Enhancing Document Retrieval in Large Corpora: A Keyphrase and Reference-Based Approach

Zoltán Szoplák[0000−0003−1823−0536], Dávid Varga[0000−0002−3176−8106], Peter Gurský[0000−0002−4744−7390], Šimon Horvát[0000−0002−3191−8469], and Stanislav Krajči[0000−0001−5612−3534]

Institute of Computer Science, Pavol Jozef Šafárik University in Košice, Slovakia

Abstract. The task of document retrieval, especially in large corpora, remains a challenge despite the recent advances in NLP and LLM technology. The context windows for such models are often not sufficient enough to fit a large number of documents inside them. To handle such issues, many NLP tasks, such as document retrieval for RAGs, perform a two-step process of filtering out the most relevant documents and then using those as input for the LLM to rank them. However, the larger the corpus, the greater the need for a robust document retrieval method. We propose a solution that augments the standard embedding similarity retrieval method with a score based on matching extracted keyphrases and a further extension using references in legal documents. The former reached a mean average precision of 80.25% while the latter improved on the former's result even further, achieving a mAP score of 92.85%.

Keywords: document retrieval · keyphrases · legal references · information retrieval

1 Introduction

Document retrieval, especially in large text corpora, has been a significant problem in machine learning and natural language processing, especially in the recently popular field of Retrieval Augmented Generation (RAG). Despite the recent advances in the field of NLP and LLMs, retrieval tasks remain an issue due to the limited context window sizes of most LLM models. And doing it iteratively through an entire corpus is currently too costly to be feasible. Therefore, document retrieval systems usually operate in two stages. The first consists of selecting the most relevant set of documents based on embedding similarity which can then fit inside the context window of an LLM that would perform a reranking step. However, such approaches pose their own challenges. In the case of very large datasets, the embedding-based search may return many matches, making the reranking part of such algorithms computationally expensive. In the same vein, limiting the number of retrieved documents may cause relevant documents to be missed from the reranking step of the process. The size of the text can also cause major problems regarding such approaches. In extensive texts without a proper chunking algorithm, the uniqueness of a given text within a corpus can

easily be lost, especially when it comes to texts with a semi-formulaic structure, such as legal texts and judicial decisions. And while it is entirely possible to split the given document into chunks to avoid this problem, however chunking algorithms have their own challenges.

2 Related Work

Legal document retrieval is a specialized area of information retrieval (IR) that has evolved through advancements in natural language processing (NLP) and machine learning (ML). Key challenges include handling the complexity and length of legal texts, where paragraph-level retrieval [6] and embedding-based similarity models [2] improve access to relevant case law. Deep learning approaches, such as neural classifiers for legal opinions [7] and BERT-based case retrieval [1], further enhance retrieval accuracy by capturing nuanced relationships in legal texts.

Relevance feedback mechanisms play a critical role in refining retrieval results [9]. Pipitone and Alami [4] introduce LegalBench-RAG, a benchmark designed to assess retrieval performance in Retrieval-Augmented Generation (RAG) systems for legal texts, demonstrating the importance of precise document mapping and chunking strategies. Similarly, Wiratunga et al. [10] propose CBR-RAG, which incorporates Case-Based Reasoning (CBR) into RAG to improve retrieval by leveraging case indexing and similarity matching, reducing hallucinations in legal LLMs.

These advancements highlight the growing importance of integrating NLP, deep learning, and retrieval-specific enhancements to improve legal information systems. By combining effective chunking, domain-specific embeddings, and retrieval frameworks, modern systems achieve greater accuracy in retrieving relevant legal texts for case law analysis and question answering.

3 Methods

3.1 Overview

Before we dive into the details of our methods for retrieving legal documents, let us present the core parts of the algorithm. We can divide them into preprocessing and retrieval parts. Although some steps are marked as optional, they are quite beneficial for a quality result as will be discussed in section 4.2. Formal labels used in the outputs of the steps are defined in the following sections.

1. Preprocessing
   (a) Semantical chunking of all judicial decisions - output: list of chunks for each document.
   (b) Creating chunk embeddings via language model - output: vector embedding for each chunk $v(c_i)$
   (c) Extraction of keyphrases from chunks - output: top t keyphrases for each chunk - $KPP(c_i)$

i. Computing modified TF-IDF score for each keyphrase.
ii. Using Self-Attention MAPS (AttentionSeek) to create SAM score for each keyphrase.
iii. Combining the two metrics, extracting top t keyphrases from a chunk.

(d) Extraction of references from document chunks (optional) - output: list of referenced legal regulations and judicial decisions of each chunk.
(e) Computing IDF values of references - output: IDF value for each legal regulation and chunk (each chunk defers to the IDF value of the document it belongs to).
(f) Extraction of keyphrases from references (optional) - output: list of keyphrases for each legal regulation and each chunk of judicial decision.

2. Retrieval
(a) Creating embedding of query - output: embedding vector of query - $v(q)$
(b) Determining semantic similarity between query and chunk embeddings - output: cosine similarity between query and chunk - $cos(v(q), v(c_i))$
(c) Extracting keyphrases from the query - output: list of keyphrases from the query - $KPP(q)$
(d) Finding matching keyphrases between query keyphrases and chunk keyphrases - output: intersection between $KPP(q)$ and $KPP(c_i)$
(e) Combining the matching keyphrases with the cosine similarity and to get a relevance score of each document of judicial decision - output: $score(d_i)$
(f) Enhancing relevance score of documents by using cosine similarity and shared keyphrases of references as well. (optional) - output: final enhanced score of each document.

3.2 Basic labels

For our purposes, let's define the task of document retrieval as returning a list of documents $R = r_1, ..., r_m$ from a predefined corpus $D == d_1, ..., d_n$ where for $1 \leq k \leq m \leq n$, $r_k \in D$ with the most significant relevance to a given query text q as well as provide a ranking for said documents where for a given $1 \leq a, b \leq m \leq n$ if $similarity(q, r_a) \leq similarity(q, r_b)$ then $score(r_a) \leq score(r_b)$.

The most common method of achieving this lies in using some kind of language model to create vector embeddings $v(t)$ where t is a natural language text we expect our language model to be able to encode. We have opted to use the *kinit/slovakbert-sts-stsb* model, described in [3]. Using said language model, we are able to encode both the query q and the documents in D so that $score(d_i) = \cos(v(q), v(d_i))$ where $1 \leq i \leq n$, $d_i \in D$ and $\cos(v(q), v(d_i))$ denotes the cosine similarity between the embedding of the given document and the query.

3.3 Reference and keyphrase extraction

To improve upon existing document retrieval methods, we work within the core text and use various metadata to get more precise and specific results. Such

metadata includes references to law articles and previous court decisions, extracted using rule-based methods specific to Slovak Legal texts described in our earlier work [8].

These references can be used to perform our keyphrase extraction methods. However, we will present an approach to keyphrase extraction that doesn't require references yet still offers improvement over reranking.

Our keyphrase extraction method is inspired by the one first described in [5].

3.4 Extraction of potential keyphrases from the source document and its individual chunks

First, we establish a set of candidate keyphrases we will work with. In our case, that would be a set of phrases collected from different dictionaries into a vocabulary that will be consistent across the whole database and the operations we will perform over it. Let's denote this set of phrases as $P{=}p_1, ..., p_l$. This set of keyphrases has been manually approved by lawyers as the collection of terms and categories they wish to search for. Alternatively, one could generate candidate keyphrases based on POS tag pattern, other semantic and statistical metrics or simply take all explicit n-grams to a given degree from the target text.

Most keyphrase extraction methods, including ours, rely on the presence of a given term in the text document and its frequency. Such approaches have the downside of only being able to evaluate the phrases that directly appear in the document. To this end, we propose embedding similarity between individual phrases, and those phrases that pass a certain threshold of cosine similarity will be considered adjacent terms.

Let adj_i denote the set of phrases $\{p_j \mid p_j \cos(v(p_i), v(p_j)) \geq T$ where p_i and $p_j \in P$ and T is the threshold value.

When calculating the presence of a term, we use that specific term and all of its adjacent terms, weighted by the cosine similarity to the given term.

Let PP^t denote the present phrases within the text t. Then, for every i in P let

$$PP_i^t = \begin{cases} 1, & \text{if } p_i \in t \\ \cos(v(p_i), v(p_j)), & \text{if } p_i \notin t, p_j \in t \text{ and } p_j \in adj_i \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Then, we calculate TF-IDF on every keyphrase from a given set of keyphrases, using a maximum document frequency of 50% to exclude domain-specific stopwords and use sublinear frequency, which replaces the standard TF for $1 + \log(TF)$ to prevent the metric from overscoring with high IDF that appear far too many times in the document. We calculate the sublinear TF and IDF scores using the adjacent phrases.

Thus, TF-IDF$_{\text{mod}}$ will be caluclated as:

$$\text{TF-IDF}_{\text{mod}}(p_i, t) = 1 + \log\left(\text{TF-IDF}(p_i, t) + \sum_{p_j \in adj_i} PP_i^t \cdot \text{TF-IDF}(p_j, t)\right) \quad (2)$$

However, statistical methods like TF-IDF even with calculating frequencies of synonyms and related terms based on embedding similarity often fail to capture semantic information. For that reason, we decided to combine it with a keyphrase extraction method based on semantic features. For this purpose, we have chosen, AttentionSeek, a scoring method based on the Self-Attention Masks generating the given embedding, described in [11]. According to this method, we use Huggingface to extract the Self-Attention Masks (SAM) of our transformer model where A^{lh} will represent the attention scores for layer l and head h, where A_i^{lh} represents the attention score for layer l, head h and phrase $p_i \in P$ We create a so-called hypothesis vector H where for a given text t, $H = PP^t$ where $H_i = PP_i^t$. The relevance of a given attention vector is then calculated as the matrix multiplication for the A^{lh} of a given SAM and H vectors.

$$S^{lh} = A^{lh} \cdot H \quad (3)$$

We can calculate the relevance of each SAM, represented by R^{lh} as the average relevance of its attention vectors.

$$R^{lh} = \frac{1}{n} \sum_{i=0}^{n} S_i^{lh} \quad (4)$$

The final attention vector B^t for text t is computed as a weighted average across all SAMs' attention vectors.

$$B^t = \sum_{\forall l,h} \sum_{\forall i} A_i^{lh} * S_i^{lh} * R^{lh} \quad (5)$$

After we have calculated both of these keyphrase extraction metrics, we can create the combined measurement, the keyphrase potential KPP_i^t for a single keyphrase $p_i \in P$ in a given text t which consists of multiplying its modified TF-IDF score with the attention vector.

$$KPP_i^t = \text{TF-IDF}_{\text{mod}}(p_i, t) * B^t(p_i) \quad (6)$$

3.5 Using keyphrases for document retrieval

Let $D = d_1, .., d_n$ be the corpus of all judicial decisions. In the previous subsection, we described a way of extracting and ranking potential keyphrases from a given document. However, the texts of judicial decisions are usually long and must be split into chunks. There are several models of document chunking to consider. We found that fixed-size chunking was relatively inefficient regarding

judicial decisions. Due to the varying case complexities, the length of each document segment and semantic whole varies greatly. When it came to semantic chunking, we've had much greater success, particularly when establishing breakpoints through standard deviation, as portions of legal documents are, by their nature, more or less structured than others. Therefore, it yields greater success if we don't just consider semantic dissimilarity to the previous chunk but rather a normal change in the measure of dissimilarity within the said chunk.

Let $C(d_i) = c_{i,1}, ..., c_{i,j_i}$ be a set of all chunks for document $d_i \in D$ where $\forall_{1 \leq a \leq n} \forall_{1 \leq b \leq j_a}$

The relevant keyphrases for a given chunk can be extracted using the keyphrase extraction algorithm described in the subsection above.

$$KPP_i^{c_{a,b}} = \text{TF-IDF}_{\text{mod}}(p_i, c_{a,b}) * B^{c_{a,b}}(p_i) \tag{7}$$

Similarly, our query q may be a phrase, a collection of phrases, or a reference legal text; therefore, we can extract the keyphrases for the legal text the same way.

$$KPP_i^{q} = \text{TF-IDF}_{\text{mod}}(p_i, q) * B^{q}(p_i) \tag{8}$$

We can then select the top t keyphrases with the highest KPP scores.

Now that we have extracted the keyphrases, we need to note that not every chunk of text contributes equally to the final result. Therefore, we create a vector embedding of each chunk $v(d_{a,b})$ and calculate the cosine similarity between it and the vector embedding of our query $v(q)$. We multiply this similarity score with the number of shared keyphrases divided by the total number of keyphrases q and $d_{a,b}$ share.

$$score(d_i) = \frac{1}{|C(d_i)|} \sum_{c_j \in C(d_i)} \left(\cos\left(v(c_j), v(q)\right) \times \text{match}(c_j, q)\right) \tag{9}$$

where

$$\text{match}(c_j, q) = \frac{1 + 2 \times |top_t(KPP^{c_j} \cap top_t(KPP^q)|}{1 + |top_t(KPP^{c_j})| + |top_t(KPP^q)|} \tag{10}$$

3.6 Enhancing keyphrase augmented document retrieval using extracted references

Now that we know how to extract the keyphrases from a singular document, we can do the same for references. Let $\mathsf{L} = l_1, \ldots, l_d$ be the collection of all citeable legal regulations. Then let $\mathsf{L}_{\mathsf{ref}}(\mathsf{i}) = (l_{i1}, ...l_{in_i}) \subseteq \mathsf{L}$ the set of all legal regulations referenced by decision chunk $c_i \in D$.

Since these are short and to-the-point texts, they do not require any chunking, and their keyphrases can be extracted directly using the KPP^t formula.

However, not all references contribute equally to the meaning of the decision. Therefore, we calculate the IDF scores across all legal regulation references, as

those cited more often tend to represent standardized formal procedures more than something uniquely related to the document's content. Therefore, we calculate the IDF score for each legal regulation through the whole corpus.

We also consider how relevant a given legal regulation is to the chunk of text by examining the cosine similarity of its text to the chunk it was cited from.

$$KPP_k^{l_{ij_i}} = \cos(v(c_i), v(l_{ij_i})) \times IDF(l_{ij_i}) \times \text{TF-IDF}_{\text{mod}}(p_k, l_{ij_i}) \cdot B^{l_{ij_i}}(p_k) \quad (11)$$

As for court decision references, we can use the same formula as described, including multiplication by the IDF score and cosine similarity with the chunk it was cited from, except this step now has to be done for every individual chunk and averaged

Let the and $\mathsf{D_{ref}(i)} = (\mathsf{d_{i1}}, ...\mathsf{d_{in_i}} \subseteq \mathsf{D}$ the set of all court decisions referenced by decision chunk $c_i \in \mathsf{C(d_i)}$. Let $C(d_{ij_i}) = c_{ij_i,1}, ..., c_{ij_i,n_ij_i}$ be the set of all chunks from document $d_{ij_i}, i_1 \leq ij_i \leq in_i$.

Then the keyphrase score of d_{ij_i} be calculated as

$$KPPref_m^{d_{ij_i}} = \frac{1}{|C(d_{ij_i})|} \sum_{c_{ij_i,k_{ij_i}} \in d_{ij_i}} \cos\left(v(c_i), v(c_{ij_i,k_{ij_i}})\right) \cdot KPP_m^{d_{ij_i}} \quad (12)$$

where $KPP_m^{d_{ij_i}}$ is calculated using equation 7.

Now that we have the keyphrase scores for the references calculated, we can perform an element-wise addition to the keyphrase scores extracted from the core document using equation 7. For a given keyphrase with the index of m where $p_m \in P$, we calculate the score the following way:

$$KPPfinal_m^{c_i} = KPP_m^{c_i} + \sum_{l_j \in L_{ref}(i)} KPP_m^{l_j} + \sum_{d_k \in D_{ref}(i)} KPPref_m^{d_k} \quad (13)$$

Using this new score of keyphrases, we can then calculate the retrieval score using equation 9.

4 Results

4.1 Dataset and evaluation

We have opted to test and evaluate our methods on 17164 labelled court decisions from Supreme Court of the Slovak Republic with manually annotated keyphrases. We created a set of all the possible keyphrases and collected their definitions, this set numbering 1500 phrases total. We used these definitions as our query for the document retrieval, labelling the documents that were annotated with the specific keyphrase the definition corresponded with as positive examples and those that weren't as negative examples and evaluated our results using the mean average precision metric, arranging them into a table.

BM-25	cos_full	cos_full + BM-25	cos_averaged	cos_max	keyphrase	keyphrase+ref
78.06	52.77	74.45	69.34	69.55	80.25	92.85

Table 1. The mean average score of our approaches compared to baseline methods.

In Table 1, we compare the mean average precision score of two of our approaches along with 5 baseline algorithms:

1. **BM-25** using the phrases from the query;
2. **cos_ful** cosine similarity between the query and the full document;
3. **cos_ful + BM-25** a metric combining the previous two approaches, given equal weight;
4. **cos_averaged** cosine similarity between the query and its individual chunks after splitting;
5. **cos_max** cosine similarity between the query and its most similar chunk;
6. **keyphrase** Keyphrase based search method without using references described in 3.5;
7. **keyphrase+ref** Keyphrase based search method that uses references described in 3.6.

As we can see from these results, the worst performance belongs to the cosine similarity between the query and the entire document text. We believe this is because judicial decisions are long texts containing many formal parts irrelevant to a given case, causing the embeddings to become muddled and lose their sensitivity. We performed slightly better when we split the document into segments and then calculated the average distance to said segments. Here, the relevant parts separated into specific chunks can increase the average. It seems as though whether we compare the average cosine similarity of the chunk embeddings with that of the query does not yield a significant difference. Still, taking all the other generic texts into account is counterproductive, so if we calculate its similarity to the best matching chunk, we see another performance improvement. Still, we can see that we can only go so far, relying solely on cosine similarity.

Surprisingly, the best baseline algorithm is BM-25, albeit that may be partially due to some of the specific standardized legal terms that were present in both the query and retrieved documents. The second-best results of the baseline algorithms are achieved by combining BM-25 with the entire document cosine similarity, but this is attributed to BM-25's specific suitability. The cosine similarity is more of a hindrance, as BM-25 achieves better results without it. Finally, our proposed methods achieved the best performance, with the one that uses the references outperforming the one that does not by a significant margin. Still, both offer a significant improvement over the baseline algorithms, even if at the cost of additional time and complexity. However this time and complexity is only an issue upon initial calculation and after such calculation is performed for newly inserted documents, queries are real-time with comparable retrieval speeds to most baseline algorithms.

4.2 Ablation study of our algorithm

To determine if and how much said complexity was necessary, we analyzed the performance of our algorithm when we omit certain parts and organized the results into Table 2. Here, we use the same task and the same metric of performance. The columns of the table concern the base keyphrase calculation method from an individual text.

There are six columns, but they are combinations of a few attributes. The first uses either exact matches or similarity matches. Using **exact matches** means that when we evaluate whether a given term is to be taken into account within a given text, we require that the term be present explicitly in the text. In this route, we replace PP_i^t in (1) with the simplified one, where the given term is given a value of 1 if it is explicitly present and given a value of 0 otherwise, ignoring any semantic adjacency to other terms. When we use a so-called **similarity match**, we use PP_i^t in (1) as described, giving a non-zero score to terms semantically similar to our term. These two alternatives are then combined with three possibilities for extracting keyphrases. The first is only using our modified **TF-IDF** for extracting keyphrases, as described in equation (2), leaving out the Attention-seeker score described in equation (5). The columns labeled **Attention-seeker** are the polar opposite, leaving out the calculation of the modified TF-IDF described in equation (2), using only the Attention-seeker algorithm described in (5). Finally, the columns labeled **combined score** use the combination of both of these methods as described in (7).

As for the rows, there are seven in total, but they are also made up of different combinations. The first variant **no references** means that we didn't use any of the extracted references and merely calculated the similarity score using 3.5 or one of its modifications and didn't perform any steps described in 3.6. Rows labeled **legal regulation references** only use the references to legal regulations and ignore referenced court decisions. In comparison, rows labeled **court decision references** take into account only referenced court decisions and not the references to legal regulations. Finally, rows labeled **all references** use both references to court decisions and legal regulations. The other modification concerns whether we calculate the IDF scores for the references themselves or not.

As we can see from this table, most of the extra steps we took offer some performance improvement, and the very best results are achieved when we utilize all the steps described in our algorithm. Looking at the columns, it becomes evident that the combined metric to extract keyphrases is the most effective. Considering not just exact matches but similar phrases, albeit with a reduced relevance, is generally beneficial. We can also see that out of the two algorithms that contribute to the combined score, Attention-seeker produces better results, which is to be expected, considering the simple nature and limitations of TF-IDF. However, it seems omitting it altogether wouldn't yield better results, as the highest score still belongs to the combined metric

If we look at the rows, we can notice that, in general, including references and calculating the IDF scores of the references is rather beneficial. The only

	TF-IDF (exact match)	TF-IDF (similarity match)	Attention-seeker (exact match)	Attention-seeker (similarity match)	combined score (exact match)	combined score (similarity match)
no reference	65.25	68.18	72.91	75.30	80.08	80.25
legal regulation references (no IDF)	67.26	69.97	76.04	79.29	84.39	87.30
legal regulation references + IDF	69.27	74.57	76.37	84.12	88.47	90.58
court decision references (no IDF)	65.73	68.48	73.71	78.60	81.92	82.26
court decision references + IDF	65.95	69.46	75.17	84.09	88.34	88.71
all references (no IDF)	69.54	70.97	75.53	80.65	81.98	86.14
all references + IDF	69.78	75.31	79.81	84.77	89.72	92.85

Table 2. Ablation study of our algorithm

arguably poorly performing metric is relying solely on court decision references, especially when we don't combine it with an IDF score. Depending on the column, sometimes it even offers worse performance than having no reference. This is likely because many cited decisions have more to do with a formal process, and we don't know which part of the decision is cited. Compared to that, the references to legal regulations seem much more useful, although combining legal regulation references and court decision references produces the best results.

5 Conclusion and Future Work

This paper presents an adjustable method that improves baseline document retrieval methods using keyphrase extraction methods. Our preliminary results suggest it could produce better results than many usual approaches. Our best approach, combining parameters from the texts of extracted references, obtained a mean average precision score of 92.85% on our evaluation task. Still, even the version not using the references achieved a performance of 80.25%. We also offer several ways to omit steps from our algorithm in case we don't have the necessary data for references or some of the steps are too computationally complex. In our future work, we aim to create a dataset to test our methods to retrieve the most relevant chunks instead of entire documents, a more difficult and relevant task in Information retrieval. We also wish to test it with more robust language models and evaluate our ranking of these documents against a reranking performed by an LLM.

Acknowledgments

This research for this paper was funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I05-03-V02-00049.

References

1. Hu, W., Zhao, S., Zhao, Q., Sun, H., Hu, X., Guo, R., Li, Y., Cui, Y., Ma, L.: Bert_lf: A similar case retrieval method based on legal facts. Wireless Communications and Mobile Computing **2022**(1), 2511147 (2022)
2. Novotná, T.: Document similarity of czech supreme court decisions. Masaryk University Journal of Law and Technology **14**(1), 105–122 (2020)
3. Pikuliak, M., Grivalský, Š., Konôpka, M., Blšták, M., Tamajka, M., Bachratý, V., Šimko, M., Balážik, P., Trnka, M., Uhlárik, F.: Slovakbert: Slovak masked language model. In: Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022. pp. 7156–7168. Association for Computational Linguistics (2022). https://doi.org/10.18653/V1/2022.FINDINGS-EMNLP.530, https://doi.org/10.18653/v1/2022.findings-emnlp.530
4. Pipitone, N., Alami, G.H.: Legalbench-rag: A benchmark for retrieval-augmented generation in the legal domain. arXiv preprint arXiv:2408.10343 (2024)
5. Szoplák, Z., Gurský, P., Varga, D.: Optimizing Keyphrase Extraction for Court Decisions Using Legal References (12 2024). https://doi.org/10.3233/FAIA241265
6. Tang, L., Clematide, S.: Searching for legal documents at paragraph level: Automating label generation and use of an extended attention mask for boosting neural models of semantic similarity. In: Proceedings of the Natural Legal Language Processing Workshop 2021, NLLP@EMNLP 2021, Punta Cana, Dominican Republic, November 10, 2021. pp. 114–122. Association for Computational Linguistics (2021), https://aclanthology.org/2021.nllp-1.12
7. Undavia, S., Meyers, A., Ortega, J.: A comparative study of classifying legal documents with neural networks. In: Proceedings of the 2018 Federated Conference on Computer Science and Information Systems, FedCSIS 2018, Poznań, Poland, September 9-12, 2018. Annals of Computer Science and Information Systems, vol. 15, pp. 515–522 (2018). https://doi.org/10.15439/2018F227, https://doi.org/10.15439/2018F227
8. Varga, D., Gojdič, M., Szoplák, Z., Gurský, P., Horvát, Š., Krajči, S., Antoni, L.: Extraction of legal references from court decisions. In: Proceedings of the 23rd Conference Information Technologies - Applications and Theory (ITAT 2023), Tatranské Matliare, Slovakia, September 22-26, 2023. CEUR Workshop Proceedings, vol. 3498, pp. 89–95. CEUR-WS.org (2023), https://ceur-ws.org/Vol-3498/paper11.pdf
9. Vitório, D., Souza, E., Martins, L., da Silva, N.F., de Carvalho, A.C.P.d.L., Oliveira, A.L., de Andrade, F.E.: Building a relevance feedback corpus for legal information retrieval in the real-case scenario of the brazilian chamber of deputies. Language Resources and Evaluation pp. 1–21 (2024)
10. Wiratunga, N., Abeyratne, R., Jayawardena, L., Martin, K., Massie, S., Nkisi-Orji, I., Weerasinghe, R., Liret, A., Fleisch, B.: Cbr-rag: case-based reasoning for retrieval augmented generation in llms for legal question answering. In: International Conference on Case-Based Reasoning. pp. 445–460. Springer (2024)
11. Z., E.D.L., Tang, C., Shimada, A.: Attention-seeker: Dynamic self-attention scoring for unsupervised keyphrase extraction (2024), https://arxiv.org/abs/2409.10907