Combining lexical and neural retrieval with longformer-based summarization for effective case law retrieva

Askari, A.A.; Verberne, S.V.; Alonso, O.; Marchesin, S.; Najork, M.; Silvello, G.

Combining lexical and neural retrieval with longformer-based summarization for effective case law retrieval

Arian Askari[1], Suzan Verberne[1]

[1]Leiden Institute of Advanced Computer Science, Leiden University

Abstract

In this paper, we combine lexical and neural ranking models for case law retrieval. In this task, the query is a full case document, and the candidate documents are prior cases that are potentially relevant to the current case. Most documents are longer than 1024 tokens, which makes retrieval and classification with Transformer-based models problematic. We create shorter query documents with different methods: term extraction, noun phrase extraction, entity extraction, and automatic summarization using Longformer-Encoder-Decoder (LED). We then combine the summaries with five different ranking models: a BM25 ranker, statistical language modelling, the Deep Relevance Matching Model (DRMM), a Vanilla BERT ranker, and a Longformer ranker. We optimised all models and combined the best lexical ranker with neural retrieval models using different ensemble classifiers. We evaluate our methods on the retrieval benchmarks from COLIEE'20 and COLIEE'21. We beat state-of-the-art models for case law retrieval with both benchmark sets. Our experiments show the importance of tuning lexical retrieval methods, summarizing query documents, and combining lexical and neural models into one ranker for effective case law retrieval. In addition, training and optimizing our rankers is much faster than passage-level retrieval models (a few hours compared to several days for training).

Keywords

Legal Information Retrieval, Query Summarization

1. Introduction

In countries with *common law* systems, finding supporting precedents to a new case, is vital for a lawyer to fulfill their responsibilities to the court. However, with the large amount of digital legal records – the number of filings in the U.S. district courts for total cases and criminal defendants was 544,460 in 2020[1] – it takes a significant amount of time for legal professionals to scan for specific cases and retrieve the relevant sections manually. Studies have shown that attorneys spend approximately 15 hours in a week seeking case law [1].

This workload necessitates the need for information retrieval (IR) systems specifically designed for the legal domain. The Competition on Legal Information Extraction/Entailment (COLIEE) is a workshop that has been organized since 2014 as a series of evaluation competitions related to case law [2]. COLIEE defines four tasks. In this paper, we address legal case retrieval (Task 1).

One of the challenges of case law retrieval in COLIEE'20 is that the input is a long case document with a median length of 2,815 words instead of a keyword query. Four approaches to this problem exist. The first is the use of unsupervised keyword extraction methods to create short queries from the long query document [3]. A variant is the unsupervised extraction of phrases or entities as query terms [4]. The second approach, proposed by Tran et al. [5], is to train a supervised phrase scoring model for n-gram phrases to select the phrases that are semantically closest to an expert-written summary. The third approach, proposed by Rossi and Kanoulas [6], is to use document summarization methods for creating shorter query documents. The fourth approach, successfully employed by Shao et al. [4] and Westermann et al. [7], is to analyze the documents on the level of individual paragraphs and then aggregate the paragraph scores in a document ranking.

In this paper, we use automatic summarization for creating query documents. We experiment with term extraction, noun phrase extraction, and supervised text summarizers. As opposed to prior work, we approach the task as an *abstractive* summarization problem. The current state of the art in abstractive summarization is the use of Transformer models [8, 9]. However, the input of pre-trained available models of these architectures is limited to 1024 tokens, and the majority of case law documents in our collection is longer than that. Beltagy et al. [10] proposed Longformer-Encoder-Decoder (LED),

DESIRES 2021 – 2nd International Conference on Design of Experimental Search Information REtrieval Systems, September 15–18, 2021, Padua, Italy

a.askari@liacs.leidenuniv.nl (A. Askari); s.verberne@liacs.leidenuniv.nl (S. Verberne)
https://www.universiteitleiden.nl/en/staffmembers/arian-askari (A. Askari); https://liacs.leidenuniv.nl/~verbernes/ (S. Verberne)
0000-0003-4712-832X (A. Askari); 0000-0002-9609-9505 (S. Verberne)

[1] https://www.uscourts.gov/statistics-reports/judicial-business-2020

which is a Transformer variant that supports much longer inputs. In this paper, we evaluate the effectiveness of LED for case law retrieval.

We evaluate summarization with multiple ranking models: probabilistic lexical ranking (BM25), statistical language modelling, the Deep Relevance Matching Model (DRMM), and Transformer-based architectures including the Vanilla BERT model from Contextualized Embeddings for Document Ranking (CEDR) [11]. Although we summarize the query documents, the lengths of the documents in the retrieval collection still causes problems for Transformer-based ranking architectures, which are limited to 512 tokens as the input length. To solve that problem, we implemented a Vanilla Longformer ranker [10], with 4096 tokens as the input length, in a pair-wise ranking setting similar to Vanilla BERT. After we have optimized each individual ranker, we experiment with ensemble methods that combine lexical rankers with neural rankers.

Our contributions are four-fold: (1) We deliver a finetuned Longformer-Encoder-Decoder (LED) for abstractive summarization of legal documents; (2) we deliver a pairwise Longformer ranker for long documents; (3) We show that summarizing query documents with LED improves all ranking models; (4) We show that the combination of a lexical ranker and a Vanilla BERT ranker in a simple ensemble classifier outperforms all baselines on COLIEE'20 and an optimized BM25 ranker with keyword queries beats the state-of-the-art models on COLIEE'21.

2. Related Work

2.1. Case law retrieval

Locke et al. [3] investigate query generation from legal decisions using unsupervised keyword extraction models. They find that the best performing model is Kullback-Leibler divergence for informativeness (KLI) [12] and that the automatically generated queries were more effective than the average Boolean queries from experts.

The majority of the work addressing case law retrieval takes place in the context of COLIEE, the Competition on Legal Information Extraction and Entailment [13, 2]. Rossi and Kanoulas [6] proposed a pairwise ranking model based on BERT. They apply automatic summarization using TextRank, to make the input document length suitable for use in the BERT-based ranker. The most successful team in COLIEE 2019 is Tran et al. [14, 5] ('JNLP'). They train a phrase scoring model that extracts n-gram phrases ($n \geq 5$) to summarize the documents. The weights are calculated based on the phrase score framework that was trained on COLIEE'18 summaries. The authors achieved the state-of-the-art result on COLIEE'19, thereby showing the importance of query document summarization.

In COLIEE'20, the two best-performing teams use paragraph-level analyses to cope with the challenge of long documents. Shao et al. [4] ('TLIR') participate with their method BERT-PLI [15], which models paragraph-level interactions. They combine BERT-PLI with lexical matching features using word-entity duet model. The features are different lexical rankers (BM25, probabilistic language modelling) – without optimization – on the full document content, and entities extracted by NLTK. They reach competitive results. The method by Westermann et al. [7] ('cyber') selects the top-30 candidate documents using a paragraph similarity score based on the universal sentence encoder, and then applies an SVM model to the TF-IDF representations of the query document and the candidate documents.

We use the supervised phrase-based summarization of Tran et al. [14, 5] as comparison for our summarization task, and the highest F-score obtained in COLIEE'20 by Westermann et al. [7] as comparison for our retrieval task.

2.2. Summarization of long documents

Kanapala et al. [16] and Van de Luijtgaarden [17] give an extensive overview of research on legal document summarization up to 2019. Here we focus on the recent abstractive summarization models.

Pre-trained encoder-decoder Transformer models (e.g. BART [8] and T5 [9]) have achieved strong results in abstractive summarization tasks. However, pre-trained models of these architectures are limited to texts that are shorter than 1024 tokens. Legal documents are commonly longer than that; 72% of the query documents in COLIEE'20 are. Recently, Beltagy et al. [10] proposed Longformer-Encoder-Decoder (LED), which is a Transformer variant that supports sequence-to-sequence tasks for longer documents (up to 16k tokens). The authors show that LED outperforms the state-of-the-art models on the arXiv dataset. In this paper, we evaluate the effectiveness of LED for case law retrieval.

2.3. Transformer-based document ranking

The typical approach in neural IR is two-step retrieval, where a first set of documents is retrieved using a traditional ranker (e.g. BM25) and those document are reranked by a neural model that is trained on the relevance assessments [18, 11, 19].

MacAvaney et al. [11] propose a joint approach that integrates the classification vector of BERT into existing neural models like the Deep Relevance Matching Model (DRMM), resulting in Contextualized Embeddings for Document Ranking (CEDR). They fine-tune a pretrained

BERT model with a linear combination layer stacked atop the [CLS] token as the Vanilla BERT ranker with pairwise cross-entropy loss and the Adam optimizer. The authors demonstrate that Vanilla BERT and CEDR outperform the state-of-the-art baselines in ad-hoc rankings. However, local-interaction neural ranking architectures like DRMM are not scalable to long documents or need heavy interaction between word pairs in query and document. Therefore, we will compare our method to the Vanilla BERT ranker.

More recent work has addressed the challenges of ranking for long documents. Hofstätter et al. [20] propose a local attention Transformer model that uses a moving window over the document terms and for each term attends only to other terms in the same window. They obtain results significantly better than other state-of-the-art models on the TREC Deep learning track. Sekulić et al. [19] take a full-document approach by training a Longformer model for ranking in ad-hoc retrieval. The results they report on the MS MARCO set are low compared to the leaderboard results. Our Longformer ranker is similar to [19], but instead of implementing the ranker as a one-versus-all classifier, we train and evaluate it in a pair-wise setting and we evaluate it for case law retrieval.

3. Methods

3.1. Summarization

We experiment with three approaches to creating shorter query documents: (1) term extraction, (2) noun phrase or entity extraction, and (3) abstractive summarization.

Summarization through term extraction We adopted Kullback-Leibler divergence for Informativeness (KLI), similar to Locke et al. [3] in the implementation of Verberne et al. [21]. For each term t in a query document, we computed the KLI score:

$$KLI(t) = P(t|D) \times \log \frac{P(t|D)}{P(t|C)} \qquad (1)$$

where $P(t|D)$ is the probability of t in the query document D and $P(t|C)$ is the probability of t in a background language model. We use all candidate documents as the background collection to compute $P(t|C)$. We only consider unigrams as terms and we lowecase them.[2] We then selected the top-10% of the total number of terms in the document with the highest KLI score as a the query.

Entity and phrase extraction Inspired by Shao et al. [4], we used NLTK[3] to extract noun phrases and named entities from the content of query documents and candidate documents and used the extracted strings entities as the representation documents (see Section 4.4 for the combinations we experimented with).

Abstractive summarization We experimented with the pre-trained LED model[4] of Beltagy et al. [10] which can process documents up to 16k tokens as input. Also, we fine-tuned LED on the COLIEE'18 dataset, in which more than 80% of documents have summaries.[5] After removing duplicates, $6,257$ unique documents are left for which a summary is available. For evaluation purposes, we trained the model in a k-fold cross-validation setting ($k = 10$) for one epoch per fold, each with batch size 1. We kept the other hyperparameters (optimizer, dropout, weight decay) identical to [10] and set the global attention on the first <s> token. We only summarized query documents with LED for the lexical rankers since they do not have limitation for input length. On other hand, since Transformer-based neural models are limited in input length for the candidate document content, we also experiment with summarizing candidate documents besides query documents for the best Transformer-based neural model in our experiments.

3.2. Ranking models

As introduced in section 3.2.1, we rank the 200 candidate documents for each query document in COLIEE'20 with multiple retrieval models. We optimise the hyperparameters of each method on a validation set (see Section 4.3). In the following, we will introduce each neural ranker that we use for legal case retrieval.

3.2.1. Lexical rankers

BM25 We indexed the COLIEE'20 collection with ElasticSearch. The collection has 200 candidate documents for each query that need to be ranked. We used BM25 with the default parameter values $k = 1.2$ and $b = 0.75$, as well as with optimized hyperparameter values.

Language Modelling We used the built-in similarity functions of Elasticsearch for the implementation of Language Modelling (LM) with two different smoothing: Dirichlet smoothing and Jelinek Mercer (JM) smoothing. We only report the results for JM smoothing since we got

[2] We tried to extract n-gram phrases (2<n<=5) with $\gamma = 0.8$ but obtained the best results with unigrams.

[3] https://www.nltk.org/api/nltk.chunk.html?#nltk.chunk.util.tree2conlltags and https://www.nltk.org/api/nltk.chunk.html#module-nltk.chunk.named_entity

[4] https://huggingface.co/allenai/led-large-16384-arxiv

[5] In COLIEE'19 and COLIEE'20 the large majority (82%) of the candidate cases do not have a summary.

similar results by these two smoothing methods. We also optimised the hyperparameter value (λ) for Language Modelling with Jelinek Mercer smoothing (LM JM).

3.2.2. Neural rankers

Deep Relevance Matching Model (DRMM) As the DRMM architecture [18] is based on a local interaction input matrix, only a limited query length is possible. We took the top 10% of query terms sorted by the KLI score for each query document. Then, we selected the average resulting number of terms, 70, as the maximum length of query. Thus, we used the top-70 query terms as query in DRMM. For calculating cosine similarity in the local interaction matrix, we trained word2vec on the query candidates' texts as suggested by the DRMM authors. Furthermore, we optimize the network configuration of DRMM to find the best combination of layers and neurons in legal case retrieval on COLIEE (see details in Section 4.3).

Vanilla BERT ranker For Vanilla BERT, we fine-tuned a pre-trained BERT model (BERT-Base, Uncased) with a linear combination layer stacked atop the classifier [CLS] token on the COLIEE dataset in pairwise cross-entropy loss setting using the Adam optimizer. We used the implementation of MacAvaney et al. [11] (CEDR). We represent the query as sentence A and the document as sentence B in the BERT input:

“[CLS] query document [SEP] candidate document [SEP]"

We truncate the query and candidate document text since the BERT tokenizer is limited to 512 tokens.

Vanilla Legal BERT ranker For Vanilla Legal BERT, we used LEGAL-BERT [22] which was pre-trained on legal data.

Vanilla Longformer ranker Since the input length is limited in Vanilla BERT, we implemented the Vanilla Longformer in CEDR [11] as a ranker which can receive 4096 tokens instead of 512 and has more chances to work effectively than Vanilla BERT. In Longformer, the [CLS] and [SEP] tokens are replaced by the tokens `<s>`, and `</s>` respectively. As suggested in the Longformer paper [10] we calculate the loss based on the `<s>` token with the addition of global attention to the `<s>` token. Inspired by Sekulić et al. [19] we feed the query document as sequence A and the candidate document as sequence B to the tokenizer, which yields the following input to the model:

“`<s>` query document `</s>` candidate document `</cls>`".

Our code is integrated with Vanilla BERT in CEDR [11], and is available for future work.[6]

3.2.3. Ensemble models

For combining the advantages of neural rankers and lexical rankers in one integrated system, we train ensemble models that take the scores of multiple rankers as features. We experiment with three different classifiers for this purpose: SVM with a linear kernel, SVM with an RBF kernel, Naive Bayes, and Multi Layer Perceptron (MLP). We experiment with different combinations of rankers' scores to find the best combination of rankers for having an effective ranking.

4. Experiments and results

4.1. Data

For our experimental evaluation, we work with data from the COLIEE competitions in 2018, 2020, and 2021.

The COLIEE'18 data contains human-written summaries of the case documents, which we use for the training evaluation of our summarization models (Section 4.2). For our retrieval experiments (Section 4.3), we use data from COLIEE'20 and '21.[7] The Federal Court of Canada provided case laws with metadata for task 1. The metadata contains references to the noticed cases that are the golden relevance labels for the query document. In COLIEE'20, there is a pool of 200 candidates for each query document and the competitors should re-rank a limited number of documents per query. The pool of candidates includes the noticed cases and non-relevant candidates, which are selected randomly. In contrast, in COLIEE'21, the whole collection should be considered per query without having a pool of candidates. This difference makes task 1 in COLIEE'21 more difficult than in COLIEE'20.

We use the COLIEE'20 data to evaluate all ranking models and ensembles described in Section 3. In the COLIEE'20 data, there are 520 query documents in the train set and 130 in the test set; with 104,000 candidate documents in the train set and $26,000$ in the test set. The average length of the documents in the test set is 3,232 words, with outliers upto $10,827$. After we have found the best-performing rankers and ensemble, we evaluate those on the COLIEE'21 data and compare the results to the best results reported in the competition. In the COLIEE'21 data, there are 650 query documents in the train set and 250 in the test set, with $4,415$ documents as candidate documents for both train set and test set.

[6] https://anonymous.4open.science/r/vanilla_longformer-D552/README.md

[7] https://sites.ualberta.ca/~rabelo/COLIEE2020/ and https://sites.ualberta.ca/~rabelo/COLIEE2021/

Table 1
Summarization results in terms of ROUGE for COLIEE'18. Summary length is 10% of the original text

Model\Metric	ROUGE-1			ROUGE-2			ROUGE-SU6		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
JNLP [5]	0.482	**0.409**	0.405	0.186	**0.152**	0.152	0.258	**0.199**	0.167
Pre-trained LED on arXiv [10]	**0.634**	0.164	0.260	**0.299**	0.072	0.116	**0.332**	0.078	0.127
Fine-tuned LED	0.620	0.304	**0.408**	0.295	0.138	**0.188**	0.319	0.147	**0.201**

The average length of the candidate documents is $1,274$, with outliers upto $76,818$ words.

4.2. Summarization experiments

We set the local attention window size to 512 tokens. To limit memory use, we use gradient checkpointing and set the input size in training to 8,192 tokens which covers more than 86% of COLIEE'18 documents completely (the longer documents are truncated at 8,912 tokens). We set the maximum length to generate a summary for an unseen document as 10% of the length of the original text.

We evaluate our fine-tuned LED to the pre-trained LED [10], and compare it to the summarizer of Tran et al. [5] (JNLP) on the COLIEE'18 data. Table 1 shows that our fine-tuned summarizer outperforms the baseline in terms of F-measure for ROUGE-1, ROUGE-2 and ROUGE-SU6 on the COLIEE'18 dataset. The pre-trained LED summarizer obtains the highest precision-ROUGE scores, and the JNLP baseline the highest recall-ROUGE scores.

4.3. Retrieval experiments

As the validation set for optimizing the rankers we use a held-out subset (10% of the training set). We optimize the rankers for each document representation. In the text below, we refer to noun phrase representations as Q NP and C NP for the query/candidate documents respectively, and to entity representations as Q Entities and C Entities. We use Precision@k, Recall@k and F-measure@k as evaluation metrics, following the COLIEE evaluation mechanism.[8] We select the best cut-off k for each method based on the validation set and use that on test set.

BM25 We found $k = 6$ as the optimal cut-off for ranking with the whole text, and $k = 4$ for summarized input. For the optimization we searched the following grid: $b \in \{0, 0.1, 0.2, \cdot, 1\}$ and $k1 \in \{0, 0.1, 0.2, \cdot, 3\}$. For BM25 with KLI the best parameters were $b = 0.9$, and $k1 = 2.8$.[9]

LM We found $k = 6$ as the optimal cut-off for all variants of LM JM. For the optimization, we searched the following values: $\lambda \in \{0, 0.1, 0.2, \cdot, 1\}$.[10]

DRMM We optimize the word2vec model and also tune the network configuration (e.g., numbers of layers and hidden nodes) on the validation set. $k = 6$ is the optimal cut-off value. We trained six word2vec and fasttext models for three configurations from the literature [18, 23, 24]. We also used the pre-trained word2vec on google-news and pre-trained fasttext on Wikipeda. We found that word2vec which was pre-trained according to the DRMM configuration gave the best results. For the network configuration, we use a four-layer architecture throughout all experiments, i.e., one histogram input layer (30 nodes), two hidden layers in the feed-forward matching network (128 nodes for both layers), and one output layer (1 node) with the term gating network for the final matching score. We set the maximum query length to 70 tokens as explained in Section 3.2.2.

Vanilla BERT ranker We truncate the documents such that the concatenated query document (truncated at 100 words), candidate document, and the separator tokens do not exceed 512 tokens. We re-rank the top-30 BM25 results. Since the $R@30$ is about 95% for BM25, our ranker has the possibility to achieve 95% recall while still having a reasonable runtime. $k = 6$ was the optimal cut-off for the arXiv-LED summarizer queries, and $k = 4$ for the fine-tuned LED summarizer queries. We train each model for 100 epochs, each with 32 batches of 16 training pairs, with the initial learning rate of $3 * 10^{-5}$, followed by a power 3 polynomial decay.

Vanilla Longformer ranker The local window size is set to 512. We fine-tune the pre-trained Longformer using pairwise hinge loss. Positive and negative training documents are selected from the relevance judgments. We truncate the document such that the sequence of the concatenated query document (summary), candidate document, and the separator tokens do not exceed 4,096

[8] See https://sites.ualberta.ca/~rabelo/COLIEE2020/

[9] We found $b = 0.6, 0.8, 0.8$ and $k1 = 1.8, 1.4, 1.4$ as the best parameters for BM25 (Q NP C NP, Q words C NP, and Q NP C words) respectively.

[10] We found $\lambda = 0.1$ as the best parameters for LM JM with KLI. We found $\lambda = 0.1, 0.6, 0.4$ as the best parameters for LM JM (Q entities D entities, Q words D entities, and Q entities D words) respectively.

Table 2

Lexical retrieval results for the ranking of 200 candidate documents in COLIEE'20. Q refers to query content and C refers to candidate document content. SummaryQ means that the summary of the query document is used as query. NP refers to the extracted noun phrases (NP) using NLTK. Q/C NP means that the extracted noun phrases from query and candidate document are used as the query and candidate document content in the ranker's input.

Method	Extractor/Sumarizer	P %	R %	F1 %
BM25	original text	47.69	67.48	56.06
BM25 optimized	original text	57.31	59.15	58.21
BM25	KLI (1-gram)	58.85	62.28	60.51
BM25 optimized	KLI (1-gram)	**67.00**	61.17	**63.95**
BM25	summaryQ arXiv-LED	48.65	**69.03**	57.07
BM25 optimised	summaryQ arXiv-LED	54.20	64.91	59.07
BM25	summaryQ fine-tuned LED	49.72	68.59	57.65
BM25 optimised	summaryQ fine-tuned LED	55.71	63.18	59.21
BM25 optimized	Q entities & C words	62.05	50.62	55.75
BM25 optimized	Q words & C entities	48.77	58.89	53.35
BM25 optimized	Q NP & C NP	55.77	58.05	56.88
LM JM	original text	46.54	62.25	53.26
LM JM	KLI (1-gram)	58.00	61.85	59.86
LM JM optimized	KLI (1-gram)	66.24	60.28	63.11
LM JM	arXiv-LED	49.20	68.55	57.28
LM JM	fine-tuned LED	49.68	68.44	57.57
LM JM optimized	Q NP & C words	61.94	50.30	55.51
LM JM optimized	Q words & C NP	47.65	57.94	52.29
LM JM optimized	Q NP & C NP	55.19	57.67	56.40

tokens. We again re-rank the top-30 BM25 results. for the arXiv-LED summarizer queries, and $k = 4$ for the fine-tuned LED summarizer queries. We use the same training configuration as for Vanilla BERT.

Ensemble classifier We used the Scikit-learn [25] library for training the ensemble classifiers and kept the hyperparameters as default. We used the classifier prediction (relevant/non-relevant) to obtain the returned set of documents that we evaluate. Note that the cut-off parameter k is not needed in this approach because the final ranking only includes documents that are predicted relevant by classifier.

4.4. Retrieval results

COLIEE'20 results The ranking results for the lexical and neural ranking models are shown in Table 2 and Table 3, respectively. The best lexical ranker is BM25, but LM JM with KLI is very close. The best neural ranker in terms of recall and F1 is DRMM. The best single ranker overall is BM25 with an F1 score of 63.95%. The highest precision is obtained by BM25 with KLI queries and the highest recall with BM25 with arXiv-LED. This indicates that lexical matching is important for this dataset. Table 2 also shows the (mostly positive) effect of optimizing the parameters of BM25, an effort that is not taken by most of the COLIEE participants.

The results of the ensemble models are shown in Table 4. With our ensemble models we improve over the best benchmark result (the Cyber team) by a large margin. The best ensemble model in terms of F1 on the validation set is a combination of the best BM25 ranker (BM25-optimised + KLI) and the second-best neural ranker Vanilla BERT (Vanilla BERT + SummaryQ (fine-tuned LED)). This indicates that BERT can add more to the combination with BM25 than the best neural model DRMM can.

Table 3

Neural retrieval results for the ranking of 200 candidate documents in COLIEE'20. Q refers to query content and C refers to candidate document content. SummaryQ/SummaryQC means that the summary of the query/both query and candidate document (generated by fine-tuned LED) are used as the content in the ranker's input.

Method	Extractor/Sumarizer	P %	R %	F1 %
DRMM	original text	27.05	37.16	31.31
DRMM	KLI (1-gram)	47.95	**67.43**	**56.05**
Vanilla BERT	original text	36.92	50.16	42.53
Vanilla BERT	summaryQ arXiv-LED	40.26	57.02	47.19
Vanilla BERT	summaryQ fine-tuned LED	**49.23**	62.22	54.96
Vanilla BERT	summaryQC fine-tuned LED	29.10	45.50	35.49
Vanilla Legal BERT	summaryQ fine-tuned LED	46.77	61.54	53.14
Vanilla Legal BERT	summaryQC fine-tuned LED	40.26	58.34	47.64
Longformer	original text	30.38	39.00	34.15
Longformer	summaryQ arXiv-LED	35.10	43.40	38.81
Longformer	summaryQ fine-tuned LED	39.04	44.20	41.46

Table 4

Results of ensemble classifiers and comparison with the Cyber team which is the best team on COLIEE'20 [7]. Precision and recall are not reported for the COLIEE'20 best result

Method	Features	P %	R %	F1 %
Naive Bayes	Best BM25 & best Vanilla BERT	**72.95**	83.66	**76.02**
SVM linear	Best BM25 & best Vanilla BERT	71.74	**83.77**	74.61
SVM RBF	Best BM25 & best Vanilla BERT	70.11	83.24	72.37
MLP	Best BM25 & best Vanilla BERT	70.68	83.46	73.20
Cyber	(first team in COLIEE'20)	-	-	67.74

Table 5

Retrieval results for the ranking of documents of COLIEE'21. Precision and recall are not reported for the COLIEE'21 best result. k is the optimal cut-off value for each method on this data.

Method	Extractor/Sumarizer	P %	R %	F1 %
BM25	original text ($k = 7$)	7.77	19.59	11.13
BM25	KLI (1-gram) ($k = 6$)	9.83	19.80	13.13
BM25 optimized	KLI (1-gram) ($k = 4$)	**17.00**	**25.36**	**20.35**
Vanilla BERT	fine-tuned LED ($k = 7$)	2.11	5.46	3.04
TLIR	(first team in COLIEE'21)	-	-	19.17

COLIEE'21 results For evaluating the generalizability of our results, we evaluated the best methods on the COLIEE'21 data. As explained in Section 4.1, the COLIEE'21 task is more difficult than the COLIEE'20 task because it requires retrieval from a full document collection instead of re-ranking 200 documents. We optimised the BM25 parameters and the cut-off value k on the COLIEE'21 validation set. As shown in Table 5, we beat the state-of-the-art result (TLIR team) with the optimized BM25 ranker. The poor result of Vanilla BERT on COLIEE'21 shows us that neural retrieval has more challenges for ranking the whole collection than with re-ranking the top-200. We also applied ensemble classifiers combining BM25 and Vanilla BERT but they could not improve the effectiveness; we suppose that it is caused by the lower performance of Vanilla BERT on COLIEE'21.

5. Discussion

The effect of summarization The results show that summarizing the query document improves all rankers. This holds for the KLI term extraction, noun phrase extraction, and the LED summarizer. This shows the importance of summarization for making the query documents shorter. Our results show that the best summarizing methods for neural ranking and lexical ranking are LED (fine-tuned) and keyword extraction (KLI) respectively. However, summarizing the candidate documents does not improve the neural ranking performance. Another observation is that fine-tuning the summarizer improves the ranking in terms of F-measure for all rankers. We see the largest effect from summarization for Vanilla BERT.

Analysis of unexpected results One unexpected result is that our Longformer ranker does not outperform the Vanilla BERT ranker. We speculate that this is because longformer receives more tokens as the input learning, which makes it more difficult to estimate the relevance between document and query, while the size of the training set is relatively small: In COLIEE'20, we have only $2,680$ relevant labels and Longformer could not converge during training – it did not find the optimal loss after 100 epochs. For future work, we will further pre-train Longformer on legal documents. This requires a GPU with 32GB Ram which is not easily accessible.

Our results also show that DRMM could not beat BM25 for case law retrieval. Some prior work has also indicated that for some datasets, DRMM is close to BM25 in quality and that sometimes BM25 works better than DRMM [26, 27, 28], especially when BM25 is properly optimized.

The third unexpected result is that the best Vanilla Legal BERT (summaryQ) does not outperform the best Vanilla BERT model. We suppose this can be related to language similarity between COLIEE cases and part of the data that BERT Base was trained on (Wikipedia, and more than thousand thousands books) because cases in COLIEE contain stories of applicant's lives and this is almost the bigger part of each document.

Analysis weights of classifiers As suggested in [29], we interpret the importance of Vanilla BERT feature in the ensemble classifiers based on the coefficient value in fitted SVM (linear). For Vanilla BERT the coefficient is higher (0.56) than BM25's coefficient (0.43).

Analysis hyperparameters of BM25 The optimal value for b in the literature is between $0.3 - 0.9$ [30, 31, 32] and we found $b = 0.9$ for the optimised BM25 with KLI. There are documents in COLIEE that, because of their length, contain multiple topics, and it was suggested before that documents that includes a variety of topics benefit from using a larger b so that unrelated topics to a user's search are penalized.[11] The normal range for $k1$ is between 0 and 3 and for long documents that contain diverse information the $k1$ should tend to larger numbers [32]. We found the optimal value for $k1 = 2.8$ for BM25 with KLI which makes sense because of the long documents in COLIEE.

Using noun phrases or named entities We experimented with noun phrases and named entities as document representation, inspired by Shao et al. [4]. Our experiments show that the effectiveness of using named entities instead of original content is much lower than of using noun phrases: The F1 score for BM25 optimised + Q entities C entities is 22.71% while F1 for BM25 optimised + Q NP C NP is 56.88%. Although the use of named entities was suggested in prior work, they do not play an important role in case law documents, at least not so much that using them as document representations leads to effective retrieval.

Future work Some prior work has obtained good results with passage-level analysis for long document ranking [24, 33]. We think it is a promising direction to combine these passage-level with document-level methods to design a more effective legal case retrieval system for future work. One challenge is that this approach is computationally expensive since each paragraph of the query document needs to be compared with each paragraph of the candidate documents. We have query documents with up to 1,139 paragraphs in COLIEE. For future work we will focus on combining lexical document retrieval with efficient paragraph-level retrieval. We will also evaluate how we can use COGLTX [5] to recognize the important sentences from the query cases and document cases without having the limitation in length as a pre-processing step. Inspired by [33], another direction is working on neural models for legal case retrieval on ranking instead of re-ranking because in coliee'21 there are $4,415$ candidates per query (whole collection).

[11] https://www.elastic.co/blog/practical-bm25

6. Conclusions

In this paper, we addressed the challenge of long documents in case law retrieval. We experimented with the Longformer-Encoder-Decoder (LED) for abstractive summarization of case law documents in the COLIEE benchmark data. The fine-tuned LED outperforms the 2019 baseline in terms of F-measure for all three ROUGE metrics.

Second, we implemented a pairwise Longformer ranker for long documents and compared it to four other ranking models on the COLIEE'20 benchmark data: BM25, LM, DRMM, and Vanilla BERT. We found however that BM25 outperforms all neural rankers on this task and that the Longformer ranker is outperformed by BERT.

Third, we evaluated the merits of query document summarization for the BM25, BERT and Longformer rankers. We found that summarizing the query document improves the quality of each of the rankers compared to using the original document. For BM25, we also compared the LED-summary to statistical query term extraction (KLI), and we found that summarization gives a higher recall, but term extraction gives a higher precision.

Fourth, we showed the effectiveness of combining an optimised BM25 ranker and a BERT ranker, outperforming the state of the art on two benchmark sets. We conclude that retrieval for long query documents in legal case retrieval can be helped by optimising lexical models, automatic summarization, and a combination of both.

References

[1] S. A. Lastres, Rebooting legal research in a digital age, 2015.

[2] J. Rabelo, M.-Y. Kim, R. Goebel, M. Yoshioka, Y. Kano, K. Satoh, COLIEE 2020: Methods for Legal Document Retrieval and Entailment, 2020. URL: https://sites.ualberta.ca/~rabelo/COLIEE2021/COLIEE_2020_summary.pdf.

[3] D. Locke, G. Zuccon, H. Scells, Automatic query generation from legal texts for case law retrieval, in: Asia Information Retrieval Symposium, Springer, 2017, pp. 181–193.

[4] Y. Shao, B. Liu, J. Mao, Y. Liu, M. Zhang, S. Ma, Thuir@ coliee-2020: Leveraging semantic understanding and exact matching for legal case retrieval and entailment, arXiv preprint arXiv:2012.13102 (2020).

[5] V. Tran, M. Le Nguyen, S. Tojo, K. Satoh, Encoded summarization: summarizing documents into continuous vector space for legal case retrieval, Artificial Intelligence and Law 28 (2020) 441–467.

[6] J. Rossi, E. Kanoulas, Legal information retrieval with generalized language models, Proceedings of the 6th Competition on Legal Information Extraction/Entailment. COLIEE (2019).

[7] H. Westermann, J. Šavelka, K. Benyekhlef, Paragraph similarity scoring and fine-tuned BERT for legal information retrieval and entailment., in: COLIEE 2020, 2020.

[8] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, arXiv preprint arXiv:1910.13461 (2019).

[9] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, arXiv preprint arXiv:1910.10683 (2019).

[10] I. Beltagy, M. E. Peters, A. Cohan, Longformer: The long-document transformer, arXiv preprint arXiv:2004.05150 (2020).

[11] S. MacAvaney, A. Yates, A. Cohan, N. Goharian, Cedr: Contextualized embeddings for document ranking, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 1101–1104.

[12] T. Tomokiyo, M. Hurst, A language model approach to keyphrase extraction, in: Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment, 2003, pp. 33–40.

[13] J. Rabelo, M.-Y. Kim, R. Goebel, M. Yoshioka, Y. Kano, K. Satoh, A summary of the coliee 2019 competition, in: JSAI International Symposium on Artificial Intelligence, Springer, 2019, pp. 34–49.

[14] V. Tran, M. L. Nguyen, K. Satoh, Building legal case retrieval systems with lexical matching and summarization using a pre-trained phrase scoring model, in: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law, 2019, pp. 275–282.

[15] Y. Shao, J. Mao, Y. Liu, W. Ma, K. Satoh, M. Zhang, S. Ma, BERT-PLI: Modeling Paragraph-Level Interactions for Legal Case Retrieval, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, ????

[16] A. Kanapala, S. Pal, R. Pamula, Text summarization from legal documents: a survey, Artificial Intelligence Review 51 (2019) 371–402.

[17] N. Van de Luijtgaarden, Automatic Summarization of Legal Text, Master's thesis, Utrecht University, 2019. URL: https://dspace.library.uu.nl/handle/1874/384802.

[18] J. Guo, Y. Fan, Q. Ai, W. B. Croft, A deep relevance matching model for ad-hoc retrieval, in: Proceedings of the 25th ACM international on conference on information and knowledge management, 2016,

pp. 55–64.

[19] I. Sekulić, A. Soleimani, M. Aliannejadi, F. Crestani, Longformer for ms marco document re-ranking task, arXiv preprint arXiv:2009.09392 (2020).

[20] S. Hofstätter, H. Zamani, B. Mitra, N. Craswell, A. Hanbury, Local self-attention over long text for efficient document retrieval, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 2021–2024.

[21] S. Verberne, M. Sappelli, D. Hiemstra, W. Kraaij, Evaluation and analysis of term scoring methods for term extraction, Information Retrieval Journal 19 (2016) 510–545.

[22] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, I. Androutsopoulos, Legal-bert: The muppets straight out of law school, arXiv preprint arXiv:2010.02559 (2020).

[23] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics 5 (2017) 135–146.

[24] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[26] J. Frej, P. Mulhem, D. Schwab, J.-P. Chevallet, Learning term discrimination, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1993–1996.

[27] I. Chios, S. Verberne, Helping results assessment by adding explainable elements to the deep relevance matching model, 2020. URL: https://ears2020.github.io/accept_papers/2.pdf.

[28] J. Frej, D. Schwab, J.-P. Chevallet, Mlwikir: A python toolkit for building large-scale wikipedia-based information retrieval datasets in chinese, english, french, italian, japanese, spanish and more (????).

[29] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of machine learning research 3 (2003) 1157–1182.

[30] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, C. Burges, Optimisation methods for ranking functions with multiple parameters, in: Proceedings of the 15th ACM international conference on Information and knowledge management, 2006, pp. 585–593.

[31] A. Trotman, A. Puurula, B. Burgess, Improvements to bm25 and language models examined, in: Proceedings of the 2014 Australasian Document Computing Symposium, 2014, pp. 58–65.

[32] A. Lipani, M. Lupu, A. Hanbury, A. Aizawa, Verboseness fission for bm25 document length normalization, in: Proceedings of the 2015 International Conference on the Theory of Information Retrieval, 2015, pp. 385–388.

[33] H. Zamani, M. Dehghani, W. B. Croft, E. Learned-Miller, J. Kamps, From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing, in: Proceedings of the 27th ACM international conference on information and knowledge management, 2018, pp. 497–506.