Optimisation du RAG sur hypergraphes : vers une meilleure extraction de faits et récupération de chunks

Houda Khrouf, Pedro Fillastre, Sebastiao Correia

Applied Research, Qlik

{houda.khrouf, pedro.fillastre, sebastiao.correia}@qlik.com

Résumé

GraphRAG améliore le raisonnement en structurant les connaissances sous forme de graphes mais reste limité pour représenter les faits n-aires. HyperGraphRAG exploite des hypergraphes pour préserver la sémantique des faits et améliorer la précision des réponses RAG, mais dépend d'une extraction par LLM sujette à des erreurs et d'une récupération de chunks peu efficace. Nous adressons ces limites en intégrant le self-consistency prompting afin d'améliorer l'extraction, ainsi qu'un algorithme de Personalized PageRank sur hypergraphe pour optimiser la récupération des chunks.

Abstract

GraphRAG enables deeper reasoning by structuring knowledge as graphs but struggles with n-ary facts. HyperGraphRAG uses hypergraphs for richer semantics, improving accuracy, yet relies on error-prone LLM extraction and inefficient standard chunk retrieval. We address this by employing self-consistency prompting to improve the extraction, and Personalized PageRank algorithm over hypergraph to enhance chunk retrieval.

Keywords

GraphRAG, HyperGraph, n-ary relations, Personalized PageRank

1 Introduction

L'intégration de connaissances externes dans les grands modèles de langage (LLM) via la génération augmentée par récupération (RAG - Retrieval-Augmented Generation) s'est imposée comme une stratégie clé pour améliorer l'exactitude factuelle et réduire les hallucinations [12]. Les systèmes RAG standards, fondés sur une recherche par similarité vectorielle entre segments de texte (chunks), privilégient la similarité sémantique superficielle au détriment des relations complexes entre entités. S'ils s'avèrent efficaces pour des requêtes factuelles ciblées, ils manquent de profondeur contextuelle pour les problématiques nécessitant une compréhension transversale.

Pour pallier ces insuffisances, le GraphRAG [22, 7, 2] a émergé comme une évolution stratégique, structurant les connaissances sous forme de graphes de connaissances (Knowledge Graphs). En modélisant les dépendances entre les nœuds, le GraphRAG permet une compréhension contextuelle plus fine et une navigation efficace à travers des chemins relationnels. Cette structure facilite non seulement le raisonnement multi-sauts (multi-hop) et la désambiguïsation des entités, mais elle permet aussi de répondre à des requêtes globales complexes (ex : "Quels sont les thèmes principaux ?") que les systèmes RAG traditionnels peinent à traiter. Cependant, malgré leur capacité à capturer des relations binaires, les graphes de connaissances classiques atteignent leurs limites face à la complexité intrinsèque des données réelles. De nombreuses re-

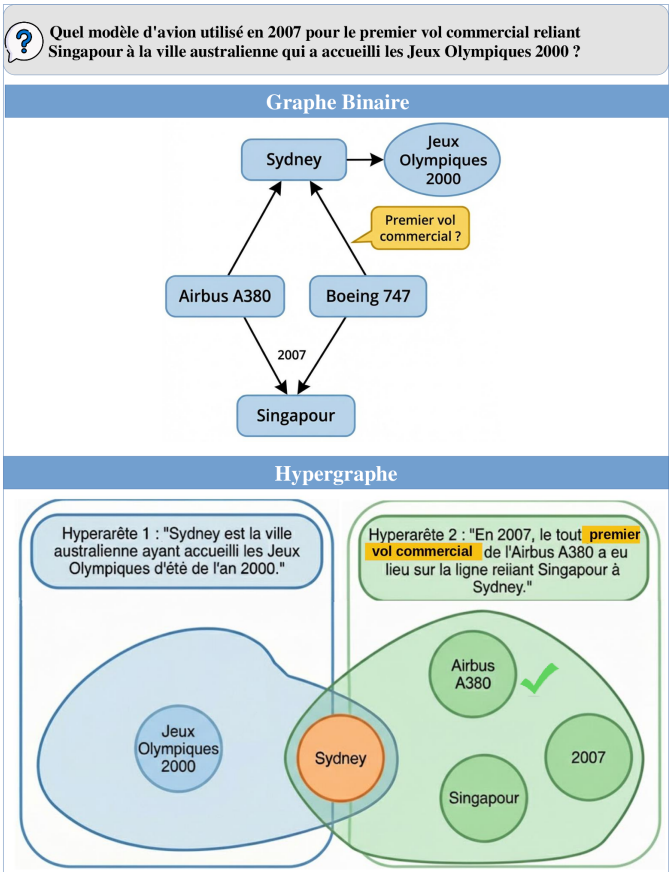

FIGURE 1 – Comparaison entre un Graphe de Connaissances binaire (KG) et un Hypergraphe de Connaissances (KHG) pour répondre à une question multi-sauts

lations sont par nature n-aires, impliquant simultanément plusieurs entités (par exemple, une collaboration entre trois entreprises ou un événement liant plusieurs acteurs à un lieu précis). La décomposition de ces relations en simples arêtes binaires entraîne inévitablement une perte d'information sémantique et structurelle. C'est dans ce contexte que des approches [14, 21, 6], basées sur les hypergraphes de connaissances, ont été proposées. Contrairement aux graphes binaires, les hypergraphes utilisent des hyper-arêtes pour représenter des faits connectant plusieurs nœuds simultanément, offrant ainsi une expressivité supérieure et une conservation fidèle de l'intégrité des données. Comme illustré dans la Figure 1, la relation décrivant "En 2007, le premier vol commercial de l'Airbus A380 a relié Singapour à Sydney" connecte quatre entités. Dans un graphe binaire, cette relation est fragmentée en liens indépendants, rendant impossible la distinction avec d'autres vols effectués par d'autres appareils (comme un Boeing 747) partageant certains attributs (départ, destination, date).

Cependant, malgré leur expressivité supérieure, les approches basées sur les hypergraphes présentent deux limitations majeures :

- Complexité de construction de l'hypergraphe : Bien que la représentation par hyperarêtes préserve mieux l'intégrité factuelle que les graphes binaires — souvent sensibles à la complexité linguistique du texte (formes temporelles, négations, etc.) [19] — son extraction par LLM demeure sujette à des instabilités structurelles : omission d'entités ou d'hyperarêtes générant des nœuds isolés, et résolution de coréférences défaillante entravant la continuité des données. Ces lacunes fragmentent la topologie du graphe et limitent la capacité du modèle à naviguer entre les connaissances, impactant directement le raisonnement multi-sauts.
- Sous-exploitation de la connectivité structurelle : La stratégie de récupération employée dans HyperGraphRAG [14] repose sur une recherche sémantique combinée à une expansion locale du voisinage. Ce mécanisme sous-exploite la topologie globale de l'hypergraphe et souffre d'un problème d'horizon : dépendant du périmètre sémantique de la requête, il ne peut capturer les fragments structurellement connectés au-delà du voisinage immédiat. Cette approche occulte les fragments présentant une dépendance structurelle significative. Elle manque l'opportunité de prioriser les passages par leur degré de connectivité avec les entités pertinentes, limitant le raisonnement multi-sauts.

Pour pallier ces limites, nous proposons deux contributions. Premièrement, une méthode d'extraction optimisée (EXT^{++}) fondée sur le *self-consistency prompting*, qui améliore la complétude et la connectivité de l'hypergraphe sans surcoût d'extraction. Ensuite, un mécanisme de récupération basé sur le Personalized PageRank (PPR) opérant sur l'hypergraphe, inspiré d'HippoRAG2 [8], qui identifie les chunks les plus pertinents via leur connectivité structurelle plutôt que par simple similarité vectorielle. Le code source et les données d'évaluation sont disponibles publiquement [1] pour assurer la reproductibilité des résultats.

1. https://github.com/qlik-oss/HyperGraphRAG/

2 État de l'art

L'architecture RAG repose sur un pipeline modulaire articulé autour de trois phases successives : la pré-récupération (pre-retrieval), la récupération et la génération. La phase initiale de pré-récupération définit la représentation et la structure des données sources à travers la segmentation du texte et le choix de la granularité (de la phrase au document), intégrant parfois une modélisation en Graphes de Connaissances pour optimiser le contexte sémantique. La segmentation du texte est essentielle pour surmonter les contraintes de tokens des modèles de langage, tout en améliorant la précision et l'efficacité du système RAG. Les stratégies initiales, telles que le découpage à taille fixe ou récursif, sont simples à implémenter, mais peuvent nuire à la cohérence sémantique. Cela rend le système inefficace pour répondre à des questions globales et accentue les hallucinations sur les relations complexes. Pour y remédier, la segmentation contextuelle [1] enrichit chaque chunk par un résumé de son contexte global dans le document, limitant ainsi la perte d'information lors de l'indexation et améliorant la recherche des chunks pertinents. D'autres approches basées sur des graphes de connaissances dépassent les limites de segmentation en interconnectant les entités à travers des relations sémantiques, passant d'une recherche de similarité textuelle plate à une compréhension structurelle et relationnelle profonde des données. Au cœur de cette évolution, GraphRAG proposé par Microsoft [4] structure le corpus sous forme de graphe de connaissances et exploite la détection de communautés pour produire des résumés hiérarchiques, facilitant le traitement de requêtes globales. D'autres travaux récents ont cherché à améliorer l'expressivité de la représentation des connaissances et la qualité du raisonnement sur graphe. LightRAG [7] repose sur une indexation incrémentale et une récupération à deux niveaux : une recherche fine centrée sur les entités et relations pertinentes, combinée à une recherche thématique de plus haut niveau. HippoRAG2 [8] propose une architecture inspirée par la mémoire associative humaine et l'hippocampe, utilisant Personalized PageRank pour identifier les chunks pertinents en exploitant la topologie du graphe. PathRAG [2] affine la récupération à travers un élagage des chemins de raisonnement (path pruning), afin de sélectionner les sous-graphes les plus informatifs et de limiter l'introduction de bruit dans le contexte transmis au modèle.

L'évolution de ces systèmes a conduit à une diversification des types de graphes modélisant les données. On distingue notamment : (i) le graphe de connaissances classique (KG) qui extrait des relations binaires entre les entités [8, 22], parfois enrichi par une couche taxonomique ou ontologique afin d'améliorer la structuration sémantique et la désambiguïsation [13], (ii) le graphe textuel (ou graphe à attributs) où chaque nœud correspond à un segment textuel doté de métadonnées permettant une récupération multi-sauts via des mécanismes neuronaux (GNN) ou structurels [9, 10] ; et (iii) le graphe bipartite (passage–entité) qui modélise explicitement les co-occurrences pour offrir une indexation compacte visant à guider la recherche contextuelle [11].

Ces représentations restant limitées aux relations binaires, la recherche récente se tourne vers les hypergraphes pour modéliser les interactions d'ordre supérieur évoquées en introduction. À cet égard, HyperGraphRAG [14] pose les jalons de ce paradigme en exploitant les hyper-arêtes pour représenter les faits complexes. Cette approche surpasse les graphes classiques par une capture plus fidèle de la cooccurrence d'entités, offrant ainsi une expressivité structurelle supérieure. Parallèlement, Hyper-RAG [6] propose une architecture hybride. En extrayant conjointement des relations binaires et N-aires couplées à une indexation vectorielle, il fournit au LLM un contexte factuel d'une densité inédite lors des phases de récupération et de génération, réduisant drastiquement le risque d'hallucinations. Plus récemment, PRoH [21] marque une étape supplémentaire en axant son approche sur la planification adaptative du raisonnement. Le système décompose les requêtes en un graphe orienté acyclique (DAG) de sous-questions et navigue itérativement dans le voisinage local de l'hypergraphe. Bien que la qualité de génération s'en trouve nettement améliorée, l'impact de ce processus itératif sur la latence du système demeure un angle mort de son évaluation. Ces travaux laissent ouvertes les questions de fiabilité de l'extraction à grande échelle et de l'exploitation de la topologie globale de l'hypergraphe lors de la récupération.

3 Méthodologie

La méthodologie proposée s'articule autour de deux axes complémentaires visant à fiabiliser la construction de l'hypergraphe et à optimiser la récupération des chunks pertinents.

3.1 Optimisation de l'extraction (EXT^{++})

L'extraction de graphes de connaissances par LLM s'oriente de plus en plus vers le paradigme de l'Extraction d'Information Ouverte (OpenIE). S'affranchissant des ontologies *a priori*, les LLMs sont exploités pour découvrir et extraire dynamiquement les entités et relations latentes directement depuis les corpus documentaires (bottom-up discovery). Cependant, les architectures de LLM étant fondamentalement optimisées pour la génération séquentielle de tokens plutôt que pour l'extraction d'information structurée, cette nature générative soulève d'importants défis. Outre la prolifération sémantique, le processus d'extraction se heurte à une rupture de niveau de granularité entre la fluidité du langage naturel et la rigidité du format triplet. Il souffre principalement de défaillances structurelles, telles que l'incomplétude face aux verbes d'action complexes ou la fragmentation du graphe due à une résolution de coréférence imparfaite. S'y ajoutent des biais cognitifs du modèle, notamment la dérive sémantique : en simplifiant abusivement les relations, le système transforme des nuances (incertitudes, hypothèses, temporalité) en faits avérés. Cette incapacité à modéliser la négation ou le doute génère un « bruit » factuel, dégradant la fidélité du graphe [19]. L'extraction sous forme d'hypergraphe atténue intrinsèquement ces défaillances cognitives en préservant l'intégrité sémantique des faits. Cependant, elle demeure vulnérable à des instabilités structurelles : omission d'entités, production d'hyperarêtes isolées, parfois dénuées de réelle signification sémantique (ex. fragments résiduels de type « Que suis-je »).

Bien que le fine-tuning sur des tâches d'extraction de graphe constitue une piste prometteuse [19], son application reste conditionnée par la disponibilité de corpus labellisés qui font à ce jour défaut pour les hypergraphes. Nous avons donc orienté notre choix vers une optimisation par ingénierie de prompt avancée. Nous proposons une extension de l'extraction few-shot d'HyperGraphRAG [14] en y intégrant un mécanisme d'auto-cohérence : le *self-consistency prompting*. Ce mécanisme s'inspire du paradigme Universal Self-Consistency (USC) [3], une méthode permettant aux LLMs d'évaluer et d'agréger de multiples chemins de génération pour faire émerger un consensus robuste, sans évaluateur externe. Cette intégration présente trois avantages fondamentaux. Premièrement, elle réduit les hallucinations et la dérive sémantique [18] : en s'appuyant sur la convergence de multiples extractions plutôt que sur un unique décodage glouton (greedy decoding), les relations erronées ou spéculatives sont statistiquement filtrées. Deuxièmement, elle atténue la variance inhérente aux LLMs et leur sensibilité au biais de position dans les longs contextes, un problème fréquent conduisant à l'omission d'entités périphériques. Enfin, l'agrégation par union des différentes itérations maximise la complétude de l'hypergraphe.

Concrètement, notre méthode EXT^{++} repose sur la définition d'un prompt présenté dans l'Annexe A. Ce prompt exécute trois itérations d'extraction au sein d'un seul appel LLM pour un même chunk, fusionnées en interne par union pour garantir une topologie finale plus dense et connectée. Outre cette dynamique d'agrégation, EXT^{++} repense la structure des instructions dans le prompt : le modèle est explicitement contraint de lister les entités concernées immédiatement après la définition de chaque hyperarête, en respectant leur ordre d'apparition dans le texte source. De plus, une consigne de résolution des coréférences est appliquée dès l'extraction (ex. substitution d'un pronom par l'entité nommée complète). Cette ingénierie d'extraction ciblée permet (1) d'améliorer la traçabilité de l'information extraite, (2) de prévenir l'oubli d'entités, (3) de faciliter l'association topologique univoque entre une hyperarête et ses entités, et (4) de minimiser les erreurs liées aux ambiguïtés pronominales.

3.2 Récupération optimisée via PageRank personnalisé (PPR) sur hypergraphe

Dans l'approche HyperGraphRAG [14], la récupération repose sur une recherche sémantique d'entités et d'hyperarêtes suivie d'une expansion topologique bidirectionnelle à un saut, complétée par des chunks issus d'une recherche vectorielle classique. Comme évoqué dans l'introduction, cette stratégie sous-exploite la topologie globale du graphe et la recherche vectorielle dense tend à introduire un bruit contextuel — passages redondants ou structurellement déconnectés du besoin réel — ce qui dilue les signaux utiles

et accroît la vulnérabilité du modèle aux hallucinations.

Pour pallier ces limites, nous intégrons directement les chunks en tant que nœuds dans l'hypergraphe, les reliant explicitement aux entités et hyperarêtes qui en sont issues. Sur cette structure unifiée, nous appliquons l'algorithme Personalized PageRank (PPR), inspiré d'HippoRAG2 [8], en traitant les entités, hyperarêtes et chunks comme des nœuds d'un graphe triparti. L'objectif est d'identifier les chunks à la fois sémantiquement pertinents et fortement connectés, sur le plan topologique, au contenu de la requête. Le PPR agit comme un filtre robuste face aux faux positifs de la recherche dense, permettant d'affiner la qualité du sous-graphe extrait avant la génération de réponse.

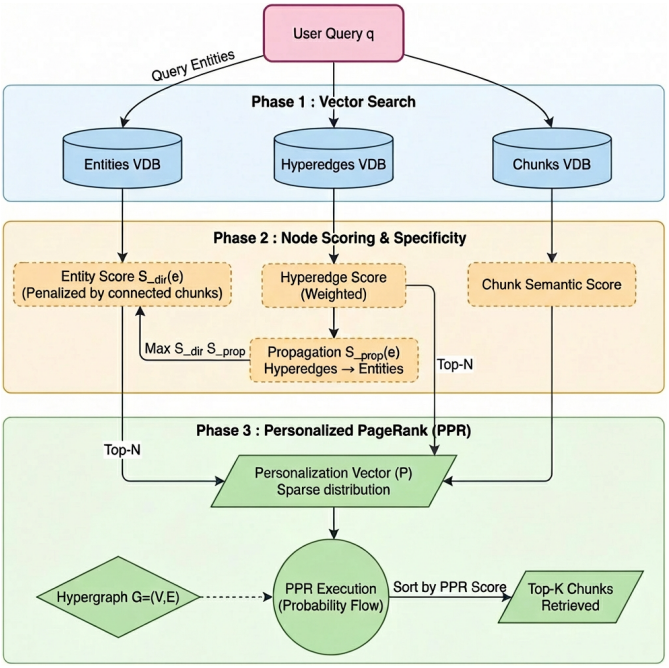

FIGURE 2 – Pipeline de récupération basé sur Hypergraphes et PageRank Personnalisé (PPR)

Comme illustré dans la Figure 2, la récupération optimisée est composée de trois étapes :

Étape 1 : Recherche vectorielle. La requête utilisateur q est utilisée pour effectuer une recherche vectorielle dans les bases des hyperarêtes et des chunks, tandis que les entités extraites de q interrogent la base des entités. Les nœuds récupérés sont triés par similarité décroissante, et seuls les k premiers de chaque type sont conservés. Toutefois, les hyperarêtes font l'objet d'un tri spécifique fondé sur un critère hybride combinant leur similarité et leur poids d'importance dans le document source, attribué lors de l'extraction.

Étape 2 : Notation des noeuds. Chaque élément récupéré (entité, hyperarête, chunk) reçoit un score de pertinence initial correspondant à sa similarité avec la requête. Des stratégies de pondération spécifiques sont appliquées pour calibrer ces scores. D'une part, en s'inspirant de l'efficacité reconnue des signaux globaux d'importance tels que l'IDF (Inverse Document Frequency) pour la recherche d'information, une pénalité de spécificité est appliquée aux entités : le score est divisé par le degré de connectivité aux chunks ($|C_e|$) afin de pénaliser les entités génériques :

$$s_{\text{local}}(e) = \frac{\text{sim}(e, q)}{|C_e|} \tag{1}$$

Enfin, un mécanisme clé de cette étape est la propagation (S_{prop}) de pertinence des hyper-arêtes vers les entités associées. Cela permet d'enrichir le vecteur de personnalisation du PPR avec des entités qui n'auraient pas été identifiées par la seule recherche vectorielle sur les noms d'entités, élargissant la couverture du signal de la requête dans le graphe. Le score propagé d'une entité e est calculé comme la moyenne des scores des hyperarêtes incidentes, pondérée par la connectivité :

$$\bar{s}(e) = \frac{1}{|\mathcal{H}_e|} \sum_{h \in \mathcal{H}_e} \frac{\text{sim}(h, q)}{|C_e|} \tag{2}$$

Ensuite, une agrégation Noisy-OR amplifie le score des entités connectées à plusieurs hyperarêtes tout en le maintenant borné dans $[0, 1]$:

$$s_{\text{prop}}(e) = 1 - (1 - \bar{s}(e))^{1+\ln|\mathcal{H}_e|} \tag{3}$$

où $\mathcal{H}_e$ désigne l'ensemble des hyperarêtes retrouvées incidentes à e, $\text{sim}(h, q)$ la similarité vectorielle entre l'hyperarête h et la requête q, et $|C_e|$ le nombre de chunks associés à e. L'intuition du Noisy-OR est que chaque hyperarête incidente constitue une observation indépendante : plus une entité est liée à des hyperarêtes pertinentes, plus son score augmente. Le score final de chaque entité est ensuite déterminé par :

$$s(e) = \max(s_{\text{local}}(e),\ s_{\text{prop}}(e)) \tag{4}$$

D'autre part, le score des chunks correspond à leur similarité modulée par un facteur de pondération w_{chunk}, équilibrant leur influence par rapport aux nœuds conceptuels (entités et hyperarêtes) dans l'algorithme PPR.

Étape 3 : PageRank Personnalisé. Les scores calculés sont normalisés par Min-Max puis assemblés en un vecteur de personnalisation éparse fusionnant les Top-N entités et hyperarêtes, ainsi que les scores de similarité pondérés des chunks. Ce vecteur encode le biais de la requête sur l'ensemble des nœuds de l'hypergraphe. L'algorithme PPR propage ce signal à travers la structure, redistribuant l'importance de chaque nœud en fonction de sa proximité structurelle avec les éléments pertinents. Les nœuds de type chunk sont enfin extraits et triés par score PPR décroissant, fournissant les Top-K passages les plus pertinents pour la génération.

4 Expérimentations

4.1 Cadre d'évaluation :

Données. Afin d'évaluer l'efficacité de notre stratégie d'optimisation, nous avons sélectionné trois jeux de données

Method	FICTION			CS			MAUD		
	C-Recall	A-Correct	A-Complete	C-Recall	A-Correct	A-Complete	C-Recall	A-Correct	A-Complete
Standard RAG	0.40	7.11	5.01	0.60	8.56	6.34	0.10	3.38	2.22
Contextual Chunking	0.45	8.00	5.62	0.55	8.39	6.21	**0.36**	7.62	4.76
Graph Transformer (Neo4j)	0.40	6.96	4.95	0.60	8.65	6.45	0.12	7.73	5.04
HippoRAG2	0.44	8.51	6.19	0.52	8.51	6.77	0.35	**8.69**	4.12
HyperGraphRAG	0.39	9.54	7.17	0.67	9.37	8.27	0.13	8.28	5.24
HyperGraphRAG + PPR	0.55	9.64	7.82	**0.73**	9.43	**8.28**	0.19	8.48	5.39
HyperGraphRAG + PPR + EXT++	**0.59**	**9.75**	**7.94**	0.71	**9.45**	**8.28**	0.22	8.68	**5.80**

TABLE 1 – Comparaison des performances à travers différents domaines. Les meilleurs scores sont en gras et les seconds sont soulignés pour chaque jeu de données.

issus de différents benchmarks présentant des défis distincts. Une attention particulière a été portée à l'exclusion des corpus basés sur Wikipedia, omniprésents dans les jeux de données d'entraînement des LLM, afin de prévenir tout biais de mémorisation.

— **Fiction** : Issu du benchmark UltraDomain [17], il évalue les systèmes RAG sur l'analyse de récits longs et complexes. En raison de la longueur extrême des documents — atteignant jusqu'à 560k tokens et nécessitant un nombre élevé d'appels au LLM par document pour l'extraction de l'hypergraphe — nous avons constitué une version réduite par sélection aléatoire de 13 documents parmi 30, afin de maîtriser le coût et le temps d'expérimentation. Ce sous-ensemble comprend 84 questions, dont 72 requêtes multi-sauts nécessitant la synthèse de 2 à 10 chunks. Ces questions couvrent cinq dimensions analytiques : la thématique, le développement des personnages, l'intrigue, les techniques littéraires et le contexte méta-fictionnel.

— **CS** (Computer Science) : Issu du papier HyperGraphRAG [14] et initialement dérivé du benchmark UltraDomain [17], il repose sur des manuels universitaires volumineux spécialisés en architecture informatique, algorithmes mathématiques et apprentissage automatique. Il contient 3 documents techniques denses faisant l'objet de 398 questions, dont 175 requêtes multi-sauts. Ces questions sont construites à partir de fragments de connaissances situés à une distance de 1 à 3 sauts, impliquant des tâches de récupération de faits et de synthèse multi-entités.

— **MAUD** (Merger & Acquisition Understanding Dataset) : Issu du benchmark LegalBench-RAG [16], il est dédié à l'analyse de clauses contractuelles complexes au sein d'accords de fusions-acquisitions. Identifié comme le volet le plus exigeant du benchmark en raison de sa terminologie juridique et financière hautement spécialisée, il se compose de questions factuelles à saut unique (single-hop), signifiant que la réponse peut être extraite directement d'un seul passage. Pour nos travaux, nous avons construit une version réduite et représentative en sélectionnant aléatoirement 21 documents parmi les 150 disponibles. Ce sous-ensemble comprend 74 questions et permet de limiter le coût et le temps d'expérimentation.

Baselines. Afin d'évaluer la performance de notre approche, nous l'avons comparée à plusieurs méthodes de l'état de l'art, réparties en quatre catégories distinctes :
(i) **RAG standard** : l'approche conventionnelle basée sur la segmentation en chunks et exploitant une recherche de similarité vectorielle dense ou hybride (combinant recherche sémantique et lexicale) ; (ii) **RAG avec segmentation enrichie** explorant des stratégies de segmentation avancées. Bien que des méthodes aient été évaluées sur LegalBench-RAG [16] (notamment le semantic chunking, le proposition chunking et le summary chunking), c'est le chunking contextuel d'Anthropic [1] qui s'est avéré être le plus performant dans nos tests préliminaires et retenu comme baseline ; (iii) **RAG basé sur les graphes** où deux baselines sont retenues : HippoRAG2 [8] décrite dans la section 2 et GraphTransformer [15] qui s'appuie sur une représentation binaire du graphe d'entités et bénéficie d'une intégration native avec les bibliothèques LangChain et Neo4j. Lors de la récupération, cette approche identifie les entités de la requête et traverse le graphe pour en extraire les voisinages pertinents ; (iv) **RAG basé sur les hypergraphes** : Pour modéliser les relations d'ordre supérieur, nous avons intégré HyperGraphRAG [14] que nous étendons dans ce travail. Évaluée sans notre optimisation, elle sert de baseline pour mesurer l'apport de chaque contribution.

Métriques. Notre évaluation repose sur trois métriques principales : le rappel contextuel (Contextual Recall), l'exactitude (Correctness) et la complétude de la réponse (Completeness). Le rappel contextuel (Contextual Recall), implémenté dans la bibliothèque RAGAS [5], mesure la part des informations pertinentes extraites des documents de référence qui sont effectivement réutilisées dans la réponse générée. Autrement dit, il évalue la capacité du système à récupérer et mobiliser le contexte nécessaire pour produire une réponse pertinente.
En outre, pour évaluer l'exactitude et la complétude, nous adoptons une approche LLM-as-a-judge exploitant le modèle GPT-4.1-mini. Cette approche compare la réponse générée à une réponse de référence à l'aide d'un prompt inspiré de celui défini dans [14]. L'exactitude évalue dans quelle mesure la réponse est logiquement et factuellement alignée avec la référence, tandis que la complétude évalue si elle couvre l'ensemble des aspects essentiels. Ces deux

dimensions sont notées sur une échelle de 0 à 10, selon des critères bien définis dans le prompt.
Nous avons choisi d'écarter le score F1 de notre évaluation, bien qu'il soit largement utilisé dans la littérature. En effet, cette métrique repose sur une similarité lexicale entre la réponse générée et la référence, ce qui la rend particulièrement sensible aux variations de formulation, de longueur et de tokenisation. Dans le contexte du RAG, où une même question peut admettre plusieurs réponses correctes formulées différemment, cette dépendance aux correspondances lexicales peut introduire un biais d'évaluation.

Implémentation. Le modèle GPT-4o-mini (distinct du modèle utilisé pour l'évaluation) est utilisé pour l'extraction de l'hypergraphe et la génération des réponses, tandis que `NovaSearch/stella_en_400M_v5` est utilisé pour la représentation vectorielle. Le paramètre top-k est fixé à 5 chunks pour les jeux de données CS et MAUD, et à 10 pour Fiction où certaines requêtes requièrent plus de 5 chunks. Pour la phase de récupération, nous conservons les paramètres d'HyperGraphRAG d'origine : $k_V = 60$ entités candidates et $k_H = 60$ hyperarêtes candidates sont extraites par recherche vectorielle, puis tronquées selon un budget de 4,000 tokens chacune. Pour le Personalized PageRank (PPR), nous fixons son facteur d'amortissement $\alpha = 0.5$ (valeur retenue par HippoRAG2 à l'issue de leur réglage d'hyperparamètres sur un sous-ensemble des données d'entraînement), le facteur de pondération des chunks $w_{\text{chunk}} = 0.5$ (déterminé par validation empirique sur les 3 jeux de données en maximisant le rappel), et nous initialisons le vecteur de personnalisation sur les top $k_{\text{ent}} = 5$ entités et les $k_{\text{hyp}} = 10$ hyperarêtes. Nous utilisons NetworkX pour la gestion de l'hypergraphe et NanoVectorDB pour le stockage et la recherche vectorielle des chunks. NanoVectorDB est une base vectorielle légère, conçue pour stocker et rechercher des embeddings en mémoire.

4.2 Résultats

Comparaison avec les baselines : Le Tableau 1 présente les résultats de notre évaluation comparative sur les trois jeux de données (Fiction, CS, MAUD). Notre approche (HyperGraphRAG + PPR + EXT++) obtient la meilleure performance ou se place en seconde position sur la quasi-totalité des métriques. Elle réalise un gain remarquable en rappel contextuel (+51% sur Fiction et +69% sur MAUD), ainsi que la complétude, avec +11% sur ces deux corpus par rapport à HyperGraphRAG. Ces gains illustrent la complémentarité de PPR et d'EXT++ : le premier améliore la sélection des passages pertinents via l'exploration probabiliste du graphe, le second renforce la qualité du graphe sous-jacent en réduisant les hyperarêtes isolées et redondantes. Par rapport au Standard RAG, les écarts sont encore plus importants : la complétude progresse de +59% sur Fiction et +31% sur CS, et les gains dépassent +150% sur MAUD en exactitude comme en complétude, confirmant l'incapacité de la recherche vectorielle seule à traiter des corpus multi-sauts (Fiction, CS), mais aussi des corpus spécialisés comme MAUD, où la terminologie standardisée et la redondance du langage contractuel rendent la discrimination entre passages particulièrement difficile.
L'analyse des résultats met en évidence les limitations structurelles de chaque baseline que notre approche résout progressivement. Le Standard RAG, fondé sur une simple similarité vectorielle entre la requête et des chunks, ne dispose d'aucun mécanisme explicite de structuration contextuelle ni de raisonnement multi-hop. Il échoue particulièrement dans MAUD (rappel contextuel=10%, exactitude de réponse=3,38) où de nombreux chunks sémantiquement très proches les uns des autres, empêchant la recherche dense de discriminer le passage pertinent parmi des dizaines de formulations quasi-identiques et explique la forte chute de rappel. Le Contextual Chunking d'Anthropic atténue en partie ce problème en enrichissant chaque chunk avec un résumé contextuel du document auquel il appartient. Cette contextualisation renforce les signaux sémantiques utiles et permet au moteur de recherche dense de mieux identifier le document pertinent. Elle s'avère particulièrement efficace sur MAUD, où le contexte du chunk fournit des indices forts permettant d'identifier le bon contrat parmi des documents à la formulation très similaire — par exemple, localiser le bon merger agreement contenant les covenants recherchés. En revanche, cette stratégie introduit également un effet secondaire : l'ajout systématique de contexte peut amplifier certains signaux lexicaux fréquents mais peu discriminants dans le corpus. Lorsque ces signaux dominent les embeddings, la recherche dense peut être détournée vers des passages contextuellement proches mais non pertinents. Ce phénomène est particulièrement visible dans des corpus narratifs comme Fiction, où l'enrichissement contextuel améliore légèrement la récupération mais ne se traduit que par un gain limité en exactitude (8,00), car les indices contextuels restent diffus et difficiles à exploiter pour identifier précisément l'information cible. Le Graph Transformer de LangChain, qui extrait un graphe de connaissances binaire et récupère les entités voisines à un saut (*hop-1*), souffre d'un double problème de fragmentation et de bruit. D'une part, le graphe binaire est souvent très fragmenté, ce qui rend difficile l'expansion de contexte sans introduire d'informations non pertinentes. D'autre part, il existe un compromis intrinsèque entre la profondeur de récupération et la pertinence : une profondeur faible produit un contexte insuffisant, tandis qu'une profondeur élevée, en l'absence d'une stratégie d'élagage intelligente, introduit un bruit considérable qui dégrade la qualité des réponses. HippoRAG2 pallie partiellement ces limitations en appliquant Personalized PageRank sur un graphe binaire, offrant un mécanisme de propagation probabiliste plus sophistiqué que la simple traversée *hop-1*. Cela améliore certains scores, mais la complétude reste faible car les triplets binaires ne capturent pas fidèlement les relations n-aires, temporelles et modales fréquentes dans les corpus.
Notre approche résout ces limitations en étendant l'approche HyperGraphRAG par la combinaison de trois mécanismes complémentaires. D'abord, l'hypergraphe d'origine offre déjà une représentation sémantiquement plus riche que les triplets binaires et un graphe naturellement moins fragmenté — ce qui explique le bond de performance de

HyperGraphRAG par rapport aux baselines à graphe binaire [14]. Cependant, sans mécanisme de propagation globale, la sélection des passages repose uniquement sur la similarité locale entre la requête et les entités du graphe, limitant le rappel contextuel. L'intégration de PPR résout cette limitation en permettant une propagation de pertinence probabiliste et pondérée à travers l'ensemble du graphe, évitant l'introduction de bruit grâce à l'atténuation des scores avec la distance topologique. Enfin, EXT^{++} intervient en amont en améliorant la qualité de l'hypergraphe lui-même, réduisant la fraction d'hyperarêtes isolées et renforçant la connectivité du graphe. Cela se traduit par un gain léger en rappel et en complétude des réponses. La synergie entre ces trois composants explique la supériorité de notre approche sur l'ensemble des jeux de données. Toutefois, sur MAUD, le rappel contextuel de notre approche reste inférieur à celui d'HippoRAG2. La redondance du langage contractuel pénalise les hyperarêtes, plus riches en contenu qu'un simple triplet : leurs formulations se chevauchent davantage, bruitant le signal de similarité lors de l'initialisation du PPR. Les triplets binaires d'HippoRAG2, plus courts et discriminants, offrent un ancrage plus précis dans ce type de corpus. En revanche, cette richesse explique la supériorité en complétude (un gain +41%) : les entités, hyperarêtes et chunks récupérés fournissent au LLM un contexte sémantiquement plus dense, permettant des réponses plus exhaustives même à partir d'un ensemble de passages plus restreint.

Comparaison de performance des modèles LLM : La Figure 3 et le Tableau 2 présentent respectivement les scores de qualité et le coût moyen par requête obtenus par quatre modèles de génération (Claude Sonnet 4.5, GPT-5.1, Gemini Flash 3 et GPT-4o mini) utilisés exclusivement à l'étape de génération de réponse dans notre approche. L'extraction des entités et relations demeure assurée par GPT-4o mini pour l'ensemble des configurations expérimentales. Ce choix architectural est motivé par des contraintes économiques : l'extraction d'un hypergraphe requiert un appel au modèle de langage pour chaque chunk du corpus, soit par exemple plus de 1 500 appels pour le jeu de données MAUD. L'utilisation d'un modèle coûteux à cette étape engendrerait un surcoût prohibitif — un facteur 25 entre GPT-4o mini et Sonnet 4.5 — rendant l'indexation économiquement non viable à grande échelle. La comparaison des modèles sur la génération de réponse est importante car ils diffèrent dans leur capacité à identifier et synthétiser l'information pertinente dans le contexte récupéré contenant du bruit.

Model	Fiction	MAUD	CS
Sonnet 4.5	\$0.0720	\$0.0900	\$0.0660
GPT-5.1	\$0.0261	\$0.0360	\$0.0214
Gemini Flash 3	\$0.0087	\$0.0121	\$0.0087
GPT-4o mini	\$0.0028	\$0.0036	\$0.0026

TABLE 2 – Coût moyen par requête selon le modèle et le cas d'usage

Les résultats révèlent que les scores d'*exactitude* demeurent

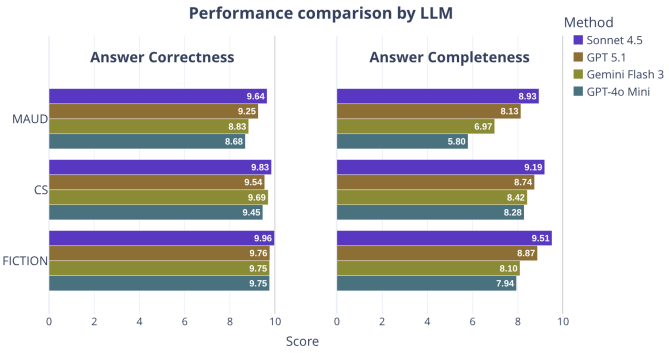

FIGURE 3 – Comparaison de performance de modèles LLM pour la génération de réponse

élevés et relativement homogènes entre les modèles (8,68 à 9,96 sur une échelle de 10), suggérant que l'architecture de récupération par hypergraphe fournit un contexte suffisamment pertinent pour permettre à l'ensemble des modèles évalués de produire des réponses factuellement correctes. En revanche, la métrique de *complétude* met en évidence des écarts plus prononcés : Sonnet 4.5 atteint des scores de 9,51 sur Fiction et 8,93 sur MAUD, tandis que GPT-4o mini obtient respectivement 7,94 et 5,80, soit une dégradation entre 16% et 35%. Cette disparité indique que les modèles plus puissants excellent à discriminer les passages pertinents et à couvrir exhaustivement les informations clés, là où les modèles plus compacts tendent à omettre certains éléments dans les domaines spécialisés.

Ce compromis coût-qualité offre une flexibilité opérationnelle selon les exigences applicatives. Pour des cas d'usage critiques tels que l'analyse juridique ou la due diligence, l'investissement dans un modèle lecteur de haute capacité est justifié, le surcoût par requête demeurant modéré ($\sim$0,07\$). Pour des applications tolérant une complétude légèrement réduite ou des domaines narratifs moins exigeants, GPT-4o mini constitue une alternative économiquement viable tout en préservant un niveau d'exactitude satisfaisant.

Impact de l'extraction EXT^{++} : Nous examinons le bénéfice de l'extraction optimisée EXT^{++} sur la qualité structurelle des hypergraphes construits. Comme l'illustre la Figure 4, EXT^{++} réduit significativement la fraction d'hyperarêtes isolées sur les trois jeux de données, faisant passer le taux d'isolation de 62% à 20% sur Fiction. Par conséquent, la quasi-totalité des faits extraits se trouve ancrée à au moins une entité, ce qui renforce le signal relationnel et assure une bien meilleure navigabilité du graphe lors de la phase de recherche. Parallèlement, l'augmentation du degré moyen des entités témoigne d'une densification du réseau sémantique ; cette topologie favorise le raisonnement multi-sauts en multipliant les chemins de propagation de l'information. Cette optimisation produit également des descriptions d'hyperarêtes plus longues et informatives (par exemple, de 28 à 49 tokens pour MAUD, soit une hausse de 72%), enrichissant ainsi la représentation sémantique. Pour MAUD, le nombre moyen d'entités par hyperarête connec-

tée passe de 1,96 à 2,41 (+23%), illustrant une meilleure modélisation des relations multi-entités, et ce malgré une baisse négligeable (inférieure à 3%) sur les deux autres jeux de données. En résumé, EXT^{++} produit systématiquement des hypergraphes plus connectés, plus descriptifs et structurellement plus riches, ce qui se traduit directement par une meilleure couverture et une meilleure qualité de réponse lors du raisonnement.

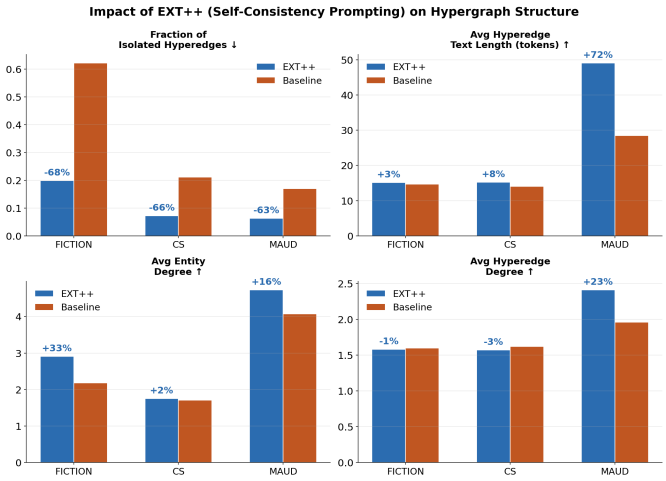

FIGURE 4 – Amélioration de la structure des hypergraphes par la méthode EXT^{++}

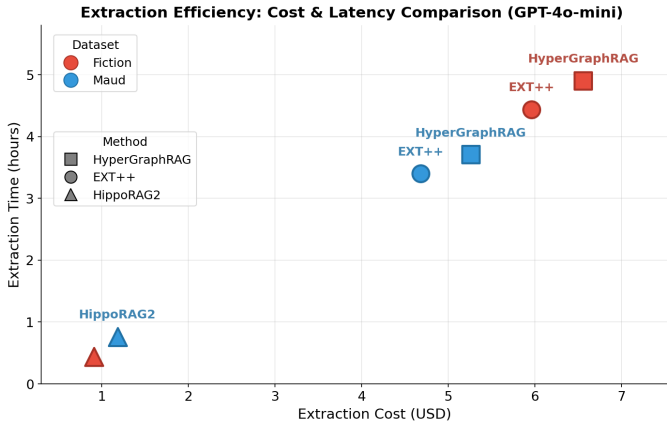

FIGURE 5 – Comparaison du coût et de la latence d'extraction entre HyperGraphRAG, EXT^{++} et HippoRAG2

Coût et latence de l'extraction : La Figure 5 compare le coût et la latence totale de l'extraction entre HyperGraphRAG (avec et sans EXT^{++}) et HippoRAG2 sur les jeux de données Fiction et MAUD. Contrairement à l'intuition, EXT^{++} — qui demande au modèle d'effectuer trois passes d'extraction internes avant d'émettre l'union dédupliquée — s'avère plus rapide et moins coûteux que l'extraction simple. Ce résultat s'explique par deux facteurs : (i) le prompt plus long d'EXT^{++} (600 tokens supplémentaires) bénéficie davantage du cache de préfixes (+20 à +24% de tokens mis en cache), réduisant le coût de complétion ; (ii) l'agrégation interne fusionne les extractions redondantes avant de générer la sortie, réduisant le nombre de tokens de complétion (-20% sur Fiction, -14% sur MAUD). La latence LLM étant dominée par le décodage autorégressif, cette réduction se traduit directement par un temps d'extraction plus court.

En revanche, HippoRAG2 affiche des performances nettement supérieures en termes de coût et de latence, avec un facteur d'environ 5× à 7× selon la métrique. Toutefois, cette efficacité découle directement de la nature de sa représentation : HippoRAG2 extrait un graphe de connaissances binaire intrinsèquement moins verbeux qu'un hypergraphe. Cette concision a un coût qualitatif : les graphes binaires sont connus pour leurs difficultés à représenter fidèlement les relations temporelles, les modalités épistémiques, ainsi que les relations n-aires fréquentes dans les domaines complexes. L'hypergraphe, bien que plus coûteux à construire, préserve la richesse sémantique du texte source et offre une meilleure robustesse face à ces limitations.

Latence de Personalized PageRank : Nous évaluons le coût computationnel introduit par l'algorithme de Personalized PageRank (PPR) en comparant la latence de récupération par requête avec et sans cette optimisation. Les résultats dans la Figure 6 montrent une latence remarquablement faible dans les deux configurations. Sans PPR, le temps moyen par requête se situe entre 0,135 s et 0,146 s selon le jeu de données. L'activation du PPR n'engendre qu'un surcoût inférieur à 0,2 secondes par requête sur l'ensemble des corpus testés. Ce comportement uniforme s'explique par le fait que le PPR opère sur un sous-graphe local centré sur les entités pertinentes à la requête, et non sur la totalité de l'hypergraphe de connaissances. Ce surcoût demeure négligeable dans un scénario de récupération augmentée par graphe, compte tenu de l'amélioration de la qualité du contexte fourni au modèle.

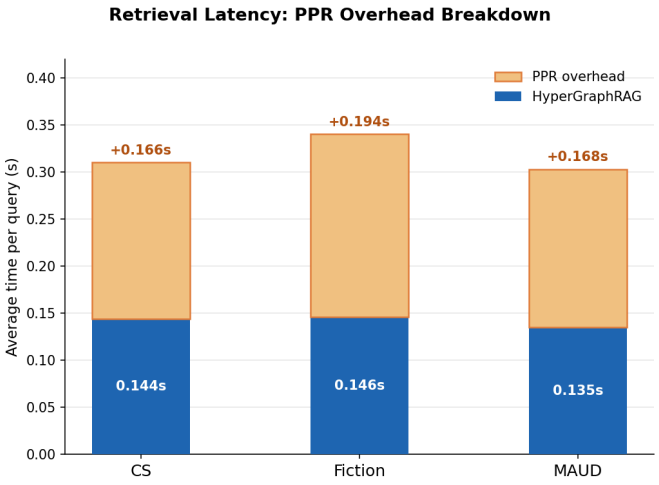

FIGURE 6 – Latence moyenne de récupération par requête avec et sans PPR

5 Conclusion et perspectives

Ce travail a étendu l'approche HyperGraphRAG par l'intégration de Personalized PageRank (PPR) et de l'extraction self-consistency (EXT^{++}). Le PPR permet une propagation de pertinence probabiliste à travers l'hypergraphe, améliorant significativement le rappel contextuel et la complétude des réponses. EXT^{++} fiabilise le graphe en amont en réduisant les hyperarêtes isolées, sans surcoût en latence ni en coût d'extraction. L'évaluation sur trois corpus de natures distinctes — narratif, scientifique et juridique — confirme la supériorité de notre approche, avec des gains particulièrement marqués en exactitude et en complétude de réponse.

Pour les perspectives, plusieurs axes sont envisagés. D'abord, des méthodes de diffusion nativement conçues pour les hypergraphes — marches aléatoires sur hypergraphes [20] — pourraient mieux exploiter la structure n-aire des hyperarêtes que la diffusion PPR. Ensuite, repenser chaque hyperarête comme un résumé condensé de fragments atomiques plutôt que des phrases verbatim réduirait le volume de tokens produits, diminuant ainsi le coût et la latence d'extraction. Par ailleurs, évaluer notre approche sur des bases vectorielles exploitant des algorithmes de recherche avancés permettrait de mesurer l'impact sur des benchmarks single-hop spécialisés à vocabulaire redondant, tels que MAUD. Enfin, un mécanisme de *pruning* du contexte permettrait d'optimiser l'usage des tokens et réduire le bruit contextuel avant la génération, ouvrant la voie à l'évaluation sur d'autres benchmarks nécessitant un raisonnement plus profond ou des synthèses complexes.

Références

[1] Anthropic. Introducing contextual retrieval, 2024. https://www.anthropic.com/engineering/contextual-retrieval.

[2] Boyu Chen, Zirui Guo, Zidan Yang, Yuluo Chen, Junze Chen, Zhenghao Liu, Chuan Shi, and Cheng Yang. Pathrag : Pruning graph-based retrieval augmented generation with relational paths, 2025.

[3] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language models. In *ICML 2024 Workshop on In-Context Learning*, 2024.

[4] Darren Edge, Ha Trinh, Yanlo Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Shweti Truitt, and Gagan Bansal. From local to global : A graph rag approach to query-focused summarization. *arXiv preprint arXiv :2404.16130*, 2024.

[5] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. RAGAs : Automated evaluation of retrieval augmented generation. In Nikolaos Aletras and Orphee De Clercq, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics : System Demonstrations*, 2024.

[6] Yifan Feng, Hao Hu, Xingliang Hou, Shiquan Liu, Shihui Ying, Shaoyi Du, Han Hu, and Yue Gao. Hyper-rag : Combating llm hallucinations using hypergraph-driven retrieval-augmented generation, 2025.

[7] Zirui Guo, Zhaojun Fan, Zhuoheng Lu, Jun Xu, and Ji-Rong Wen. Lightrag : Constructing and utilizing light-weight knowledge graph for retrieval-augmented generation. *arXiv preprint arXiv :2410.05779*, 2024.

[8] Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory : Non-parametric continual learning for large language models. In *Forty-second International Conference on Machine Learning*, 2025.

[9] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever : Retrieval-augmented generation for textual graph understanding and question answering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[10] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. GRAG : Graph retrieval-augmented generation. In *Findings of the Association for Computational Linguistics : NAACL*, 2025.

[11] Yiqian Huang, Shiqi Zhang, and Xiaokui Xiao. Ketrag : A cost-efficient multi-granular indexing framework for graph-rag. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, 2025.

[12] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

[13] Lei Liang, Zhongpu Bo, Zhengke Gui, Zhongshu Zhu, Ling Zhong, Peilong Zhao, Mengshu Sun, Zhiqiang Zhang, Jun Zhou, Wenguang Chen, Wen Zhang, and Huajun Chen. Kag : Boosting llms in professional domains via knowledge augmented generation. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, page 334–343, 2025.

[14] Haoran Luo, Haihong E, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, and Luu Anh Tuan. Hypergraphrag : Retrieval-augmented generation via hypergraph-structured knowledge representation, 2025. NeurIPS 2025 poster.

[15] Neo4j. Creating knowledge graphs from unstructured data. https://neo4j.com/developer/genai-ecosystem/importing-graph-from-unstructured-data/.

[16] Nicholas Pipitone and Ghita Houir Alami. Legalbench-rag : A benchmark for retrieval-augmented generation in the legal domain. *arXiv preprint arXiv :2408.10343*, 2024.

[17] Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. Memorag : Boosting long context processing with global memory-enhanced retrieval augmentation. In *Proceedings of the ACM Web Conference)*, 2025.

[18] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda*, 2023.

[19] Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, and Enhong Chen. Large language models for generative information extraction : a survey. *Frontiers of Computer Science*, 18, 2024.

[20] Mu-Rong Yang and Xin-Jian Xu. Recent advances in hypergraph neural networks. *Journal of the Operations Research Society of China*, 2025.

[21] Xiangjun Zai, Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, and Wenjie Zhang. Proh : Dynamic planning and reasoning over knowledge hypergraphs for retrieval-augmented generation, 2026.

[22] Zulun Zhu, Tiancheng Huang, Kai Wang, Junda Ye, Xinghe Chen, and Siqiang Luo. Graph-based approaches and functionalities in retrieval-augmented generation : A comprehensive survey. *ACM Comput. Surv.*, 2026.

A Prompts

Le Prompt 1 présente les instructions utilisées par la méthode EXT^{++} pour l'extraction optimisée d'un hypergraphe de connaissances.

Given a text document, identify all entities and relationships present within the text.

 # Self-Consistency Protocol

1. Extraction Runs :
   — Internally perform entities and knowledge segments extraction three times independently.
   — Each run must complete all steps as a separate, uninfluenced process.
2. Aggregation and Deduplication after completing all extraction runs :
   — For Knowledge Segments (hyper-relations) :
      — Merge semantically similar or overlapping knowledge segments into a single, comprehensive segment.
      — If multiple segments describe the same concept/relationship, combine them into ONE segment with the highest completeness score.
   — For Entities :
      — Merge entities that refer to the same concept even if worded differently and choose the most specific entity name.
      — If contradictions are found, resolve in favor of the most consistent information.

 # Extraction Steps

1. Divide the text into complete knowledge segments (hyper-relations). For each knowledge segment, extract the following information :
   — `knowledge_segment` : Extract 1–3 consecutive sentences EXACTLY as they appear in the source text. DO NOT paraphrase, modify, or rewrite the text. The extracted span should be semantically complete and self-contained.
   — `completeness_score` : A score from 0 to 10 indicating the completeness of the knowledge segment.
2. For each knowledge segment, identify all related entities. For each identified entity, extract the following information :
   — `entity_name` : Name of the entity. Resolve coreferences based on the whole context and provide consistent naming.
   — `entity_type` : Type of the entity.
   — `entity_description` : Comprehensive description of the entity's attributes and activities.

Output each knowledge segment followed immediately by its associated entities.

Prompt 1: Prompt d'extraction par auto-cohérence (EXT^{++}) pour la construction de l'hypergraphe