et al. (2020) comme un graphe de données destiné à accumuler et à transmettre des connaissances sur le monde réel, dont les nœuds représentent des entités d'intérêt et dont les arêtes représentent des relations entre ces entités. Ils constituent une riche source d'informations et permettent de construire diverses applications, comme celles listées par Hogan et al. (2020). Le schéma définit la structure de haut niveau du graphe de connaissances et permet de lier des données issues de sources variées.

Malgré leur utilité, les graphes de connaissances sont difficilement accessibles pour des utilisateur·rice·s finaux·ales car ils nécessitent de maîtriser le langage de requête du formalisme dans lequel ils sont construits. C'est pour cela que des systèmes de questions-réponses (*Question Answering*, QA) ont été largement développés, comme expliqué par Liang et al. (2020). Ils permettent aux utilisateur·rice·s finaux·ales de poser leur question directement en langue naturelle. Nous nous intéressons aux systèmes de QA sur des domaines fermés utilisant des graphes de connaissances spécifiques à un domaine, contrairement aux systèmes de QA en domaine ouvert qui utilisent plutôt des graphes de connaissances génériques comme DBpedia, Lehmann et al. (2015), ou Wikidata, Vrandečić et Krötzsch (2014).

Un cas concret de cette difficulté d'accès à l'information est le SustainGraph, proposé par Fotopoulou et al. (2022), qui regroupe des connaissances autour des Objectifs du Développe-

ment Durable (ODD) [1]. Les ODD sont un enjeu majeur pour l'humanité. Ce sont 17 principes rédigés par les Nations Unies (NU) dans le but de parvenir à un avenir meilleur et plus durable. L'interface actuelle du SustainGraph est plutôt destinée à des personnes maîtrisant l'informatique. La difficulté principale est de traduire la question en langue naturelle sous forme de requête structurée à destination du graphe de connaissances dans un langage supporté par l'outil utilisé pour son développement.

Dans cet article, nous adressons ce problème et proposons un système de questions-réponses fondé sur les grands modèles de langage (*Large Language Models*, LLM) qui permet d'interroger directement un graphe de connaissances en langue naturelle via une interface conversationnelle. Les réponses sont également fournies en langue naturelle pour alimenter la conversation avec l'utilisateur·rice. Un agent conversationnel permettant de répondre aux questions sur les ODD est disponible à l'adresse https://bit.ly/wikit-egc-2024. Pour faciliter la reproductibilité des résultats, le code source du démonstrateur des expérimentations est également disponible [2].

2 État de l'art

Les graphes de connaissances s'interrogent communément via des requêtes formulées dans un langage de requête particulier. Les systèmes de questions-réponses sur un graphe de connaissances (*Question Answering over Knowledge Graph*, QA-KG) tentent de simplifier cela en utilisant une question en langue naturelle qu'ils convertissent en requête avant de l'exécuter. Dans cette section, nous présentons des travaux sur le QA-KG ainsi que les techniques de génération de texte qui peuvent être exploitées pour cette tâche.

2.1 Système de questions-réponses sur un graphe de connaissances

Les systèmes de QA-KG sont de plus en plus présents dans la littérature car ils permettent d'accéder plus simplement à l'information mais aussi de construire de nombreuses applications comme par exemple des agents conversationnels. Les étapes de ces systèmes sont souvent semblables, comme présenté par Liang et al. (2020) :

— extraction des entités nommées dans la question utilisateur·rice
— désambiguïsation et alignement des entités extraites avec les entités du graphe
— construction de la requête à partir de son type et des entités extraites
— classement des requêtes par pertinence par rapport à la question utilisateur·rice
— exécution de la requête sur le graphe de connaissances.

La construction de la requête est souvent basée sur une catégorisation de la question comme le font Liang et al. (2020); Kobeissi et al. (2021) et Srivastava et al. (2021). Les requêtes sont ensuite créées à partir de patrons plus ou moins complexes. Des représentations vectorielles peuvent également être utilisées comme dans les travaux de Srivastava et al. (2021) et Huang et al. (2019). Toutefois, ces méthodes nécessitent l'entraînement d'algorithmes supervisés et/ou l'annotation de données (e.g. des patrons) qu'il est difficile de transposer à d'autres domaines ou de faire évoluer dans le temps lorsque le périmètre fonctionnel change.

1. https://www.un.org/sustainabledevelopment/fr/objectifs-de-developpement-durable/, consulté le 7 décembre 2023
2. https://github.com/wikit-ai/kg-rag-egc2024/tree/main, consulté le 7 décembre 2023

Les étapes de désambiguïsation et d'alignement d'entités correspondent à la tâche d'*Entity Linking* (EL). Les mentions extraites de la question sont ainsi liées aux entités du graphe correspondantes comme expliqué par Al-Moslmi et al. (2020). Des techniques de similarité entre les mentions et les entités sont souvent utilisées. La méthode *ReDinED* de Ayoola et al. (2022) fait cela grâce à des représentations vectorielles. Les travaux de Pan et al. (2015) utilisent l'*Abstract Meaning Representation* (AMR) pour faire de l'EL et Diomedi et Hogan (2022) combinent différentes techniques d'EL pour maximiser les résultats. Ces méthodes se fondent sur des graphes comme DBpedia ou Wikidata pour extraire les entités. Aucune ressource n'a été trouvée pour réaliser cette tâche sur un graphe spécifique de façon non supervisée.

2.2 Génération de texte et génération augmentée de récupération

Les techniques de QA-KG précédemment citées se contentent souvent de retourner les résultats bruts issus du graphe sans formuler la réponse en langue naturelle. Cette dernière étape est cruciale dans notre cas pour la mise en place d'un agent conversationnel. Nous nous intéressons donc aux LLM, qui font beaucoup parler d'eux dans la littérature depuis plusieurs mois. Ces modèles sont entraînés à générer du texte en prédisant la suite probable d'une entrée textuelle. Plus récemment, ils ont été entraînés à suivre des instructions pour réaliser des tâches plus ou moins complexes [3]. Les instructions sont rédigées sous forme de *prompts*, qui sont les consignes associées aux données nécessaires au modèle pour produire une réponse.

Ces modèles utilisent leur grande connaissance interne acquise lors de l'entraînement pour réaliser diverses tâches. Cette connaissance peut parfois être limitante, notamment pour travailler sur des domaines très spécifiques ou avec des données récentes. La *Retrieval-Augmented Generation* (RAG) [4] permet de résoudre ce problème. Elle combine la recherche d'information et la génération de texte pour la tâche de QA. A partir de la question, des informations pertinentes pour y répondre sont recherchées dans les données disponibles : c'est la partie *retrieval*. Un prompt est construit avec les instructions de réponse, la question et les informations extraites comme contexte. Ce prompt est l'entrée d'un LLM : c'est la partie *augmented*. Le LLM génère une réponse à la question en utilisant le contexte fourni : c'est la partie *generation*. La RAG peut directement être appliquée pour des tâches de QA-KG comme le fait la méthode *KAP-ING* [5]. Les faits impliquant les entités du graphe présentes dans la question constituent le contexte. Le modèle de génération apporte ensuite une réponse en langue naturelle à partir de la question et du contexte. Cette technique améliore grandement les performances sur la tâche de QA comparé à des modèles de génération sans la partie RAG. Ils permettent également de fournir une réponse en langue naturelle là où les systèmes de QA-KG cités précédemment ne le permettent pas. Dans nos travaux, nous nous inspirons de cette architecture pour limiter les ressources nécessaires au fonctionnement du système. En effet, l'obtention d'un contexte précis permet l'utilisation d'un modèle de génération de texte de taille réduite.

3. https://arxiv.org/abs/2203.02155, consulté le 7 décembre 2023
4. https://arxiv.org/abs/2005.11401, consulté le 7 décembre 2023
5. https://arxiv.org/abs/2306.04136, consulté le 7 décembre 2023

3 Contributions

Dans cette section, nous détaillons nos contributions pour interroger un graphe de connaissances en langue naturelle. Nous nous limitons à des méthodes non supervisées avec le moins d'intervention humaine possible pour qu'elles puissent être généralisée à d'autres graphes.

3.1 Construction du graphe

Pour comparer les différentes techniques choisies et construire un démonstrateur illustrant nos contributions, nous avons utilisé le SustainGraph introduit par Fotopoulou et al. (2022). Le graphe tel qu'il est présenté à l'origine contient plus de 3 millions de nœuds et plus de 6 millions de relations. Nous travaillons sur une partie réduite du graphe qui s'intéresse à la description des ODD, de leurs cibles, des indicateurs à mesurer sans leurs observations, des grandes transformations induites par les ODD, des ministères impliqués, des interventions nécessaires ainsi que les résultats intermédiaires attendus, avec toutes les relations liant ces entités. Au total, nous travaillons avec 490 nœuds et 905 relations.

Le graphe a été construit à partir du code de Fotopoulou et al. (2022). L'implémentation est faite en utilisant le modèle de graphe attribué utilisé par Neo4j [6]. Les entités et les relations peuvent ainsi avoir différentes propriétés. Le langage de requête associé est le Cypher [7].

3.2 Génération de requêtes et de réponses avec un LLM

Une technique inspirée de la RAG pour interroger un graphe de connaissances en langue naturelle de façon non supervisée consiste à utiliser uniquement un LLM. Cette méthode réalise 2 appels au LLM pré-entraîné. Un premier appel au LLM permet de générer une requête à partir de la question et du schéma du graphe. La requête est exécutée et les résultats obtenus forment le contexte. Un second appel au LLM permet de générer une réponse en langue naturelle à partir de la question et du contexte. Nous utilisons *LangChain* [8], un framework ouvert très répandu dans la communauté autour des LLM, qui propose directement une implémentation de la méthode décrite pour Neo4j et le langage de requête Cypher [9]. À notre connaissance, il s'agit de la technique la plus efficiente pour interagir en langue naturelle sur un graphe de connaissances.

Dans la littérature, les modèles d'*OpenAI* sont la référence en termes de performances. Le modèle *gpt-3.5-turbo* a donc été utilisé avec 2 variantes. Dans la variante *2llm*, le prompt et le schéma du graphe sont ceux proposés par défaut dans *LangChain*. Dans la variante *2llm_modif*, le schéma du graphe et le prompt pour générer la requête ont été modifiés dans le but d'améliorer les performances. Pour le prompt de génération de la requête, un exemple sur les données du graphe a été ajouté avec une question et la requête correspondante. Pour le schéma, le format structuré de type JSON avec des tableaux d'objets proposé par Neo4j est remplacé par du texte descriptif.

6. https://neo4j.com/fr/, consulté le 7 décembre 2023
7. https://neo4j.com/developer/cypher/, consulté le 7 décembre 2023
8. https://python.langchain.com/docs/get_started/introduction, consulté le 7 décembre 2023
9. https://python.langchain.com/docs/use_cases/graph/graph_cypher_qa, consulté le 7 décembre 2023

3.3 Recherche d'information avec une similarité sémantique et génération de réponse avec un LLM

Pour challenger la technique précédente, nous avons souhaité limiter l'utilisation du LLM. La variante *simi_llm* effectue un unique appel par question au lieu de deux. Pour cela, nous nous sommes rapprochés des techniques de RAG originelles, principalement utilisées sur des documents. Le graphe est linéarisé pour être représenté sous forme textuelle.

Comme dans la méthode *KAP-ING*, nous décrivons les relations avec leur label et le label de leurs nœuds suivi d'une information permettant de les identifier (le titre s'il existe, sinon la description). Nous ajoutons les descriptions individuelles de chaque nœud avec le label suivi de ses propriétés (nom et contenu). Nous calculons la ressemblance entre la question utilisateur·rice et les informations linéarisées du graphe grâce à la similarité sémantique. Le texte est vectorisé avec le modèle *all-mpnet-base-v2* du framework *Sentence Transformers* [10] car il combine les meilleures performances en moyenne sur les tâches de vectorisation et de recherche sémantique. Nous calculons la proximité entre les vecteurs grâce à la similarité cosinus. Les faits avec un score de similarité au dessus d'un seuil fixé et dans la limite d'un nombre de faits maximal sont utilisés comme contexte. L'étape finale est la génération de la réponse en utilisant la question et le contexte extrait avec le modèle *gpt-3.5-turbo*.

3.4 Liaison d'entités et génération de réponse avec un LLM

Dans le but de préciser le contexte, nous nous sommes inspirés des méthodes d'EL pour expérimenter une autre technique qui extrait plus spécifiquement les informations du graphe permettant de répondre à la question. Nous listons les entités représentant les nœuds du graphe avec des synonymes. Cette liste est utilisée pour identifier les entités (nœuds du graphe) mentionnées dans une question. Une requête générique est ensuite construite avec un patron spécifique à Cypher. Si une seule entité est mentionnée dans la question, la requête sera "'*MATCH (e :Entity) RETURN e*". Dans le cas où plusieurs entités sont identifiées, toutes les combinaisons pouvant être formées par ses entités sont créées et transformées en requêtes, par exemple pour 2 nœuds :

— *MATCH (e1 : Entity1) -[r1]-> (e2 : Entity2) RETURN e1,r1,e2*
— *MATCH (e2 : Entity2) -[r1]-> (e1 : Entity) RETURN e2,r1,e1*

Les requêtes sont exécutées. Si le résultat est vide, une relation intermédiaire est ajoutée :

— *MATCH (e1 : Entity1) -[r1]-> (ne :NewEntity) -[r]-> (e2 : Entity2) RETURN e1,r1,e2*
— *MATCH (e2 : Entity2) -[r1]-> (ne :NewEntity) -[r]-> (e1 : Entity) RETURN e2,r1,e1*

Lorsque des informations complémentaires sont présentes dans la question (valeurs numériques ou noms communs), l'entité la plus proche est désignée comme l'entité dont elles dépendent et "*WHERE e.property CONTAINS value*" est ajouté à la requête avant le "*RETURN*".

Le résultat obtenu est utilisé comme contexte pour générer une réponse avec le LLM. Nous avons utilisé deux LLM différents pour la génération : *gpt-3.5-turbo* pour la version *el_gpt* et *mistralai/Mistral-7B-Instruct-v0.1* [11] pour la version *el_mistral* afin de comparer les performances avec un LLM plus petit (7 milliards de paramètres) et open-source. Notre démonstrateur [12] est construit avec la version *el_mistral*.

10. https://www.sbert.net/, consulté le 7 décembre 2023
11. https://arxiv.org/abs/2310.06825, consulté le 7 décembre 2023
12. https://bit.ly/wikit-egc-2024

4 Résultats

Pour évaluer les différentes versions de notre agent conversationnel fondé sur de la RAG, nous avons constitué un jeu de données de 50 questions sur les ODD. Un exemple de question est *"How many development goals are there ?"*.

La température est un hyperparamètre des LLM permettant d'ajuster le caractère aléatoire du texte généré. La température du modèle de génération de texte *gpt-3.5-turbo* est fixée à 0 et celle du modèle *Mistral-7B-Instruct-v0.1* est fixée à 0.1 (la valeur 0 déclenche une erreur) pour limiter le caractère aléatoire des réponses.

	2llm	*2llm_modif*	*simi_llm*	*el_gpt*	*el_mistral*
Contexte utile	36%	60%	78%	66%	66%
Contexte exact	30%	52%	0%	42%	42%
Réponse utile	52%	68%	84%	74%	74%
Réponse exacte	34%	52%	40%	54%	46%

TAB. 1 – *Pourcentage des questions satisfaisant utilité et exactitude pour le contexte et la réponse.*

Les résultats sont présentés dans le tableau 1. Chacun des modèles a été évalué sur l'ensemble du jeu de données. Le contexte permet d'évaluer la partie recherche d'information et la réponse permet d'évaluer la partie génération. Les contextes et les réponses obtenus sont annotés manuellement sur la partie recherche d'information et sur la partie génération pour obtenir le pourcentage de questions qui satisfont les critères listés ci-après. Un contexte est utile s'il apporte des informations permettant de répondre à la question. Un contexte est exact si ses informations permettent de répondre précisément et exactement à la question posée, sans informations manquantes ou inutiles. Une réponse est utile si elle apporte des éléments de réponse sans qu'elle soit forcément complète ou dans un format parfait. Une réponse est exacte si elle contient tous les éléments de réponse nécessaires dans un format parfaitement acceptable pour la tâche. Nous n'utilisons pas de métriques standards pour la partie recherche d'information car la génération de requête pour les modèles *2llm* et *2llm_modif* peut conduire à un contexte contenant des informations non présentes explicitement dans le graphe (par exemple lorsque la requête *COUNT* est utilisée, le contexte est le nombre total de faits d'intérêts et non la liste de faits en question). Cependant, le contexte exact se rapproche d'une précision et d'un rappel égaux à 1, c'est-à-dire que l'ensemble des informations retrouvées sont pertinentes et que toutes les informations pertinentes ont été retrouvées.

On remarque que la modification du prompt améliore significativement le contexte avec les méthodes *2llm* et *2llm_modif* car il améliore les requêtes générées. Cela impacte directement les réponses avec une nette augmentation du pourcentage de réponses utiles et exactes. Bien que le modèle *simi_llm* soit celui avec le plus haut pourcentage de contexte utile (78% des cas), c'est aussi le seul modèle qui n'a aucun contexte exact. Cela conduit à un grand écart entre le pourcentage de réponses utiles et celui de réponses exactes, avec le deuxième moins bon score en terme de réponses exactes. Cette technique pourrait être améliorée en modifiant le seuil de similarité ou le nombre de faits maximum à utiliser mais la définition de règles statiques est contraignante et demande de les retravailler lors d'un nouveau cas d'usage. Les modèles *el_gpt*

et *el_mistral* ont des performances comparables à celles du modèle *2llm_modif*. Il y a respectivement plus de réponses avec des contextes utiles et moins de réponses avec des contextes exacts. Le modèle *el_gpt* comptabilise le plus grand nombre de réponses exactes alors que le modèle *el_mistral* se classe en troisième position. Il est donc intéressant de favoriser des techniques utilisant peu de ressources comme l'association d'EL et de patrons de requêtes puisqu'elles conduisent à de meilleurs résultats et nécessitent des efforts humains comparables (adaptation d'un prompt ou du patron de requête). Grâce au contexte précis obtenu, nous pouvons envisager d'utiliser des modèles de génération de texte plus petits comme *Mistral* afin de réduire les ressources nécessaires. De plus, le modèle est open-source et regroupe une large communauté pour améliorer les techniques de prompts ou les spécialisations du modèle. Les résultats pourront donc être encore améliorés.

Globalement, le pourcentage de réponses utiles dépasse le pourcentage de contextes utiles et le pourcentage de réponses exactes est supérieur ou égal au pourcentage de contextes exacts pour tous les modèles. Ce phénomène illustre l'utilisation de la connaissance interne des modèles de génération de texte pour produire une réponse, bien qu'il leur soit explicitement demandé de ne pas le faire dans le prompt. Cela permet d'apporter des réponses dans des cas où le contexte ne les contient pas mais cela augmente toutefois le risque d'hallucinations. Cette problématique sera très intéressante à traiter dans de prochains travaux.

5 Conclusion et perspectives

Dans ces travaux, nous proposons de construire un agent conversationnel pour des applications de QA-KG. Nous avons évalué différentes techniques inspirées de la RAG et choisi d'utiliser de l'EL avec des patrons de requêtes et le LLM *Mistral Instruct*. Cette méthode est compétitive avec les alternatives de l'état de l'art en utilisant un modèle de génération de texte européen open-source de taille limitée et en utilisant la génération de texte une seule fois au lieu de deux. Notre approche s'adapte également à tout graphe de connaissances. Seule la modification des patrons de requête est nécessaire suivant le formalisme (par exemple pour un autre langage que le Cypher) et les spécificités du graphe.

Un plus grand nombre de LLM peuvent être testés pour améliorer les performances mais nous envisageons plutôt de concentrer nos efforts sur la partie recherche d'information. En effet, nous avons montré que l'utilité et l'exactitude de la réponse sont directement liées à celles du contexte. Nous souhaitons donc développer un module d'EL pour n'importe quel graphe afin d'améliorer significativement le contexte. L'objectif sera de lier les mentions aux entités avec des techniques fondées sur la sémantique sans besoin de lister les entités, pour des graphes même autres que DBpedia ou Wikidata. L'amélioration du module de création de requêtes pourra également être envisagée pour permettre la création de requêtes plus complexes ou pour essayer différentes alternatives à la requête testée si celle-ci ne retourne aucun résultat.

Références

Al-Moslmi, T., M. G. Ocaña, A. L. Opdahl, et C. Veres (2020). Named entity extraction for knowledge graphs : A literature overview. *IEEE Access 8*, 32862–32881.

Ayoola, T., S. Tyagi, J. Fisher, C. Christodoulopoulos, et A. Pierleoni (2022). ReFinED : An efficient zero-shot-capable approach to end-to-end entity linking. In *NA Chapter of the ACL : Human Language Technologies : Industry Track*, pp. 209–220. Association for Computational Linguistics.

Diomedi, D. et A. Hogan (2022). Entity linking and filling for question answering over knowledge graphs. In *NLIWoD@ESWC*.

Fotopoulou, E., I. Mandilara, A. Zafeiropoulos, C. S. Laspidou, G. Adamos, P. Koundouri, et S. Papavassiliou (2022). Sustaingraph : A knowledge graph for tracking the progress and the interlinking among the sustainable development goals' targets. In *Frontiers in Environmental Science*.

Hogan, A., E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A.-C. N. Ngomo, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, et A. Zimmermann (2020). Knowledge graphs. *ACM Computing Surveys (CSUR) 54*, 1 – 37.

Huang, X., J. Zhang, D. Li, et P. Li (2019). Knowledge graph embedding based question answering. *Twelfth ACM Int Conf on Web Search and Data Mining*.

Kobeissi, M., N. Assy, W. Gaaloul, B. Defude, et B. Haidar (2021). An intent-based natural language interface for querying process execution data. *2021 3rd Int Conf on Process Mining (ICPM)*, 152–159.

Lehmann, J., R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, et C. Bizer (2015). DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *6*(2), 167–195.

Liang, S.-Y., K. Stockinger, T. M. de Farias, M. O. Anisimova, et M. Gil (2020). Querying knowledge graphs in natural language. *Journal of Big Data 8*.

Pan, X., T. Cassidy, U. Hermjakob, H. Ji, et K. Knight (2015). Unsupervised entity linking with abstract meaning representation. In *NA Chapter of the ACL*.

Srivastava, S., M. Patidar, S. A. Chowdhury, P. Agarwal, I. Bhattacharya, et G. M. Shroff (2021). Complex question answering on knowledge graphs using machine translation and multi-task learning. In *European Chapter of the ACL*.

Vrandečić, D. et M. Krötzsch (2014). Wikidata : A Free Collaborative Knowledgebase. *57*(10), 78–85.

Summary

Knowledge graphs are widely used to represent a domain. In particular, they enable data from different sources to be brought together and stored in a structured way. However, accessing these resources can be complex, as we need to master a query language that is compatible with the formalism used to represent the knowledge graph. In this article, we propose a question-answering system using Large Language Models to query knowledge graphs in natural language. We compare different techniques in order to best combine frugality and performance with the aim of controlling the resources used. Our work is based on a knowledge graph describing the Sustainable Development Goals defined by the United Nations.