Recherche d'information juridique par réseaux de neurones de graphe sur un corpus de droit

Gabriel Kerdraon[1] Nada Mimouni[1]
(1) CNAM, CEDRIC, 292 rue Saint-Martin, 75003 Paris, France
gabriel.kerdraon.auditeur@lecnam.net, nada.mimouni@lecnam.net

RÉSUMÉ

Les praticiens du droit peinent souvent à identifier les textes applicables en raison tant de la volumétrie que de la dispersion des sources normatives. Nous avons développé un prototype fondé sur une architecture de type Retrieval-Augmented Generation, dont l'usage en situation concrète s'avère cependant coûteux sur le plan computationnel. Nous proposons une alternative reposant sur la modélisation du corpus sous forme de graphe. Des réseaux de neurones de graphe propagent l'information du contexte, enrichissent les représentations denses des articles et améliorent la récupération des textes pertinents en amont de la génération. Les résultats montrent que l'exploitation explicite de la structure permet de simplifier l'architecture tout en maintenant, voire en améliorant, les performances lorsque certains modules du pipeline initial sont conservés.

ABSTRACT

Legal Information Retrieval Using Graph Neural Networks on a Legal Corpus

Legal practitioners often struggle to identify the applicable legal provisions due both to the sheer volume of legal materials and the dispersion of normative sources. We developed a prototype based on a Retrieval-Augmented Generation architecture ; however, its use in practical settings proves computationally expensive. We therefore propose an alternative approach based on modeling the corpus as a graph. Graph neural networks propagate contextual information, enrich the dense representations of legal articles, and improve the retrieval of relevant texts upstream of the generation step. The results show that explicitly exploiting structural information makes it possible to simplify the architecture while maintaining, and in some cases improving, performance when certain modules of the initial pipeline are retained.

MOTS-CLÉS : recherche d'information, droit, question/réponse réglementaire, recherche d'information par vecteurs denses, réseaux de neurones sur graphes, corpus législatif français.

KEYWORDS: information retrieval, legal domain, regulatory question answering, dense retrieval, graph neural networks, French regulatory corpus.

1 Introduction

Ce travail s'inscrit dans le contexte d'un service d'information juridique au sein duquel les agents rencontrent des difficultés récurrentes pour accéder aux textes nécessaires au traitement des dossiers quotidiens. La diversité des situations traitées et la dispersion des sources (bases juridiques natio-

nales ou supranationales, circulaires internes, documents d'intranet ou échanges informels) rendent complexe l'identification rapide des dispositions pertinentes.

Cette surcharge documentaire, soulignée dans la littérature récente sur l'extraction et la recherche d'information juridique, constitue un défi persistant pour les environnements professionnels (Premasiri *et al.*, 2025). En pratique, chaque agent se constitue son propre « corpus » personnel informel, ce qui favorise des pratiques hétérogènes, limite la mutualisation des connaissances et rend plus difficile l'intégration des nouveaux agents. Si la demande initiale portait sur la génération automatique de réponses juridiques complètes, l'analyse des usages a montré que l'attente principale concernait avant tout l'identification fiable des textes applicables et un accès facilité aux sources. Ce constat rejoint des travaux récents montrant que, dans les contextes juridiques, la qualité de la récupération d'articles constitue le facteur déterminant de l'utilité opérationnelle des systèmes (Nguyen *et al.*, 2025).

Un premier prototype fondé sur une architecture de type Retrieval-Augmented Generation a été développé pour répondre à ce besoin. Bien que jugé opérationnellement satisfaisant, ce pipeline mobilisait à plusieurs reprises des modèles de langage ainsi que des mécanismes de reclassement de type cross-encoder, ce qui en augmentait sensiblement le coût de calcul par requête et limitait sa capacité à être déployé à grande échelle. Les expérimentations ainsi que les retours d'usage recueillis lors de la conception et du déploiement du prototype ont conduit à déplacer la réflexion vers la modélisation même du corpus juridique. Plutôt que d'optimiser marginalement les différentes briques du pipeline, était-il possible d'exploiter directement la structure du corpus afin d'enrichir les représentations ? L'objectif devient alors double : renforcer la robustesse de la récupération d'articles et simplifier l'architecture du système. La suite de l'article est structurée comme suit. La section 2 présente l'état de l'art et positionne notre contribution. La section 3 décrit les deux méthodes de recherche d'information proposées. La section 4 analyse les résultats expérimentaux. La section 5 conclut.

2 Etat de l'Art

Les systèmes de *question answering* juridique sont fréquemment organisés en pipeline à deux étapes, avec une récupération rapide (*retriever*) suivie d'un reclassement plus coûteux (*re-ranker*). Ce schéma est devenu standard dans les approches de *dense retrieval* ainsi que dans l'usage de *cross-encoders* pour le reclassement (Karpukhin *et al.*, 2020). Malgré ces avancées, la récupération d'information juridique demeure un problème ouvert, en particulier lorsque la pertinence dépend de relations (négations, précisions, abrogations, etc.) qui ne sont pas directement présentes dans l'unité documentaire, souvent l'article ou le jugement. Dans les textes juridiques, cette difficulté est accentuée par les choix techniques de segmentation utilisés dans les systèmes modernes de récupération. Les articles présentent en effet une forte variabilité de longueur ; pour cette raison, de nombreux systèmes effectuent la récupération au niveau de passages, avant d'agréger les résultats vers l'unité documentaire pertinente(Callan, 1994). Ce principe, historiquement lié à la localisation de la pertinence dans les documents longs, est aujourd'hui renforcé par les contraintes de longueur des encodeurs neuronaux. Toutefois, ce découpage fragmente artificiellement des unités normatives continues ; une modélisation explicite de la structure permet alors de réintroduire un contexte global dépassant le segment local.

Indépendamment des contraintes techniques, la pertinence juridique est en outre structurellement relationnelle. La théorie du droit souligne depuis longtemps que les normes s'inscrivent dans un ordre hiérarchisé et interdépendant (Hart, 1961). L'interprétation d'un article dépend ainsi de sa position

dans la hiérarchie normative et de ses renvois explicites vers d'autres dispositions, ce qui conduit certains auteurs à décrire le droit non seulement comme une pyramide, mais aussi comme un réseau (Ost & van de Kerchove, 2002). Le sens ne résulte donc pas uniquement du contenu textuel d'un segment, mais de son inscription dans une organisation formelle plus large.

Dans un cadre de recherche d'information général, la structure interne d'un document (sections, sous-sections, voisinages) peut être modélisée comme un graphe et exploitée pour améliorer la récupération de passages (Albarède *et al.*, 2022). Dans le domaine juridique, (Louis *et al.*, 2023) introduisent un *graph-augmented dense statute retriever* (G-DSR) pour la récupération d'article (*statutory article retrieval*) : les représentations d'articles issues d'un retriever dense sont enrichies par propagation sur un graphe législatif, ce qui améliore la qualité de récupération sur un jeu annoté par experts.

En parallèle, les travaux en *case retrieval*, c'est-à-dire en recherche de décisions de justice, montrent que des graphes structuraux peuvent aussi être exploités pour contourner la longueur des textes et injecter de l'information relationnelle. CaseGNN représente chaque dossier comme un graphe attribué (*text-attributed case graph*) et apprend une représentation par propagation, avec des gains sur les benchmarks COLIEE (Tang *et al.*, 2024a). CaseLink étend cette idée en modélisant une connectivité globale entre plusieurs affaires afin d'améliorer la récupération lorsque les liens explicites ne sont pas présents (Tang *et al.*, 2024b). D'autres travaux soulignent également l'intérêt de graphes hétérogènes pour la recommandation de documents juridiques (Yang *et al.*, 2022) et de jeux de données structurés autour de réseaux de citations comme benchmarks (Harde *et al.*, 2025).

Ces travaux convergent vers l'idée que l'intégration explicite de la structure et des relations entre unités juridiques peut améliorer les performances. Dans notre cas, nous avons conçu une méthode de RI reposant sur une architecture hybride (reformulation, recherche dense et reclassement). La structure hiérarchique des codes y est d'abord introduite de manière implicite, manuellement, sous forme de chaînes de caractères concaténées aux segments textuels ; cette méthode constitue le cas de base. Nous proposons ensuite une seconde méthode qui remplace cette contextualisation textuelle par une modélisation explicite du graphe législatif, afin d'enrichir les représentations d'articles et de réduire la dépendance à certaines briques coûteuses du pipeline initial.

3 Matériel et Méthodes

La tâche étudiée correspond à une recherche d'articles juridiques à partir d'une question en langage naturel. Nous partons d'une architecture de référence, reposant sur une phase de récupération suivie d'un classement plus fin. Nous en analysons les limites et comparons cette référence à une variante dans laquelle nous remplaçons certaines composantes du pipeline initial par un encodeur hiérarchique et un réseau de neurones sur graphe (GNN) exploitant explicitement la hiérarchie et les citations du corpus.

3.1 Données

Trois jeux de données ont été mobilisés à des fins distinctes d'entraînement et d'évaluation. Le jeu BSARD (*Belgian Statutory Article Retrieval Dataset*) (Louis & Spanakis, 2022) est utilisé exclusivement en phase d'entraînement préalable. Il comprend 1,108 questions juridiques en français, annotées par des juristes, associées à des articles pertinents issus d'un corpus de 22,600 dispositions législatives belges. Le jeu comprend également un ensemble distinct de questions synthétiques destiné

à fournir des exemples supplémentaires pour l'entraînement de modèle. Il constitue un jeu supervisé de recherche d'articles législatifs en langue française, distinct du corpus du cas d'usage.

Le corpus opérationnel constitue le cœur de l'étude. Il est issu d'un recensement avec des experts métiers et comprend 25,017 articles issus de codes et textes pertinents pour l'activité considérée. À partir de ces articles, un graphe juridique est construit via l'API Légifrance : le chemin hiérarchique complet (textes racines, livres, titres, chapitres, sections) de chaque article ainsi que ses citations sortantes sont extraits, fournissant respectivement les nœuds intermédiaires et les arêtes du graphe. L'ensemble conduit à un graphe de 79,118 nœuds, comprenant 70,513 articles, 8,502 sections et 103 textes racines, reliés par 301,146 arêtes.

Un jeu d'entraînement synthétique interne de 4,500 paires question–article a été généré à partir du corpus cible. Les questions ont été dérivées des articles, puis paraphrasées par un modèle de langage avant filtrage (manuel et automatique). Ce choix fait suite à deux contraintes. D'une part, il n'existe pas de jeu public de question–réponse sur la loi française sur une tâche de recherche d'articles. D'autre part, l'apprentissage du graphe nécessite des questions alignées avec des articles effectivement présents dans le corpus étudié. Même en présence d'un jeu de données externe, un sous-échantillonnage aligné avec le corpus aurait été nécessaire, impliquant probablement la génération de questions complémentaires.

L'évaluation finale repose sur un jeu de test distinct composé de 186 questions annotées manuellement par des experts, chacune associée à un ou plusieurs articles pertinents.

3.2 Système de recherche d'information par similarité sémantique et reclassement

L'architecture de référence est un pipeline dense en plusieurs étapes comme le montre la Figure 1. Comme vu précédemment, les articles juridiques présentent de fortes variabilités de longueur, allant de quelques lignes à plusieurs pages. Ceci pose un double problème : d'une part, les contraintes de longueur des modèles d'encodage qui imposent un découpage des articles, d'autre part, le risque de dilution sémantique lorsque des textes longs sont encodés globalement (Beltagy *et al.*, 2020; Callan, 1994). Nous adoptons une stratégie de type *parent–child retrieval*, dans laquelle la récupération est

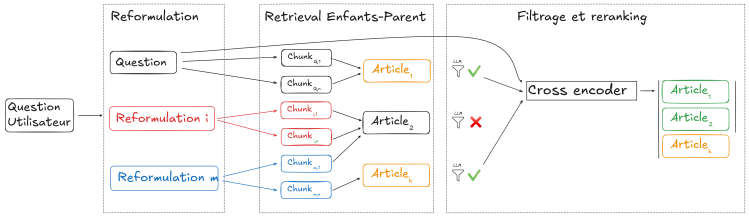

FIGURE 1 – Architecture du pipeline de référence.

effectuée au niveau de segments enfants, tandis que le score final pour le classement est calculé au niveau de l'unité parent (l'article). Les articles sont découpés en fenêtres chevauchantes de taille fixe. Chaque segment, ou « chunk enfant », contient au plus N caractères, avec un recouvrement de k caractères entre segments adjacents afin d'éviter les coupures artificielles au niveau des définitions, listes, exceptions ou renvois internes (Khalid & Verberne, 2008). Lors de la phase de récupération, la similarité est calculée au niveau des segments.

Chaque segment est encodé par un modèle dense. Deux encodeurs sont considérés : CamemBERT-base [1] et Solon-embeddings-large-0.1 [2], modèle spécialisé pour la similarité sémantique en français incluant le domaine juridique.

CamemBERT-base a d'abord été utilisé comme encodeur généraliste de référence. Nous avons ensuite évalué Solon-embeddings-large-0.1, dont les performances en similarité sémantique se sont révélées supérieures dans notre contexte expérimental. Les deux configurations sont conservées dans l'évaluation afin de comparer leur comportement dans les différentes variantes de l'architecture.

Au moment de la requête utilisateur, une étape de reformulation contrôlée génère un nombre limité de variantes afin d'élargir la couverture lexicale (Ma *et al.*, 2023; Wang *et al.*, 2023). Pour chaque question (originale + reformulation), une récupération par proximité sémantique est effectuée afin d'extraire les n segments enfants les plus proches. Les blocs parents correspondants sont ensuite remontés et dédoublonnés. Les candidats sont filtrés par un modèle de langage pour éliminer les résultats hors sujet, puis reclassés via un cross-encodeur calculant une similarité plus fine requête–article. Bien que jugée satisfaisante par les utilisateurs, cette architecture présente une limite principale. La similarité dense calculée sur des segments isolés génère de nombreux faux positifs et doublons, ce qui impose des étapes de filtrage et de reclassement coûteuses. Couplée à un nombre croissant d'utilisateurs, cette complexité se traduit par des temps de réponse pouvant atteindre plusieurs secondes, voire plusieurs dizaines de secondes, ce qui freine l'intégration de l'outil dans les usages.

Lors de la conception du pipeline, nous avons observé empiriquement que la concaténation explicite du chemin hiérarchique complet d'un article — par exemple *« Code de la commande publique / Partie législative / Première partie : Définitions et champ d'application / Livre Ier : Contrats de la commande publique / Titre Ier : Marchés publics / Chapitre Ier : Marchés / Section 1 : Définition / Article L1111-1 »* améliorait la qualité de la récupération. Ce résultat indique que l'information hiérarchique constitue un signal utile pour la désambiguïsation et la contextualisation des articles.

3.3 Système de recherche d'information enrichi par Graph Neural Network

3.3.1 Modélisation du corpus comme graphe.

Afin de capturer explicitement les relations internes au corpus, celui-ci est représenté sous la forme d'un graphe orienté hétérogène $G = (V, E)$. Les nœuds V correspondent aux articles, aux sections intermédiaires et aux textes racines. Deux types de relations sont modélisés. Les arêtes hiérarchiques relient chaque article à sa section parente, chaque section à son parent, et ainsi de suite jusqu'au texte racine. Chaque relation est dupliquée avec son inverse afin de distinguer explicitement la propagation ascendante et descendante dans la hiérarchie, ce qui permet d'apprendre des pondérations distinctes selon le sens de circulation de l'information. Les arêtes de citation relient un article source à chaque article explicitement cité dans son texte. Ces relations sont également dupliquées en sens inverse afin de permettre la propagation de l'information entre articles citants et cités. Les poids des arêtes selon leur type et direction (hiérarchique ascendante/descendante, citation entrante/sortante) sont appris dynamiquement via le mécanisme d'attention du GNN (voir section 3.3.3), plutôt que d'être fixés a priori. Une illustration d'un sous-graphe législatif extrait du corpus est présentée en annexe (Figure 3).

1. https://huggingface.co/camembert-base
2. https://huggingface.co/OrdalieTech/solon-embeddings-large-0.1

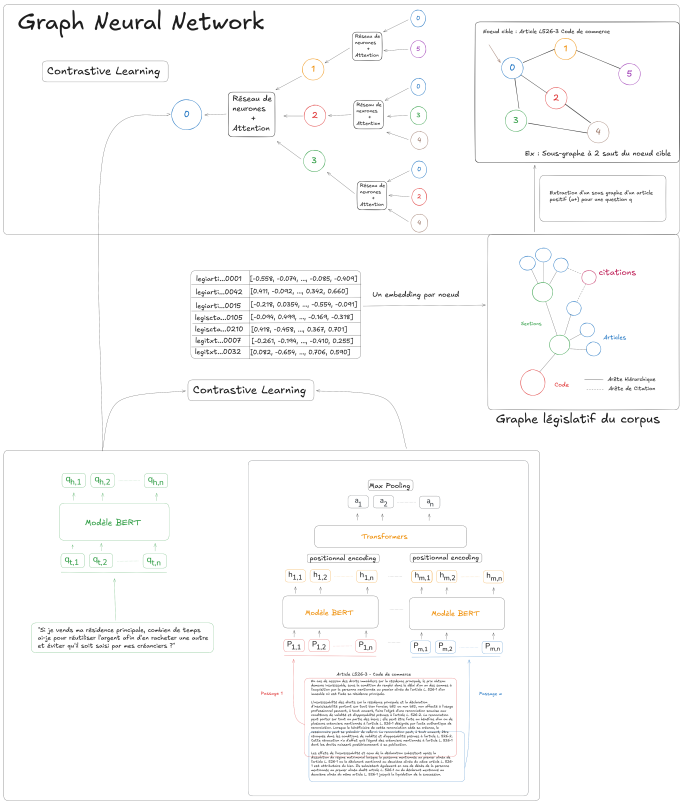

FIGURE 2 – Architecture encodeur hiérarchique + GNN.

Nous avons conçu un système de recherche d'information qui combine (1) un encodeur hiérarchique produisant une représentation unique par article en vue de les situer dans un graphe, et (2) une propagation structurée sur un graphe législatif via un réseau de neurones de graphe. Cette approche s'inscrit dans la lignée des travaux de (Louis *et al.*, 2023), dont elle se distingue par l'intégration des citations en complément de la hiérarchie ainsi que par la modélisation différenciée des types et directions de relations dans le graphe. La Figure 2 illustre les différentes étapes du processus d'entraînement des représentations d'articles enrichies par la propagation d'information structurelle via le GNN. Les nœuds du graphe représentent soit les articles (issus du corpus ou identifiés par citations), soit les différentes sections hiérarchiques ; dans ce dernier cas, l'attribut du nœud est le texte du titre de la section.

3.3.2 Représentations d'articles : encodeur hiérarchique

Les articles du corpus présentent une forte variabilité de longueur. Sur les 70,513 articles considérés, la longueur moyenne est de 2,360 caractères ($\sim$ 660 tokens), avec une médiane de 711 caractères. Le premier quartile est à 332 caractères et le troisième quartile à 1,633 caractères. Les extrêmes vont de 7 caractères à 433,301 caractères.

Afin d'obtenir une représentation unique par article sans perte arbitraire d'information due à des troncatures, nous adoptons un encodage hiérarchique (Pappagari *et al.*, 2019; Zhang *et al.*, 2019; Beltagy *et al.*, 2020). Un article est d'abord découpé en segments chevauchants de longueur maximale fixe. Chaque segment est encodé par un transformer. Plutôt que de retenir uniquement le jeton spécial [CLS], nous utilisons une moyenne masquée des vecteurs de sortie (Reimers & Gurevych, 2019), afin d'obtenir une représentation plus stable du segment :

$$h_s = \frac{1}{|T_s|} \sum_{t \in T_s} H_t \tag{1}$$

où T_s désigne l'ensemble des tokens valides du segment s, et H_t leur représentation contextuelle. Les représentations de chaque segments obtenus sont enrichies par des encodages positionnels, puis à nouveau passées dans un transformer inter-segments afin de modéliser les dépendances internes à l'article. Un *max pooling* produit un vecteur unique h_a pour l'article.

Une couche linéaire finale projette la représentation dans l'espace d'embedding utilisé pour la similarité. Les requêtes sont quant à elles encodées par un encodeur jumeau partageant l'espace vectoriel. Nous utilisons une perte contrastive de type InfoNCE (Oord *et al.*, 2018) avec une température $\tau \in \{0.1, 0.07\}$ selon l'encodeur. Chaque requête est associée à un article positif et à 5 négatifs durs obtenus par récupération lexicale préalable. La fonction de perte contrastive utilisée pour entraîner l'encodeur est définie comme suit :

$$\mathcal{L} = -\log \frac{\exp(\mathrm{sim}(q, a^+)/\tau)}{\sum_{a \in \mathcal{B}} \exp(\mathrm{sim}(q, a)/\tau)} \tag{2}$$

où a^+ est l'article pertinent, $\mathcal{B}$ l'ensemble des candidats du batch, sim une similarité cosinus et τ un paramètre de température.

L'entraînement de l'encodeur hiérarchique est réalisé en trois passes successives : d'abord sur le split synthétique de BSARD, puis sur son split d'entraînement, avant un réalignement final sur notre jeu synthétique interne. Cette organisation progressive permet d'aligner graduellement les représentations sur la distribution cible tout en conservant une diversité initiale d'exemples. Au terme de cette étape, chaque article est représenté par un vecteur dense unique h_a, et chaque requête par un vecteur h_q projeté dans le même espace latent. La tâche de recherche d'information se ramène alors à un calcul de similarité dans cet espace vectoriel partagé, permettant de classer les articles selon leur proximité sémantique avec la requête.

3.3.3 Propagation structurée par réseau de neurones de graphe

Les vecteurs d'articles issus de l'encodeur hiérarchique constituent l'initialisation des nœuds du graphe. Afin d'enrichir ces représentations par leur contexte structurel, nous mobilisons des modèles de type Graph Neural Networks (GNN), qui permettent d'apprendre des représentations de nœuds en agrégeant l'information provenant de leur voisinage dans un graphe (Hamilton *et al.*, 2017). Plusieurs familles de modèles étaient envisageables : Graph Convolutional Networks (GCN) (Kipf & Welling, 2017), GraphSAGE (Hamilton *et al.*, 2017), ou encore des architectures relationnelles telles que les R-GCN (Schlichtkrull *et al.*, 2018). Toutefois, notre graphe étant hétérogène et comportant plusieurs types d'arêtes (hiérarchiques ascendantes/descendantes et citations entrantes/sortantes), il est nécessaire de modéliser des contributions différenciées selon le type et la direction des relations.

Nous retenons donc une architecture de type Graph Attention Network (GAT) (Veličković *et al.*, 2018), étendue aux relations multiples. Le mécanisme d'attention permet d'apprendre dynamiquement des coefficients de pondération $\alpha_{ij}^{(r)}$ pour chaque voisin j d'un nœud i sous une relation r. Cette capacité à pondérer différemment les contributions des voisins nous semble pertinente dans un contexte juridique où toutes les dépendances peuvent ne pas jouer un rôle équivalent. À chaque couche l, la mise à jour d'un nœud i s'écrit :

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{j\in\mathcal{N}_r(i)} \alpha_{ij}^{(r)} W_r h_j^{(l)}\right) \tag{3}$$

où R est l'ensemble des types de relations, $\mathcal{N}_r(i)$ les voisins de i sous la relation r, W_r une matrice spécifique à la relation, et $\alpha_{ij}^{(r)}$ un coefficient d'attention appris. La profondeur est limitée à $L \leq 3$ couches, correspondant à quelques sauts dans le graphe.

Ce choix reflète à la fois une hypothèse métier, selon laquelle le contexte juridiquement pertinent demeure généralement local (hiérarchie immédiate, citations proches), et une contrainte computationnelle : au-delà de quelques *hops*, le voisinage croît rapidement, ce qui augmente significativement le coût mémoire et le temps d'entraînement.

L'entraînement reste sur une fonction de perte contrastive. L'apprentissage est réalisé sur des sous-graphes extraits dynamiquement autour des nœuds du batch. Les paires positives sont les couples requête–article pertinents, et les négatifs sont les autres articles du batch, considérés comme des *hard negatives* car souvent thématiquement proches en raison de l'échantillonnage par sous-graphe.

Après propagation, chaque article est représenté par un vecteur enrichi intégrant contenu textuel et contexte structurel. La récupération finale repose sur la similarité entre l'embedding de la requête et celles des représentations enrichies.

4 Résultats et analyse

4.1 Protocole d'évaluation et métriques

L'évaluation est limitée aux performances de récupération et exclut la génération de réponses. Toutes les configurations sont évaluées sur le même ensemble de 186 questions de référence annotées, en comparant l'architecture d'origine à des variantes enrichies. La qualité de la récupération est mesurée au niveau de l'article à l'aide du *Recall@k* et de l'*Average Precision@k*, macro-moyennés sur les requêtes, pour $k \in \{5, 10\}$. Le Recall@k correspond à la proportion d'articles pertinents présents parmi les k premiers résultats. L'AP@k correspond à la moyenne des précisions observées aux rangs où un article pertinent apparaît dans les k premiers résultats, normalisée par le nombre total d'articles pertinents pour la requête.

L'architecture de référence effectue la récupération au niveau de segments « enfants », tandis que l'évaluation est réalisée au niveau de l'article. Ce décalage dépend des choix de segmentation ; plusieurs configurations parent–enfant ont donc été testées, en faisant varier la taille des segments et les chevauchements. Seules les configurations principales sont présentées ici.

En complément des architectures proposées, BM25 est utilisé comme baseline de récupération lexicale en recherche d'information juridique (Rosa *et al.*, 2021). Nous comparons une version ajustée de BM25 ($k1 = 2{,}5$, $b = 0{,}2$) aux différentes configurations de récupération dense évaluées dans l'étude.

Les configurations évaluées sont les suivantes :

1. L'architecture de référence : un système de question-réponse fondé sur un pipeline de type Retrieval-Augmented Generation avec reclassement.
2. Une seconde architecture reposant sur un encodeur hiérarchique et une propagation sur GNN.
3. Ce même pipeline, couplé à une étape de reformulation multi-requêtes héritée de la première architecture.

Nous comparons deux encodeurs denses afin d'évaluer l'effet de la spécialisation au domaine par rapport aux choix d'architecture. CamemBERT constitue une référence généraliste, tandis que Solon représente une configuration dense spécialisée pour la similarité sémantique en français. Nous présentons dans la suite les performances obtenues selon les variations d'encodeur, de segmentation et d'intégration de la structure par graphe.

4.2 Performances de base

Le Tableau 1 compare le rappel moyen (Recall@5, Recall@10) des baselines en requête simple ($K = 1$) pour BM25, CamemBERT et Solon, en fonction de la taille des segments enfants. Le Tableau 2 donne les valeurs correspondantes de AP@5 et AP@10.

TABLE 1 – Recall@5 et Recall@10 selon la taille des segments enfants (baseline, $K = 1$).

Taille enfant (char)	BM25		CamemBERT		Solon	
	R@5	R@10	R@5	R@10	R@5	R@10
400	0,175	0,202	0,219	0,279	0,661	0,727
600	0,180	0,224	0,240	0,333	0,672	0,770
800	0,180	0,208	0,257	0,322	0,667	0,761

TABLE 2 – AP@5 et AP@10 selon la taille des segments enfants (baseline, $K = 1$).

Taille enfant (char)	BM25		CamemBERT		Solon	
	AP@5	AP@10	AP@5	AP@10	AP@5	AP@10
400	0,123	0,126	0,143	0,151	0,519	0,528
600	0,126	0,132	0,167	0,179	0,521	0,535
800	0,132	0,135	0,168	0,177	0,526	0,540

Plusieurs constats se dégagent des Tableaux 1 et 2 : sans surprise, les modèles denses surpassent nettement BM25, confirmant l'intérêt d'une représentation sémantique pour la recherche d'articles. L'écart est particulièrement marqué pour Solon, dont les performances sont significativement supérieures à celles de CamemBERT dans toutes les configurations. Cette différence souligne l'importance de la spécialisation au domaine juridique. La taille des segments a un impact limité. Le passage de 400 à 600 caractères améliore légèrement les résultats, tandis qu'un allongement supplémentaire n'apporte que peu de gains. Une fenêtre intermédiaire apparaît donc suffisante pour capturer l'essentiel de l'information pertinente. Enfin, l'écart entre Recall@5 et Recall@10 montre que des articles pertinents continuent d'apparaître entre les rangs 6 et 10 pour les deux modèles, mais Solon conserve un avantage net dès les premiers rangs. Globalement, la spécialisation de l'encodeur constitue le facteur explicatif principal des performances observées en configuration de base sans reformulation.

4.2.1 Impact de la reformulation multi-requêtes

La reformulation multi-requêtes ($K > 1$) est testée sur les mêmes configurations. Elle génère plusieurs variantes de la requête initiale, ce qui élargit le vocabulaire couvert. Les gains au rappel sont positifs mais décroissants. Plusieurs tailles de segments ont été évaluées ; la configuration à 600 caractères s'est révélée la plus performante de manière quasi systématique. Le Tableau 3 illustre ce comportement pour $K = 1, 2, 3$ avec cette segmentation.

TABLE 3 – Recall@10 selon K (segments de 600 char).

Encodeur	$K = 1$	$K = 2$	$K = 3$
BM25	0,224	0,263	0,282
CamemBERT	0,333	0,409	0,447
Solon	0,770	0,810	0,830

Les reformulations améliorent systématiquement le rappel, avec un effet plus marqué pour CamemBERT que pour Solon, déjà performant en requête simple. Les gains présentent toutefois des rendements décroissants : l'essentiel de l'amélioration est obtenu avec deux requêtes, la troisième apportant un bénéfice plus marginal. En pratique, $K = 2$ ou $K = 3$ constitue un compromis raisonnable entre couverture et coût computationnel.

4.3 Encodeur hiérarchique et GNN

Nous examinons d'abord l'effet de l'intégration de la structure du corpus par propagation sur graphe. Différentes variantes du GNN ont été évaluées, en faisant varier le nombre de couches (2 ou 3). Les résultats présentés correspondent à la configuration la plus performante, utilisant trois couches et les types du graphe.

Pour CamemBERT, le Tableau 4 récapitule les performances étape par étape. Base correspond à l'encodeur hiérarchique sans propagation ni reformulation.

L'introduction de l'encodeur hiérarchique produit un gain très marqué pour CamemBERT. L'ajout du GNN apporte ensuite un gain supplémentaire, plus modéré. La propagation sur le graphe enrichit les représentations en intégrant explicitement le contexte structurel, ce qui améliore à la fois le rappel et la précision.

TABLE 4 – Performances cumulées pour CamemBERT.

Configuration	R@5	AP@5	R@10	AP@10
Base (CamemBERT)	0,621	0,481	0,661	0,551
GNN (3 couches)	0,715	0,543	0,825	0,568

Pour Solon, l'amélioration est plus progressive comme le montre le Tableau 5. Le GNN apporte un gain mesuré mais systématique.

Ces résultats montrent que l'intégration explicite de la structure documentaire via l'encodage hiérarchique améliore nettement la récupération pour les deux encodeurs. L'ajout du GNN apporte

TABLE 5 – Performances cumulées pour Solon.

Configuration	R@5	AP@5	R@10	AP@10
Base (Solon)	0,760	0,520	0,795	0,552
GNN (3 couches)	0,795	0,538	0,835	0,582

ensuite un gain supplémentaire, plus modéré, en enrichissant les représentations par propagation sur le graphe.

4.3.1 Ajout des reformulations

Même si le GNN enrichit les représentations en intégrant la structure du corpus, la récupération reste sensible à la formulation initiale de la requête. Les reformulations constituent donc un levier complémentaire pour explorer plusieurs variantes sémantiques. Dans l'ensemble des expériences, elles sont générées à l'aide du modèle *Meta-Llama-3-70B*.

Les reformulations complètent le gain apporté par le GNN en élargissant la couverture lexicale côté requête. L'amélioration est particulièrement visible pour CamemBERT, tandis que Solon, déjà performant en requête simple, progresse plus modérément. Dans les deux cas, la combinaison encodage hiérarchique, propagation sur graphe et reformulations permet d'atteindre un niveau de rappel élevé.

TABLE 6 – Effet des reformulations après ajout du GNN.

	CamemBERT				Solon			
Configuration	R@5	AP@5	R@10	AP@10	R@5	AP@5	R@10	AP@10
GNN (3 couches)	0,715	0,543	0,825	0,568	0,795	0,538	0,835	0,582
Reformulations ($K = 2$)	0,819	0,568	0,876	0,608	0,819	0,554	0,876	0,625
Reformulations ($K = 3$)	0,803	0,609	0,893	0,648	0,821	0,615	0,890	0,637

Les gains restent toutefois décroissants : l'essentiel de l'amélioration est obtenu avec deux reformulations, la troisième apportant un bénéfice plus marginal. Une fois la structure intégrée par le GNN et les reformulations appliquées, l'écart initial entre CamemBERT et Solon se réduit nettement, les deux configurations atteignant des niveaux de rappel comparables. Une fois l'entraînement effectué, les représentations enrichies des articles peuvent être pré-calculées. La phase de récupération et de classement repose alors sur des opérations de similarité vectorielle classiques, ce qui permet de conserver des temps d'inférence rapides malgré l'intégration préalable du graphe.

4.4 Discussion et limites

Les résultats mettent en évidence plusieurs tendances. En configuration de base, l'encodeur spécialisé pour la similarité sémantique juridique obtient de meilleures performances que l'encodeur généraliste. L'introduction d'un encodeur hiérarchique améliore ensuite nettement la récupération. En agrégeant les représentations des segments pour construire une représentation unique de l'article, cette architecture permet d'évaluer la similarité au niveau de l'unité documentaire plutôt qu'à partir de segments isolés. Cette amélioration pourrait s'expliquer par une représentation plus stable de l'article, moins sensible aux variations locales du texte et moins dépendante d'un passage particulier.

L'ajout du GNN apporte ensuite un gain supplémentaire, plus modéré, en enrichissant ces représentations par propagation sur le graphe. Enfin, la reformulation multi-requêtes augmente systématiquement le rappel, avec des rendements décroissants lorsque le nombre de reformulations augmente. Au total, la combinaison encodage hiérarchique, propagation sur graphe et reformulation permet d'atteindre un rappel de 89,3 % en R@10 et réduit sensiblement l'écart initial entre encodeur spécialisé et encodeur généraliste. Ces résultats suggèrent que l'intégration explicite de la structure documentaire peut constituer un signal utile pour améliorer la récupération d'articles juridiques.

Plusieurs limites doivent toutefois être soulignées. D'une part, l'intégration d'un graphe du corpus interrogeable implique un entraînement spécifique des représentations enrichies. Dans un contexte juridique marqué par des évolutions fréquentes des textes, toute modification structurelle majeure peut nécessiter une mise à jour ou un réentraînement du modèle, ce qui pose la question du maintien en conditions opérationnelles. D'autre part, le jeu d'évaluation demeure de taille modeste et l'entraînement des modèles repose en partie sur des données synthétiques. Cette contrainte s'explique notamment par l'absence de jeux de données juridiques annotés à grande échelle pour des tâches de recherche d'articles dans des corpus législatifs structurés. Les performances observées ne préjugent pas entièrement de la capacité de généralisation à d'autres corpus juridiques, ni à d'autres branches du droit ou à des requêtes issues d'usages réels.

Ce travail étudie l'apport d'une modélisation explicite de la structure des corpus législatifs pour la recherche d'information juridique. Nous avons comparé une architecture dense classique à une approche intégrant un encodeur hiérarchique et une propagation sur un graphe législatif combinant relations hiérarchiques et citations. Les résultats montrent que la prise en compte explicite de la structure documentaire peut améliorer la qualité de la récupération dans des corpus juridiques structurés.

Plusieurs pistes de travail restent ouvertes. L'évaluation pourrait d'abord être étendue à des corpus juridiques plus larges et à des jeux de requêtes issus d'usages réels afin d'examiner la capacité de généralisation de l'approche. Par ailleurs, l'exploration de graphes plus riches, davantage représentatifs de la dimension dynamique du droit, pourrait permettre de mieux modéliser l'évolution normative. L'intégration de relations temporelles entre les dispositions, par exemple pour représenter les abrogations, modifications ou successions de versions, constitue à cet égard une direction naturelle. De manière plus générale, ces travaux suggèrent que l'articulation entre représentations textuelles et structures juridiques explicites pourrait participer à la conception de systèmes d'assistance à la recherche juridique plus robustes.

A Annexes

A.1 Exemples de structures de graphe

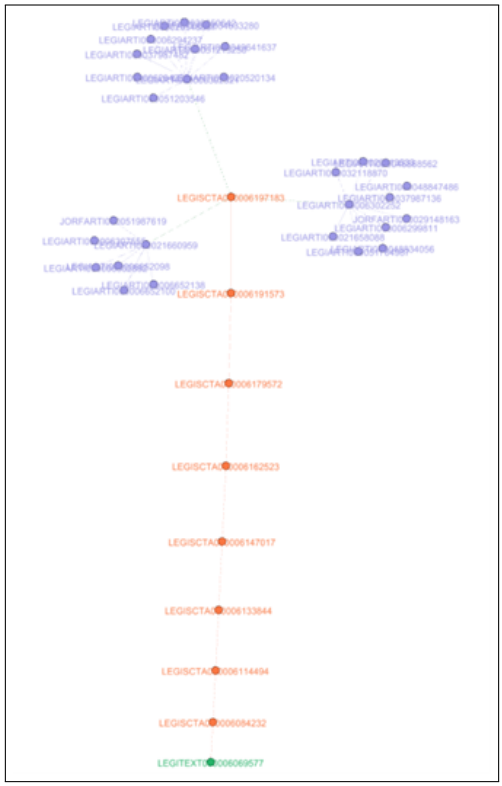

FIGURE 3 – Exemple de sous-graphe législatif extrait du corpus.

Références

ALBARÈDE L., MULHEM P., GOEURIOT L., PAPE-GARDEUX C. L., MARIÉ S. & CHARDIN-SEGUI T. (2022). Passage retrieval on structured documents using graph attention networks. In *Advances in Information Retrieval : 44th European Conference on IR Research (ECIR 2022), Proceedings, Part II*, volume 13186 de *Lecture Notes in Computer Science*, p. 13–21 : Springer. DOI : 10.1007/978-3-030-99739-7_2.

BELTAGY I., PETERS M. E. & COHAN A. (2020). Longformer : The long-document transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, p. 5713–5724 : Association for Computational Linguistics. DOI : 10.18653/v1/2020.acl-main.447.

CALLAN J. P. (1994). Passage-level evidence in document retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94)*, p. 302–310 : ACM. DOI : 10.1145/188490.188561.

HAMILTON W. L., YING R. & LESKOVEC J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, p. 1024–1034.

HARDE P., JAIN B. & JAIN S. (2025). Lecnet : A legal citation network benchmark dataset. In *Proceedings of the 1st Workshop on NLP for Empowering Justice (JUST-NLP 2025)*, p. 18–28, Mumbai, India : Association for Computational Linguistics.

HART H. (1961). *The Concept of Law*. Oxford University Press.

KARPUKHIN V., OGUZ B., MIN S., LEWIS P., WU L., EDUNOV S., CHEN D. & YIH W.-T. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 6769–6781. DOI : 10.18653/v1/2020.emnlp-main.550.

KHALID M. & VERBERNE S. (2008). Passage retrieval for question answering using sliding windows. In *Coling 2008 : Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, p. 26–33 : Coling 2008 Organizing Committee.

KIPF T. N. & WELLING M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

LOUIS A. & SPANAKIS G. (2022). A statutory article retrieval dataset in french. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, p. 6789–6803 : Association for Computational Linguistics. DOI : 10.18653/v1/2022.acl-long.468.

LOUIS A., VAN DIJCK G. & SPANAKIS G. (2023). Finding the law : Enhancing statutory article retrieval via graph neural networks. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, p. 2761–2776. DOI : 10.18653/v1/2023.eacl-main.203.

MA X., GONG Y., HE P., ZHAO H. & DUAN N. (2023). Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, p. 5303–5315. DOI : 10.18653/v1/2023.emnlp-main.322.

NGUYEN C., NGUYEN P. & NGUYEN L.-M. (2025). Retrieve–revise–refine : A novel framework for retrieval of concise entailing legal article sets. *Information Processing & Management*. DOI : 10.1016/j.ipm.2024.103949.

OORD A. V. D., LI Y. & VINYALS O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv :1807.03748*.

OST F. & VAN DE KERCHOVE M. (2002). *De la pyramide au réseau ? Pour une théorie dialectique du droit*. Publications des Facultés universitaires Saint-Louis.

PAPPAGARI R., ZELASKO P., VILLALBA J., CARMIEL Y. & DEHAK N. (2019). Hierarchical transformers for long document classification. In *Proceedings of the 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, p. 838–844 : IEEE. DOI : 10.1109/ASRU46091.2019.9003752.

PREMASIRI D., RANASINGHE T., MITKOV R., EL-HAJ M. & FROMMHOLZ I. (2025). Survey on legal information extraction : Current status and open challenges. *Knowledge and Information Systems*. DOI : 10.1007/s10115-025-02600-5.

REIMERS N. & GUREVYCH I. (2019). Sentence-bert : Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, p. 3982–3992 : Association for Computational Linguistics. DOI : 10.18653/v1/D19-1410.

ROSA G. M., RODRIGUES R. C., LOTUFO R. & NOGUEIRA R. (2021). Yes, bm25 is a strong baseline for legal case retrieval. *arXiv preprint arXiv :2104.14231*. Version 2, October 2021.

SCHLICHTKRULL M., KIPF T. N., BLOEM P., VAN DEN BERG R., TITOV I. & WELLING M. (2018). Modeling relational data with graph convolutional networks. In *The Semantic Web (ESWC 2018)*, p. 593–607 : Springer.

TANG Y., QIU R., LIU Y., LI X. & HUANG Z. (2024a). Casegnn : Graph neural networks for legal case retrieval with text-attributed graphs. In *Advances in Information Retrieval : 46th European Conference on Information Retrieval (ECIR 2024), Proceedings, Part II*, volume 14609 de *Lecture Notes in Computer Science*, p. 80–95 : Springer. DOI : 10.1007/978-3-031-56060-6_6.

TANG Y., QIU R., YIN H., LI X. & HUANG Z. (2024b). Caselink : Inductive graph learning for legal case retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)* : ACM. DOI : 10.1145/3626772.3657693.

VELIČKOVIĆ P., CUCURULL G., CASANOVA A., ROMERO A., LIÒ P. & BENGIO Y. (2018). Graph attention networks. In *International Conference on Learning Representations (ICLR)*.

WANG L., YANG N. & WEI F. (2023). Query2doc : Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, p. 9516–9529. DOI : 10.18653/v1/2023.emnlp-main.585.

YANG J., MA W., ZHANG M., ZHOU X., LIU Y. & MA S. (2022). Legalgnn : Legal information enhanced graph neural network for recommendation. *ACM Transactions on Information Systems*, **40**(2), 33 :1–33 :29. DOI : 10.1145/3469887.

ZHANG X., WEI F. & ZHOU M. (2019). Hibert : Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, p. 5059–5069 : Association for Computational Linguistics. DOI : 10.18653/v1/P19-1499.