Finding the Law: Enhancing Statutory Article Retrieval via Graph Neural Networks

Antoine Louis  Gijs van Dijck  Gerasimos Spanakis
Law & Tech Lab, Maastricht University
{a.louis, gijs.vandijck, jerry.spanakis}@maastrichtuniversity.nl

Abstract

Statutory article retrieval (SAR), the task of retrieving statute law articles relevant to a legal question, is a promising application of legal text processing. In particular, high-quality SAR systems can improve the work efficiency of legal professionals and provide basic legal assistance to citizens in need at no cost. Unlike traditional ad-hoc information retrieval, where each document is considered a complete source of information, SAR deals with texts whose full sense depends on complementary information from the topological organization of statute law. While existing works ignore these domain-specific dependencies, we propose a novel graph-augmented dense statute retriever (G-DSR) model that incorporates the structure of legislation via a graph neural network to improve dense retrieval performance. Experimental results show that our approach outperforms strong retrieval baselines on a real-world expert-annotated SAR dataset.

1 Introduction

Today, the high cost of legal expertise prevents less fortunate people from understanding and reacting to legal issues that may arise (Ponce et al., 2019). In recent years, an increasing number of works have focused on legal text processing (Zhong et al., 2020) with the intent to assist legal practitioners and citizens while reducing legal costs and improving equal access to justice for all. Statutory article retrieval (SAR), the task of retrieving statute law articles relevant to a legal question, marks the first and one of the most crucial steps in any legal aid process. Our goal is to help reduce the gap between people and the law by improving SAR systems that could provide citizens with the first component of a free professional legal aid service.

Prior work has addressed SAR with standard information retrieval approaches such as term-based models or dense embedding-based models (Kim et al., 2019; Nguyen et al., 2021). While good performance has been achieved, these approaches rely on the flawed assumption that articles are complete and independent sources of information. In reality, statute law is an ensemble of interdependent rules meticulously organized into different codes, books, titles, chapters, and sections, as illustrated in Figure 1. Each level in the structure of legislation comes with a unique heading that informs about the content of the articles below it. An article takes on its whole meaning only when considered at its rightful place in the structure with the complementary information from its neighboring articles.

This work shows that such a structure can be highly beneficial for retrieving statutes. We propose a graph-augmented dense statute retriever (G-DSR) model that incorporates the structure of legislation via a graph neural network to improve dense retrieval performance. Experimental results show that our approach outperforms strong retrieval baselines on a real-world expert-annotated SAR dataset.
DSR) model that leverages the topological structure of legislation to enhance the article content information. Specifically, the proposed model extends the document encoder of a dense retriever with a graph neural network to learn knowledge-rich cross-article representations. Similar to previous work, we adopt a contrastive learning strategy to optimize the similarity between the representations of relevant query-article pairs.

The contributions of this paper are threefold:

• We propose a graph-augmented dense retriever model for statutory article retrieval that explicitly utilizes the topological organization of statute law to enrich the article information.

• We conduct empirical evaluations on our model and demonstrate improvements over strong retrieval baselines.

• We perform ablation studies on various model components and training strategies to understand the impact of several design options on the effectiveness of our model.

2 Preliminaries

In this section, we formally introduce the task of statutory article retrieval and discuss the specific difficulties associated with it. We then explain how we identify the structure of legislation as an essential consideration in SAR.

Problem formulation. Given a simple legal question, such as "Who should pay for the construction of the common wall?", SAR aims to return one or several relevant articles from the legislation. Formally speaking, a SAR system can be expressed as a function \( R : (q, C) \rightarrow F \) that takes as input a question \( q \) along with a corpus of articles \( C = \{a_1, a_2, \ldots, a_N\} \), and returns a much smaller filter set \( F \subset C \) of the supposedly relevant articles, ranked by decreasing order of relevance. For a fixed \( k = |F| \ll |C| \), the retriever can be evaluated in isolation with multiple rank-based metrics. Most modern retriever systems follow a two-stage retrieval approach (Guo et al., 2016; Hui et al., 2017; McDonald et al., 2018), where a pre-fetcher first aims to return all relevant documents in the filter set \( F \) and a re-ranker then attempts to make more relevant documents in \( F \) appear before less relevant ones. In this work, we focus our research on improving the pre-fetcher component for SAR.

Challenges. SAR comes with two core challenges that make the task unique compared to traditional information retrieval. First, the statutes to be retrieved are written in a language that dramatically differs from the ordinary plain language used in the questions. The legal language uses a specialized jargon known for its frequent and deliberate use of formal words, Latin phrases, lengthy sentences, and expressions with flexible meanings (Charrow and Crandall, 1990). Second, statutory articles are long text sequences that may reach several thousand words. This implies overcoming the maximum input length limit of 512 tokens imposed by BERT-based models, which have recently become the standard in neural information retrieval due to their effectiveness.

Structure of legislation. The legislation comes with a well-thought-out organization of its written rules to facilitate access to provisions covering a given subject (Onoge, 2015). This organization is established in a hierarchical manner, where higher-level divisions cover broad legal domains while lower-level divisions deal with specific legal concepts. To examine the importance of this hierarchy in the SAR process, we conduct a preliminary investigation in which we study the reasoning legal experts follow when performing the task. We summarize these experts’ approach in Appendix A.1. We observe that legal experts rely heavily on the structure of law when retrieving relevant articles to a legal question, which indicates that the different divisions’ headings in the legislation carry valuable information that retrieval systems should consider. Additionally, we analyze the degree to which neighboring articles cover related subjects in Appendix A.2 and find high levels of similarities, which suggests that information from neighboring articles should be considered to capture an article’s whole meaning.

3 Approach

In this section, we present a new general approach for SAR that learns to retrieve relevant statues by using both the textual semantic information from articles and the structural graph information from the legislation. Our model, called graph-augmented dense statute retriever (G-DSR), consists of two main building blocks, as depicted in Figure 2, that are trained independently with the same objective. We first describe the dense retriever component of our approach in Section 3.1 and then explain how our legislative graph encoder builds upon it in Section 3.2.
3.1 Dense Statute Retriever

Our approach’s first component, called dense statute retriever (DSR), aims to learn high-quality low-dimensional embedding spaces for questions and articles so that relevant question-article pairs appear closer than irrelevant ones in those vector spaces. Below, we review the overall architecture of the retriever and detail the design of its query and article encoders. We then describe the contrastive learning strategy we employ and choice of negative pairs.

Bi-encoder. We use the widely adopted bi-encoder architecture (Bromley et al., 1993) as the foundation of our dense retriever. The latter maps queries and articles into dense vector representations and calculates a relevance score $s : (q, a) \mapsto \mathbb{R}_+$ between query $q$ and article $a$ by the similarity of their embeddings, i.e.,

$$s(q, a) = \sim(\theta^q, \phi^a),$$  \hspace{1cm} (1)

where $\theta^q, \phi^a \in \mathbb{R}^d$ denote the query and article embeddings respectively, and $\sim : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ is a similarity function such as cosine or dot-product.

Query encoder. To encode the queries, we feed them into a BERT-based (Devlin et al., 2019) model $E^q : \mathcal{W}^n \mapsto \mathbb{R}^d$ with weights $\theta$, that maps an input sequence of $n$ tokens from vocabulary $\mathcal{W}$ to $d$-dimensional real-valued token embeddings. We take the last layer’s [CLS] token representation as the query embedding, i.e.,

$$E^q(q) = \text{BERT}_{[CLS]}(q).$$  \hspace{1cm} (2)

Hierarchical article encoder. Since statutory articles may be longer than the maximum input length of a standard BERT-based encoder, we use a hierarchical variation that can process longer textual sequences (Pappagari et al., 2019; Zhang et al., 2019; Yang et al., 2020a). Each article $a$ is first split into smaller text passages $[p_1, p_2, \ldots, p_m]$, where a passage $p_i$ is a sequence of tokens $[t_1^{(i)}, t_2^{(i)}, \ldots, t_n^{(i)}]$ with $n \leq 512$. These passages are then independently passed through a shared BERT-based model to extract a list of context-unaware passage representations using the respective [CLS] token embeddings, as illustrated in Figure 2. Next, the hierarchical model sums the [CLS] token representations of each passage with learnable passage position embeddings and feeds the resulting representations into a small Transformer encoder to make them aware of the surrounding passages. The final article representation is computed through a pooling operation over the
context-aware passage representations, i.e.,
\[ E^\phi_A(a) = \text{pool} \left( \left[ \tilde{h}^{(1)}_{[\text{CLS}]}, \ldots, \tilde{h}^{(m)}_{[\text{CLS}]} \right] \right), \tag{3} \]
where \( \tilde{h}^{(i)}_{[\text{CLS}]} \in \mathbb{R}^d \) is the contextualized embedding of passage \( p_i \), and \( \text{pool} : \mathbb{R}^{m \times d} \mapsto \mathbb{R}^d \) is either mean or max pooling.

**Contrastive learning.** The training objective of the bi-encoder is to learn effective embedding functions \( E^Q_A(\cdot) \) and \( E^\phi_A(\cdot) \) such that relevant question-article pairs have a higher similarity than irrelevant ones. Let \( D = \{ (q_i, a^+_i) \}_{i=1}^N \) be the training data where each of the \( N \) instances consists of a query \( q_i \) associated with a relevant article \( a^+_i \). By sampling a set of negative articles \( A^- \) for each question \( q_i \), we can create a training set \( T = \{ (q_i, a^+_i, A^-) \}_{i=1}^N \). For each training instance in \( T \), we contrastively optimize the negative log-likelihood of the positive article against the negative ones, i.e.,
\[ L(q_i, a^+_i, A^-) = -\log \frac{e^{s(q_i, a^+_i)/\tau}}{\sum_{a \in A^- \cup \{a^+_i\}} e^{s(q_i, a)/\tau}}, \tag{4} \]
where \( \tau > 0 \) is a temperature parameter to be set.

**Negatives.** We consider two types of negative examples: (i) in-batch (Chen et al., 2017; Henderson et al., 2017), i.e., articles paired with the other questions from the same mini-batch, and (ii) BM25, i.e., top articles returned by BM25 that are not relevant to the question.

### 3.2 Legislative Graph Encoder

Our approach’s second component, called Legislative Graph Encoder (LGE), aims to enrich article representations given by the trained retriever’s article encoder by fusing information from a legislative graph. Below, we elaborate on the legislative graph construction and the graph training process.

**Graph construction.** To leverage the hierarchical organization of statute law, we formalize the latter as a tree structure consisting of two types of node: (i) section nodes, which are titled structural units that represent the consecutive divisions in codes of law (i.e., the headings of the books, titles, chapters, and sections), and (ii) article nodes, which are textual content units that represent the different statutory articles. As illustrated in Figure 1, the edges represent the hierarchical connections between section and article nodes. Formally, such a tree can be represented as a directed acyclic graph \( G = (\mathcal{V}, \mathcal{E}) \), with \( \mathcal{V} \) as the node set and \( \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \) as the edge set.

**Node feature initialization.** Nodes in \( \mathcal{V} \) are commonly associated with \( d \)-dimensional features. We apply the article encoder \( E^\phi_A(\cdot) \) from the trained bi-encoder to encode the semantic information of nodes (i.e., section headings and article contents) offline and use the resulting embeddings as the initial node features \( X \in \mathbb{R}^{|\mathcal{V}| \times d} \).

**Node feature update.** To fuse the information of node features using the graph structure, we use a graph neural network (GNN). Such a model consists of a stack of neural network layers, where each layer aggregates local neighborhood information (i.e., features of neighbors) around each node and then passes this aggregated information on to the next layer. Generally speaking, a GNN takes as inputs the feature matrix \( X \) and the graph’s adjacency matrix \( A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|} \), with \( A_{i,j} \) as the edge weight between nodes \( i \) and \( j \), and produces a node-level output \( Z \in \mathbb{R}^{|\mathcal{V}| \times d} \) that captures each node’s structural properties. Every GNN layer can be written as a non-linear function
\[ H^{(l+1)} = f(H^{(l)}, A), \tag{5} \]
with \( H^0 = X \) and \( H^L = Z \), \( L \) being the number of layers. In its simplest form, the layer-wise propagation rule is such that
\[ f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}), \tag{6} \]
where \( W^{(l)} \) is the input linear transformation’s weight matrix for the \( l \)-th neural network layer and \( \sigma(\cdot) \) is a non-linear activation function. We propose to use a 3-layer GATv2 network (Brody et al., 2022), a variant of GAT (Velickovic et al., 2018) that has the ability to learn the strength of connection between neighboring nodes through a dynamic attention mechanism. Formally, a GATv2 layer updates a node’s hidden state as follows
\[ h_i^{(l+1)} = \sigma \left( \sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} W^{(l)} h_j^{(l)} \right), \tag{7} \]
where \( \mathcal{N}(i) \) is the set of first-order neighbors of node \( i \), and \( \alpha_{ij}^{(l)} \) are normalized attention coefficients indicating the importance of node \( j \)’s features to \( i \) in the \( l \)-th layer. The latter are computed based on the features of the connected nodes using an attention function \( \text{att} : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R} \) such that
\[ \alpha_{ij}^{(l)} = \text{softmax} \left( \sigma(\text{att}(h_i^{(l)}, h_j^{(l)})) \right). \tag{8} \]
Learning process. To optimize the GNN parameters, we adopt the same contrastive learning strategy used to train the bi-encoder. Since graph $G$ can be relatively large, performing an update of all the node features in $G$ at every training iteration would incur high computational costs. Besides, most of these computations would be of no use as only the updated representations of nodes $\{A_i^+ \cup a_i^-\}_{i=1}^{|B|}$ from batch $B$ are needed to update the model parameters. Therefore, we build a subgraph $G^{\text{sub}}$ at each training step that only contains the article nodes from batch $B$ as well as their $L$-hop neighbors (where $L$ is decided by the number of GNN layers). We then pass that sub-graph to the graph network and use the resulting article representations to compute the loss in Equation (4). Comparably to the node features, the query embeddings are pre-computed offline before training by the query encoder $E_{\theta}^Q(\cdot)$ of our trained bi-encoder.

4 Experimental Setup

In this section, we present the basic setup for experiments. In particular, Section 4.1 describes the dataset we conduct our experiments on, Section 4.2 details our model implementation, Section 4.3 reviews the different baselines we use for comparison, and Section 4.4 reports the evaluation metrics.

4.1 Dataset

We conduct experiments on the publicly available Belgian Statutory Article Retrieval Dataset (Louis and Spanakis, 2022, BSARD). To the best of our knowledge, BSARD is the only SAR dataset that provides the lists of consecutive division headings each article belongs to, which is crucial for building the graph of the legislative structure. The dataset consists of 1,100+ French native questions on various legal topics, as shown in Table 1, labeled by skilled experts with references to relevant statutory articles from the Belgian legislation. The retrieval corpus comprises 22,600+ articles collected from 32 Belgian codes covering numerous legal domains. The questions are relatively short and might have several relevant legal articles. We refer readers to the original paper for further data collection and analysis details.

4.2 Implementation Details

Model. We use the publicly released CamemBERT (Martin et al., 2020) checkpoint to initialize DSR’s query encoder. Due to the specificity of the legal language the article encoder has to deal with, we follow prior work on domain adaptation (Gururangan et al., 2020; Jørgensen et al., 2021) and continue pre-training CamemBERT on BSARD statutory articles for 50k gradient steps to adapt it to the target legal domain. We use the resulting domain-specific checkpoint to warm-start the article’s first-level encoder. The second-level encoder is a two-layer Transformer encoder of 14M parameters with a similar configuration (i.e., 768-hidden, 3072-intermediate, 12-heads, 0.1 dropout, GeLU). We use max-pooling to aggregate the final chunk representations and cosine as the decomposable similarity function.

Data augmentation. Due to the recent success in using synthetic query generation to improve dense retrieval performance (Liang et al., 2020; Ma et al., 2021; Thakur et al., 2021), we propose to augment BSARD with synthetic domain-targeted queries. We use a mT5 model (Raffel et al., 2020) fine-tuned on general domain data from mMARCO (Bonifacio et al., 2021) to synthesize queries for our target statutory articles. We generate five queries per article, which results in a total of around 118k synthetic queries. We combine the latter with the gold BSARD train samples and obtain an augmented training set of around 122.5k question-article pairs.

Optimization. We train DSR for 15 epochs with a batch size of 24 using AdamW (Loshchilov and Hutter, 2017) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-7$, weight decay of 0.01, and learning rate warm up along the first 5% of the training steps to a maximum value of 2e-5, after which linear decay is applied. We then optimize LGE parameters for 10 training steps.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family</td>
<td>216</td>
<td>56</td>
<td>67</td>
</tr>
<tr>
<td>Housing</td>
<td>203</td>
<td>38</td>
<td>66</td>
</tr>
<tr>
<td>Money</td>
<td>103</td>
<td>35</td>
<td>36</td>
</tr>
<tr>
<td>Justice</td>
<td>96</td>
<td>25</td>
<td>30</td>
</tr>
<tr>
<td>Foreigners</td>
<td>41</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>Social security</td>
<td>27</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Work</td>
<td>23</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>709</td>
<td>177</td>
<td>222</td>
</tr>
</tbody>
</table>

Table 1: Topic distribution of questions in BSARD.
epochs with a batch size of 512 using AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-7$, weight decay of 0.1, and a constant learning rate of $2e-4$. We use 16-bit automatic mixed precision to accelerate training and save memory. Details on our hyperparameter tuning process are given in Appendix B.

**Hardware & schedule.** Training is performed on a single 32 GB NVIDIA V100 GPU hosted on a server with a dual 20-core Intel Xeon E5-2698 v4 CPU @2.20GHz and 512 GB of RAM. It takes around 1 day to train DSR and 35 minutes for LGE.

**Libraries.** We implement, train, and tune our models using Transformers (Wolf et al., 2020), PyTorch (Paszke et al., 2019), PyTorch-Geometric (Fey and Lenssen, 2019), PyTorch-Lightning (Falcon, 2019), W&B Sweeps (Biewald, 2020), and DeepSpeed (Rasley et al., 2020).

### 4.3 Baselines
We compare our approach against three strong retrieval systems. As a sparse baseline model, we follow prior work and consider BM25 (Robertson et al., 1994), a popular bag-of-words retrieval function based on exact term matching. We then examine the document expansion technique docT5query (Nogueira and Lin, 2019), which augments each article with a pre-defined number of synthetic queries generated by a finetuned mT5 model, and then uses a traditional BM25 lexical index from the augmented articles for retrieval. Last, we include the results of a supervised dense passage retriever (Karpukhin et al., 2020, DPR) pre-finetuned on more than 90.5k question-context pairs from a combination of three French QA datasets.

### 4.4 Evaluation
We evaluate model performance using three commonly used ranking measures (Manning et al., 2008), namely the macro-averaged recall at different cutoffs (R@k), mean average precision (mAP), and mean r-precision (mRP). Those metrics are further defined in Appendix D. We deliberately omit to report the precision@k given that questions in BSARD have a variable number of relevant articles, which implies that questions with r relevant articles would always have P@k < 1 if k > r. Similarly, the mean reciprocal rank (mRR) is not appropriate for BSARD as only the first relevant article would be considered. As some questions might have up to 100 relevant articles, we use $k \in \{100, 200, 500\}$ for the recall@k.

## 5 Experiments
In this section, we empirically evaluate the effectiveness of our proposed approach against competitive baselines and discuss the main results in Section 5.1. Next, we provide an ablation study in Section 5.2 to understand how different design and training options affect our model’s performance.

### 5.1 Main Results
Table 2 shows retrieval performance on BSARD test set. Although we report model performance on two rank-aware metrics (i.e., mAP and mRP), we emphasize that our approach is specifically aimed at improving the pre-fetching component of a retriever (Zhang et al., 2021a) and therefore focuses on optimizing rank-unaware metrics (i.e., R@k).

First, we compare the performance of our proposed G-DSR model against other well-known retrieval approaches and find it significantly outperforms all of them on SAR. In particular, it improves over the sparse retrieval methods by around 30% on recall@k and by more than 25% on mAP and mRP. It also outperforms a competitive pre-finetuned DPR model by 6% on R@100, 9% on R@200, and 5% on R@500. However, the latter shows a better performance on rank-aware metrics compared to our DSR models, which we speculate might be due to its extensive pre-finetuning step on three domain-general retrieval datasets, leading the model to a deeper knowledge of the task at hand.

Next, we investigate the influence of different training strategies on the rank-unaware results of our base dense retriever. We find that DSR’s performance is improved when adapting the article text encoder to the legal domain before finetuning on the target data. Besides, training DSR on a larger dataset containing synthetic domain-targeted queries improves its performance even more.

Finally, our results show that using a GNN model on top of DSR allows to enrich the article representations and leads to the best overall performance. Interestingly, G-DSR also significantly improves the rank-aware performance of our best performing DSR model by $\sim 12\%$, suggesting that a GNN could act as an effective re-ranker for SAR.
5.2 Ablation Study

To further understand how different design choices and training strategies affect the results, we conduct several additional experiments and discuss our findings below.

Alternative pre-trained LMs. In addition to CamemBERT, we experiment with several other French or multilingual pre-trained language models to initialize the first-level text encoders in DSR – namely, mBERT (Devlin et al., 2019), XLM-R (Conneau et al., 2020), and ELECTRA-fr (Clark et al., 2020).6 We fine-tune the different warm-started models on BSARD training set and report dev results in Table 3. We find that a CamemBERT-initialized DSR model performs best.

Alternative GNNs. Additionally to GATv2, we explore different GNN architectures to perform the node feature update – namely, GCN (Kipf and Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Velickovic et al., 2018), and k-GNN (Morris et al., 2019) – and summarize the results in Table 4. Our experiments show that using an alternative GNN model does not affect performance much, which suggests that the act of fusing information from neighboring nodes is more important than the way the aggregation is performed.

Similarity and loss functions. Besides cosine for scoring pairs of query-article representations, we also experiment with dot-product and Euclidean distance and find both inferior to cosine. As an alternative to negative log-likelihood, we test the triplet loss (Burges et al., 2005) and observe that the latter significantly decreases model performance. More details can be found in Appendix E.

Table 2: Retrieval performance on BSARD Test set. The best results are marked in bold.

<table>
<thead>
<tr>
<th>Model</th>
<th>#Params</th>
<th>R@100 (↑)</th>
<th>R@200 (↑)</th>
<th>R@500 (↑)</th>
<th>mAP (↑)</th>
<th>mRP (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baselines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 BM25</td>
<td>-</td>
<td>49.3</td>
<td>57.3</td>
<td>63.0</td>
<td>16.8</td>
<td>13.6</td>
</tr>
<tr>
<td>2 docT5query</td>
<td>-</td>
<td>51.7</td>
<td>59.4</td>
<td>65.8</td>
<td>18.7</td>
<td>15.0</td>
</tr>
<tr>
<td>4 DPR</td>
<td>220M</td>
<td>77.9</td>
<td>81.3</td>
<td>88.2</td>
<td>45.4</td>
<td>39.1</td>
</tr>
</tbody>
</table>

Our model: DSR

5 DSR                        | 234M    | 71.7      | 81.8      | 86.7      | 35.6    | 28.8    |
6 DSR w. domain-adaptive pre-training | 234M    | 79.8      | 83.9      | 88.9      | 39.5    | 31.3    |
7 DSR w. data augmentation    | 234M    | 82.7      | 88.7      | 92.8      | 35.3    | 27.5    |
8 G-DSR                       | 262M    | **84.3**  | **90.4**  | **93.1**  | **47.1**| **40.2**|

Table 3: BSARD Dev results of DSR warm-started with different pre-trained word embedding models.

<table>
<thead>
<tr>
<th>Pre-trained LM</th>
<th>#Params</th>
<th>R@100 (↑)</th>
<th>R@500 (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLM-R</td>
<td>278M</td>
<td>59.8</td>
<td>78.7</td>
</tr>
<tr>
<td>mBERT</td>
<td>177M</td>
<td>69.2</td>
<td>86.6</td>
</tr>
<tr>
<td>ELECTRA-fr</td>
<td>110M</td>
<td>57.6</td>
<td>73.7</td>
</tr>
<tr>
<td>CamemBERT</td>
<td>110M</td>
<td>75.4</td>
<td>88.5</td>
</tr>
</tbody>
</table>

Table 4: BSARD Dev results of LGE with different (3-layer) GNN architectures.

6 Related Work

Our work operates at the intersection of several research areas, including long document modeling, dense information retrieval, graph neural networks, and legal NLP.

Long document modeling. The emergence of deep neural networks for language processing brought new challenges to text encoding, one of which is learning high-quality representations of long documents. For example, Tang et al. (2015) employ a bottom-up approach using CNN and BiLSTM-based hierarchical networks, where sentences are first encoded into vectors, which are then combined to form a single document vector. Similarly, Yang et al. (2016) build a document vector by aggregating important words into sentence vectors and then aggregating important sentence vectors to document vectors using attention mecha-
Dense information retrieval. Traditionally, lexical approaches such as TF-IDF and BM25 (Robertson et al., 1994) have been the de facto standard for textual information retrieval due to their robustness and efficiency. However, these approaches suffer from the lexical gap problem (Berger et al., 2000) and can only retrieve documents containing keywords present in the query. To overcome this limitation, recent work relies on neural-based architectures to capture semantic relationships between pairs of texts (Lee et al., 2019; Karpukhin et al., 2020; Yang et al., 2020b; Xiong et al., 2021). These models map queries and documents into dense vector representations and calculate a relevance score by the similarity of the vectors (Gillick et al., 2018), which allows the document representations to be pre-computed and indexed offline for inference. The dense retrieval approach was recently extended by hybrid lexical-dense methods, which aim to combine the strengths of both approaches (Seo et al., 2019; Gao et al., 2021; Luan et al., 2021). We refer the readers to Yates et al. (2021) for a survey on neural information retrieval.

Graph neural networks. Graph neural networks (GNNs) capture the topological relationships among the nodes of a graph using an information diffusion mechanism that propagates node features according to the underlying graph-structured data (Scarselli et al., 2009). These models have shown their effectiveness and flexibility in a wide variety of NLP tasks, including text classification (Lin et al., 2021; Yu et al., 2022), relation extraction (Zhang et al., 2018; Li et al., 2020; Carbonell et al., 2020), and question answering (Cao et al., 2019; Sun et al., 2021). Recently, GNNs have been employed for document retrieval to enhance the vector representations by leveraging the topological structure of the documents, where nodes are passages from a document and edges are relations between these passages (Xu et al., 2021a; Zhang et al., 2021b; Albarede et al., 2022).

Application to the legal domain. In recent years, the legal domain has attracted much interest in the NLP community, both for its challenging characteristics and massive volumes of textual data (Chalkidis and Kampas, 2019; Zhong et al., 2020). Researchers see it as an opportunity to develop novel automated methodologies that can reduce heavy and redundant tasks for legal professionals while providing a reliable, affordable form of legal support for laypeople (Bommasani et al., 2021). Earlier techniques for legal information retrieval were mainly based on term-matching approaches (Kim and Goebel, 2017; Tran et al., 2018). Recently, a growing number of works have used neural networks to enhance retrieval performance, including word embedding models (Landthaler et al., 2016), doc2vec models (Sugathadasa et al., 2018), CNN-based models (Tran et al., 2019), and BERT-based models (Nguyen et al., 2021; Chalkidis et al., 2021; Althammer et al., 2022). To the best of our knowledge, we are the first to exploit the structure of statute law with GNNs to improve the performance of dense retrieval models.

7 Conclusion

In this paper, we introduce G-DSR, a novel approach for statutory article retrieval (SAR) that leverages the topological structure of legislation to improve retrieval performance. Specifically, G-DSR enriches the article representations of a dense retriever designed for long document retrieval by employing a graph neural network that uses the organization of statute law to learn knowledge-rich cross-article embeddings. Experiments show that G-DSR outperforms competitive baselines on a real-world expert-annotated SAR dataset. We also include a detailed analysis to motivate our design choices and training strategies.
Limitations

While our approach performs well on statutory article retrieval, it comes with several limitations that provide avenues for future work.

First, experimental results are based on questions and labels drafted by legal professionals. It is possible that other legal professionals would draft the questions differently or, less likely yet possible, that they would deem different statutory provisions relevant. This raises the question of to what extent similar results would be obtained if the model were trained on a different dataset, for instance, based on other experts or domains, hence testing the approach’s generalizability. The main challenge in this regard is obtaining data, as organizations are unlikely to share or even collect similar data.

Second, our proposed methodology was evaluated exclusively on the Belgian legislation, whose laws are organized in a hierarchical manner where the deeper the divisions, the more closely related the legal concepts of the articles under them. Although we believe our approach could be applied to most, if not all, jurisdictions that rely on statute law (including both civil and common law countries), different jurisdictions may have different organizations of their legal provisions, which could potentially affect the model’s performance. It is also worth mentioning that the dataset used for evaluation comes with a linguistic bias as Belgium is a multilingual country with French, Dutch, and German speakers, but the provided provisions are only available in French. Studying the applicability and impact of the present work to other jurisdictions and languages is an exciting research direction that is challenging in practice due to the scarcity of high-quality multilingual statute retrieval datasets.

Then, our approach currently considers the topological structure of legislation for modeling the inter-article dependencies, which implies that information is aggregated between direct neighboring articles only while those from more distant sections are completely ignored. Nevertheless, it is common for articles to cite other articles from different sections or even different statutes. Therefore, we believe that considering richer legal graph structures, especially legal citation networks, could increase effectiveness even more. However, building such citation networks from raw texts requires a considerable text-processing effort.

Finally, although G-DSR shows promise for statutory article retrieval, it is not yet ready for practical use in the real world. One issue is that our model is designed to be an effective pre-fetcher, optimizing recall such that all articles relevant to a question appear in an unordered filter set of size $k$ ($k$ being relatively large). However, in practice, users would expect a high-quality retrieval system to not only find these relevant articles but also to sort them by decreasing order of importance, requiring an adequate re-ranker. Then, it is essential to recognize that while access to relevant legal provisions is a necessary step in helping the general public solve their legal issues, it is not a sufficient condition on its own as laypeople may still struggle to understand the legal jargon and apply the provisions to their specific situations. Ideally, the tool to be made accessible to the public should consist of a two-stage framework: (i) a legal provision retriever, which selects a small subset of relevant legal articles in response to a given question, and (ii) a legal-to-natural translator or summarizer, which examines the retrieved articles and generates an answer in natural language. In the present work, we chose to focus on the first stage of this framework and leave the second for future work.

Ethics Statement

The scope of this work is to provide a new methodology along with extensive experiments to drive research forward in statutory article retrieval. We believe the latter is an important application field where more research should be conducted to improve legal aid services and access to justice for all. We do not foresee situations where the use of our methodology would lead to harm (Tsarapatsanis and Aletras, 2021). Nevertheless, although our goal is to improve the understanding of the law by those who suffer from legal information asymmetry, it cannot be excluded that the technology presented here could exacerbate inequality if states, companies, or lawyers benefit more from its use than the intended beneficiaries (i.e., citizens, consumers, or employees).

Acknowledgments

This research is partially supported by the Sector Plan Digital Legal Studies of the Dutch Ministry of Education, Culture, and Science. In addition, this research was made possible, in part, using the Data Science Research Infrastructure (DSRI) hosted at Maastricht University.
References


Veda R Charrow and Jo Ann Crandall. 1990. Legal language: What is it and what can we do about it?.  


Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3505-3506. [Page 6]


To better understand the reasoning skilled humans follow when retrieving statutory articles, we ask several legal experts to retrieve relevant articles to questions sampled from the Belgian Statutory Article Retrieval Dataset (Louis and Spanakis, 2022, BSARD). We deliberately choose experts unfamiliar with Belgian law and thus have no past knowledge of the location of articles covering a particular subject. In what follows, we summarize the approach these experts use.

First, they determine whether the issue involves either public or private law. To distinguish between the two, it is necessary to identify to whom the rules apply (i.e., the parties involved in the issue that hold the rights or duties). Generally speaking, public law deals with issues that affect the general public or state (i.e., society as a whole), whereas private law deals with issues that affect individuals, families, and businesses. This first step allows the experts to make an initial selection among the codes of law, which generally relate to only one of either public or private law.

Next, the experts refine their search by determining the field of law (e.g., contract law), followed by the sub-field (e.g., tenant law), and so on, until a set of potentially relevant codes is created in the question’s domains. The experts then focus on the table of contents of one of the selected codes and undertake a hierarchical search that starts from the book’s headings and progressively extends to its titles, chapters, and sections. This step makes it possible to filter out many irrelevant articles by

**Appendix**

**A Preliminary Studies**

A.1 How important is the structure of law for statutory article retrieval?

To better understand the reasoning skilled humans follow when retrieving statutory articles, we ask several legal experts to retrieve relevant articles to questions sampled from the Belgian Statutory Article Retrieval Dataset (Louis and Spanakis, 2022, BSARD). We deliberately choose experts unfamiliar with Belgian law and thus have no past knowledge of the location of articles covering a particular subject. In what follows, we summarize the approach these experts use.

First, they determine whether the issue involves either public or private law. To distinguish between the two, it is necessary to identify to whom the rules apply (i.e., the parties involved in the issue that hold the rights or duties). Generally speaking, public law deals with issues that affect the general public or state (i.e., society as a whole), whereas private law deals with issues that affect individuals, families, and businesses. This first step allows the experts to make an initial selection among the codes of law, which generally relate to only one of either public or private law.

Next, the experts refine their search by determining the field of law (e.g., contract law), followed by the sub-field (e.g., tenant law), and so on, until a set of potentially relevant codes is created in the question’s domains. The experts then focus on the table of contents of one of the selected codes and undertake a hierarchical search that starts from the book’s headings and progressively extends to its titles, chapters, and sections. This step makes it possible to filter out many irrelevant articles by

**References**

analyzing the connection between the question’s subject and the different sections’ headings.

Finally, the experts explore the articles within the sections deemed potentially relevant to the question in search of the expected answer. If the experts realize that the chosen direction is a dead end, they return to the previous higher level of the structure, choose another potentially relevant direction, and narrow their search from there.

From this study, we conclude that legal experts rely heavily on the structure of law when retrieving relevant articles to a legal question, which indicates that the different divisions’ headings carry valuable information that retrieval systems should consider.

A.2 How related are neighboring articles in statute law?

In statute law, the sense of a given article is not necessarily self-contained by itself but instead spans across different articles from the same or even different sections. To confirm this, we study to what extent consecutive articles (as they appear in the statute books) address similar subjects.

We consider the Belgian Civil Code, which is the book whose articles are most cited in BSARD, and randomly sample sets of 200 consecutive articles out of it. We then normalize the articles by lowercasing, lemmatizing, and removing stop-words, punctuation, and numbers. Finally, we compute the cosine similarities between the TF-IDF representations of all articles from a given set. Figure 3 shows a heatmap of article similarities for such a set. We see that consecutive articles do indeed cover similar topics, suggesting that the information in a given article is likely to be complementary to that in its neighboring articles. Therefore, we assume that neighboring articles should be considered to capture an article’s whole meaning.

B  G-DSR Hyperparameter Tuning

We conduct hyperparameter tuning using Bayes search based on performance on BSARD development set, measured with the macro-averaged R@200. Due to limited computational resources, we train our models on BSARD training set only – which takes approximately 1 hour and 15 minutes for DSR and around 5 minutes for LGE – and use the constrained search spaces described below.

**DSR grid search space:**
- batch size: {8, 16, 24, 32}
- learning rate: {5e-5, 4e-5, 3e-5, 2e-5, 1e-5}
- weight decay: {0, 0.1, 0.01, 0.001}
- max chunk length: {64, 128, 256, 512}
- max document length: {1024, 2048}
- pooling strategy: {mean, max}
- similarity function: {dot-product, cosine, L2}
- temperature: {0, 0.1, 0.01, 0.001}

**LGE grid search space:**
- batch size: {8, 16, 32, 64, 128, 256, 512}
- learning rate: {2e-2, 2e-3, 2e-4, 2e-5, 2e-6}
- weight decay: {0, 0.1, 0.01, 0.001}
- #layers: {1, 2, 3, 4}

In total, we run 100 hyperparameter search trials for both DSR and LGE. The optimal hyperparameters, shown in Table 5, are used to re-train the models combining both train and development sets for a final evaluation on the test set.

C  BM25 Hyperparameter Tuning

Following Chalkidis et al. (2021), who show that BM25 performance is highly dependent on adequately choosing the \((k_1, b)\) values for the task at hand, we perform a hyperparameter grid search on BSARD development set and plot the results in Figure 4. We observe that, in the case of SAR, the best performance is obtained with \(k_1 = 2.5\) and \(b = 0.2\). Therefore, we use these values for the final evaluation on BSARD test set.


**D Evaluation Metrics**

Let \( \text{rel}_q(a) \in \{0, 1\} \) be the binary relevance label of article \( a \) for question \( q \), and \((i, a) \in \mathcal{F}_q \) a result tuple (article \( a \) at rank \( i \)) from the filter set \( \mathcal{F}_q \subset \mathcal{C} \) of ranked articles retrieved for question \( q \).

**Recall.** The recalling is the fraction of relevant articles retrieved for query \( q \) w.r.t. the total number of relevant articles in the corpus \( \mathcal{C} \), i.e.,

\[
R_q = \frac{\sum_{(i, a) \in \mathcal{F}_q} \text{rel}_q(a)}{\sum_{a \in \mathcal{C}} \text{rel}_q(a)}. \tag{9}
\]

When computed for a filter set of size \( k = |\mathcal{F}_q| \ll |\mathcal{C}| \), i.e., at a certain cutoff and not on the entire list of articles in \( \mathcal{C} \), we report the metrics with the suffix “@k”.

**R-Precision** The R-Precision is the proportion of the top-\( R \) retrieved articles that are relevant to query \( q \), where \( R \) is the total number of relevant articles for \( q \), i.e.,

\[
\text{RP}_q = \frac{\sum_{(i, a) \in \mathcal{F}_q} \text{rel}_q(a)}{R}. \tag{10}
\]

**Average Precision.** The average precision is the mean of the precision value obtained after each relevant article is retrieved, that is

\[
\text{AP}_q = \frac{\sum_{(i, a) \in \mathcal{F}_q} P_{q,i} \times \text{rel}_q(a)}{\sum_{a \in \mathcal{C}} \text{rel}_q(a)}, \tag{11}
\]

where \( P_{q,j} \) is the precision computed at rank \( j \) for query \( q \), i.e., the fraction of relevant articles retrieved for query \( q \) w.r.t. the total number of articles in the retrieved set \( \{\mathcal{F}_q\}_{i=1}^{|\mathcal{F}_q|} \).

\[
P_{q,j} = \frac{\sum_{(i, a) \in \mathcal{F}_q} \text{rel}_q(a)}{|\mathcal{F}_q|}. \tag{12}
\]

We report the macro-averaged recall at various cutoffs (R@k), mean Average Precision (mAP), and mean R-Precision (mRP), which are the average values over a set of \( n \) queries.

**E Ablation Details**

Besides cosine similarity and negative log-likelihood (NLL) loss, we also test the dot-product and Euclidean (inverse of distance is taken as similarity measure) as well as the triplet loss. The temperature for the NLL loss is set to 0.01, and the margin value of the triplet loss is set to 1. We report the results on BSARD development set in Table 6. For a fair comparison, all models are trained for 15 epochs with a batch size of 24, weight decay of 0.01, warm-up proportion of 0.05, an initial learning rate of 2e-5, and a linear decay learning rate schedule.

### Table 5: Training hyperparameters for DSR and LGE.

<table>
<thead>
<tr>
<th>Hyperparam</th>
<th>DSR</th>
<th>LGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Chunk Length</td>
<td>128</td>
<td>-</td>
</tr>
<tr>
<td>Maximum Document Length</td>
<td>1024</td>
<td>-</td>
</tr>
<tr>
<td>Pooling Strategy</td>
<td>max</td>
<td>-</td>
</tr>
<tr>
<td>#Layers</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Loss</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Similarity</td>
<td>cos</td>
<td>cos</td>
</tr>
<tr>
<td>Training</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Batch Size</td>
<td>24</td>
<td>512</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>0.01</td>
<td>0.1</td>
</tr>
<tr>
<td>Max Epochs</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>Peak Learning Rate</td>
<td>2e-5</td>
<td>2e-4</td>
</tr>
<tr>
<td>Learning Rate Decay</td>
<td>Linear</td>
<td>Constant</td>
</tr>
<tr>
<td>Warmup ratio</td>
<td>0.05</td>
<td>0.0</td>
</tr>
<tr>
<td>Adam(\epsilon)</td>
<td>1e-7</td>
<td>1e-7</td>
</tr>
<tr>
<td>Adam(\beta_1)</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>Adam(\beta_2)</td>
<td>0.999</td>
<td>0.999</td>
</tr>
<tr>
<td>Gradient Clipping</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

### Table 6: BSARD Dev results of DSR trained using different similarity and loss functions.

<table>
<thead>
<tr>
<th>Loss</th>
<th>Similarity</th>
<th>R@100 (↑)</th>
<th>R@200 (↑)</th>
<th>R@500 (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-entropy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cosine</td>
<td>75.0</td>
<td>80.7</td>
<td>86.7</td>
<td></td>
</tr>
<tr>
<td>Dot</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Euclidean</td>
<td>43.9</td>
<td>58.1</td>
<td>71.6</td>
<td></td>
</tr>
<tr>
<td>Triplet</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cosine</td>
<td>5.4</td>
<td>9.9</td>
<td>17.2</td>
<td></td>
</tr>
<tr>
<td>Dot</td>
<td>4.3</td>
<td>6.4</td>
<td>10.3</td>
<td></td>
</tr>
<tr>
<td>Euclidean</td>
<td>9.0</td>
<td>14.3</td>
<td>25.2</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4: Heatmap showing BM25 results on BSARD Dev set for different values of \( k_1 \) and \( b \).